**PUBLIC**
SAP HANA Platform 2.0 SPS 07
Document Version: 1.0 – 2023-04-04

# SAP HANA Predictive Analysis Library (PAL)

© 2026 SAP SE or an SAP affiliate company. All rights reserved.

# Content


**1 SAP HANA Predictive Analysis Library (PAL). . . . 7**

**2 Getting Started with PAL. . . . 8**

2.1 Prerequisites. . . . 8

2.2 Application Function Library (AFL). . . . 8

2.3 Security. . . . 9

2.4 Checking PAL Installation. . . . 9

2.5 Restricting CPU and Memory Usage. . . . 11

**3 Calling PAL Procedures. . . . 12**

3.1 Switching Between the PAL Algorithms with the Same Datatset. . . . 18

3.2 Calling PAL Procedures in Parallel with Hint PARALLEL_BY_PARAMETER_PARTITIONS. . . . 19

3.3 Calling PAL Procedures in Parallel with Hint PARALLEL_BY_PARAMETER_VALUES. . . . 21

3.4 Calling PAL Procedures in Parallel with MAP_REDUCE. . . . 30

3.5 Calling PAL Procedures in Parallel with Operator MAP_MERGE. . . . 36

3.6 State Enabled Real-Time Scoring Functions. . . . 41

3.7 Exception Handling. . . . 59

**4 Using PAL in SAP HANA AFM. . . . 61**

**5 PAL Procedures. . . . 63**

5.1 Clustering Algorithms. . . . 75

Accelerated K-Means. . . . 75

Affinity Propagation. . . . 84

Agglomerate Hierarchical Clustering. . . . 90

Anomaly Detection. . . . 99

Cluster Assignment. . . . 106

DBSCAN. . . . 111

Gaussian Mixture Model (GMM). . . . 117

Geometry DBSCAN. . . . 125

K-Means. . . . 131

K-Medians. . . . 143

K-Medoids. . . . 150

Self-Organizing Maps. . . . 157

Slight Silhouette. . . . 163

Spectral Clustering. . . . 167

Incremental Clustering on SAP HANA Streaming Analytics. . . . 173

5.2 Classification Algorithms. . . . 174

Area Under Curve (AUC). . . . 174
Confusion Matrix. . . . 183
Decision Trees. . . . 187
Hybrid Gradient Boosting Tree. . . . 203
KNN. . . . 236
Linear Discriminant Analysis. . . . 265
Logistic Regression (with Elastic Net Regularization). . . . 279
Multi-Class Logistic Regression. . . . 304
Multilayer Perceptron. . . . 330
Naive Bayes. . . . 352
Online Multi-Class Logistic Regression . . . 364
Random Decision Trees. . . . 381
Support Vector Machine. . . . 399
Incremental Classification on SAP HANA Streaming Analytics. . . . 423
5.3 Regression Algorithms. . . . 423
Bi-Variate Geometric Regression. . . . 424
Bi-Variate Natural Logarithmic Regression. . . . 431
Cox Proportional Hazard Model. . . . 438
Exponential Regression. . . . 448
Generalised Linear Models. . . . 456
Multiple Linear Regression. . . . 478
Online Linear Regression (Stateless). . . . 507
Polynomial Regression. . . . 513
5.4 Association Algorithms. . . . 524
Apriori. . . . 524
FP-Growth. . . . 542
K-Optimal Rule Discovery (KORD). . . . 552
Sequential Pattern Mining. . . . 557
5.5 Time Series Algorithms. . . . 566
Additive Model Time Series Analysis. . . . 566
ARIMA. . . . 593
Auto ARIMA. . . . 627
Auto Exponential Smoothing. . . . 651
Bayesian Change Point Detection. . . . 671
Bayesian Structural Time Series. . . . 679
Brown Exponential Smoothing. . . . 689
Change-Point Detection. . . . 695
Croston's Method. . . . 702
Croston TSB Method. . . . 707
Discrete Wavelet Packet Transform. . . . 722
Discrete Wavelet Transform . . . 732

Dynamic Time Warping. . . . . 740
Fast Dynamic Time Warping. . . . . 750
Fast Fourier Transform. . . . . 758
Forecast Accuracy Measures. . . . . 762
GARCH. . . . . 766
Hierarchical Forecast. . . . . 776
Intermittent Time Series Forecast (ITSF). . . . . 783
Linear Regression with Damped Trend and Seasonal Adjust. . . . . 788
Long-Term Series Forecasting. . . . . 794
Online ARIMA. . . . . 806
Online Bayesian Change Point Detection. . . . . 815
Outlier Detection. . . . . 821
Periodogram. . . . . 888
ROCKET. . . . . 892
Single Exponential Smoothing. . . . . 912
Double Exponential Smoothing. . . . . 917
Triple Exponential Smoothing. . . . . 924
Seasonality Test. . . . . 934
Stationarity Test. . . . . 941
Trend Test. . . . . 946
VARMA. . . . . 950
White Noise Test. . . . . 972
5.6 Pipeline. . . . . 975
Pipeline Operator. . . . . 988
Customizable Output Interface. . . . . 998
5.7 AutoML. . . . . 1013
Configuration. . . . . 1022
Tutorial Dataset. . . . . 1026
Optimization. . . . . 1038
AutoML Tutorial. . . . . 1043
5.8 Preprocessing Algorithms. . . . . 1049
Categorical Principal Component Analysis. . . . . 1050
Discretize. . . . . 1061
Discretize Apply. . . . . 1067
Feature Selection. . . . . 1071
Isolation Forest. . . . . 1078
Inter-Quartile Range. . . . . 1084
MCMC Sampling. . . . . 1087
Missing Value Handling. . . . . 1095
Time Series Missing Value Handling. . . . . 1110
Multidimensional Scaling (MDS). . . . . 1120

Partition. . . . . 1124
Power Transform. . . . . 1129
Principal Component Analysis (PCA). . . . . 1139
Random Distribution Sampling. . . . . 1147
Sampling. . . . . 1158
Scale. . . . . 1167
Scale with Model. . . . . 1172
SMOTE. . . . . 1176
SMOTETomek. . . . . 1184
Tomek's Links. . . . . 1189
Variance Test. . . . . 1195
5.9 Statistics Algorithms. . . . . 1199
Anova. . . . . 1200
Chi-Square Goodness-of-Fit Test. . . . . 1211
Chi-Squared Test of Independence. . . . . 1213
Condition Index. . . . . 1216
Correlation Function. . . . . 1220
Cumulative Distribution Function. . . . . 1229
Distribution Fitting. . . . . 1233
Distribution Quantile. . . . . 1240
Entropy. . . . . 1243
Equal Variance Test. . . . . 1247
Factor Analysis. . . . . 1250
Grubbs' Test. . . . . 1258
Kaplan-Meier Survival Analysis. . . . . 1262
Kernel Density. . . . . 1267
Kolmogorov-Smirnov Test. . . . . 1278
Multivariate Analysis. . . . . 1283
One-Sample Median Test. . . . . 1286
T-Test. . . . . 1289
Univariate Analysis. . . . . 1294
Wilcox Signed Rank Test. . . . . 1300
5.10 Social Network Analysis Algorithms. . . . . 1303
Link Prediction. . . . . 1303
PageRank. . . . . 1307
5.11 Recommender System Algorithms. . . . . 1310
Alternating Least Squares. . . . . 1310
Factorized Polynomial Regression Models. . . . . 1328
Field-Aware Factorization Machine. . . . . 1354
5.12 Miscellaneous. . . . . 1372
ABC Analysis. . . . . 1372

Model Visualization. . . . . 1375
T-distributed Stochastic Neighbour Embedding. . . . . 1378
Weighted Score Table. . . . . 1386
5.13 Unified Interface. . . . . 1390
Unified Classification. . . . . 1390
Unified Classification with Pivoted Input Data. . . . . 1575
Unified Massive Classification. . . . . 1753
Unified Massive Classification with Pivoted Input Data. . . . . 1768
Unified Clustering. . . . . 1785
Unified Clustering with Pivoted Input Data. . . . . 1831
Unified Massive Clustering. . . . . 1874
Unified Massive Clustering with Pivoted Input Data. . . . . 1883
Unified Regression. . . . . 1895
Unified Regression with Pivoted Input Data. . . . . 2015
Unified Massive Regression. . . . . 2133
Unified Massive Regression with Pivoted Input Data. . . . . 2144
Unified Exponential Smoothing. . . . . 2157
Unified Exponential Smoothing with Pivoted Input Data. . . . . 2175
5.14 Text Processing Algorithms. . . . . 2195
Conditional Random Field. . . . . 2196
Latent Dirichlet Allocation. . . . . 2209
TF-IDF Calculation. . . . . 2221

**6 Model Evaluation and Parameter Selection. . . . . 2227**
6.1 Successive Halving and Hyperband for Model Evaluation and Parameter Selection. . . . . 2240

**7 Model Comparison. . . . . 2243**

**8 End-to-End Scenarios. . . . . 2261**
8.1 Scenario: Predict Segmentation of New Customers for a Supermarket. . . . . 2261
8.2 Scenario: Analyze the Cash Flow of an Investment on a New Product. . . . . 2264
8.3 Scenario: Survival Analysis. . . . . 2272

**9 Best Practices. . . . . 2282**

**10 Important Disclaimer for Features in SAP HANA. . . . . 2283**

# 1 SAP HANA Predictive Analysis Library (PAL)

This reference describes the Predictive Analysis Library (PAL) delivered with SAP HANA. This application function library (AFL) defines functions that can be called from within SAP HANA SQLScript procedures to perform analytic algorithms.

SAP HANA's SQLScript is an extension of SQL. It includes enhanced control-flow capabilities and lets developers define complex application logic inside database procedures. However, it is difficult to describe predictive analysis logic with procedures.

For example, an application may need to perform a cluster analysis in a huge customer table with 1T records. It is impossible to implement the analysis in a procedure using the simple classic K-means algorithms, or with more complicated algorithms in the data-mining area. Transferring large tables to the application server to perform the K-means calculation is also costly.

The Predictive Analysis Library (PAL) defines functions that can be called from within SQLScript procedures to perform analytic algorithms. This release of PAL includes classic and universal predictive analysis algorithms in ten data-mining categories:

- Clustering
- Classification
- Regression
- Association
- Time Series
- Preprocessing
- Statistics
- Social Network Analysis
- Recommender System
- Miscellaneous

The algorithms in PAL were carefully selected based on the following criteria:

- The algorithms are needed for SAP HANA applications.
- The algorithms are the most commonly used based on market surveys.
- The algorithms are generally available in other database products.

## PAL on SAP HANA Streaming Analytics

PAL also provides several incremental machine learning algorithms that learn and update a model on the fly, so that predictions are based on a dynamic model.

For detailed information, see the section on machine learning with streaming in the *SAP HANA Streaming Analytics: Developer Guide*.

# 2 Getting Started with PAL

This section covers the information you need to know to start working with the SAP HANA Predictive Analysis Library.

## 2.1 Prerequisites

To use the PAL procedures, you must:

- Install SAP HANA Platform 2.0 SPS 05.
- Install the Application Function Library (AFL), which includes the PAL.
  For information on how to install or update AFL, see the section on installing or updating SAP HANA components in *SAP HANA Server Installation and Update Guide*:

  > **Note**
  >
  > The revision number of the AFL must match the revision number of SAP HANA. See SAP Note 1898497 for details.

- Enable the Script Server in HANA instance. See SAP Note 1650957 for further information.

### Related Information

SAP HANA Server Installation and Update Guide
SAP Note 1898497
SAP Note 1650957

## 2.2 Application Function Library (AFL)

You can dramatically increase performance by executing complex computations in the database instead of at the application sever level. SAP HANA provides several techniques to move application logic into the database, and one of the most important is the use of application functions. Application functions are like database procedures written in C++ and called from outside to perform data intensive and complex operations.

Functions for a particular topic are grouped into an application function library (AFL), such as the Predictive Analysis Library (PAL) and the Business Function Library (BFL). Currently, PAL and BFL are delivered in one archive (that is, one SAR file with the name AFL`<version_string>`.SAR). The AFL archive is not part of the HANA appliance, and must be installed separately by the administrator.

## 2.3 Security

This section provides detailed security information which can help administrator and architects answer some common questions.

### Role Assignment

You must be assigned one of the following roles to execute the PAL procedures. The roles for the PAL library are automatically created when the Application Function Library (AFL) is installed. The role names are:

- `AFL__SYS_AFL_AFLPAL_EXECUTE`
- `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION`

> **Note**
>
> There are 2 underscores between `AFL` and `SYS`.

Prior to SAP HANA Platform 2.0 SPS 02, you need the `AFLPM_CREATOR_ERASER_EXECUTE` role to create or drop wrapper PAL procedures using `"SYS".AFLLANG_WRAPPER_PROCEDURE_CREATE` or `"SYS".AFLLANG_WRAPPER_PROCEDURE_DROP`.

> **Note**
>
> Once the above roles are automatically created, they cannot be dropped. In other words, even when an area with all its objects is dropped and re-created during system startup, the user still keeps these roles originally granted.

## 2.4 Checking PAL Installation

To confirm that the PAL procedures were installed successfully, you can check the following three public views:

- `sys.afl_areas`
- `sys.afl_packages`
- `sys.afl_functions`

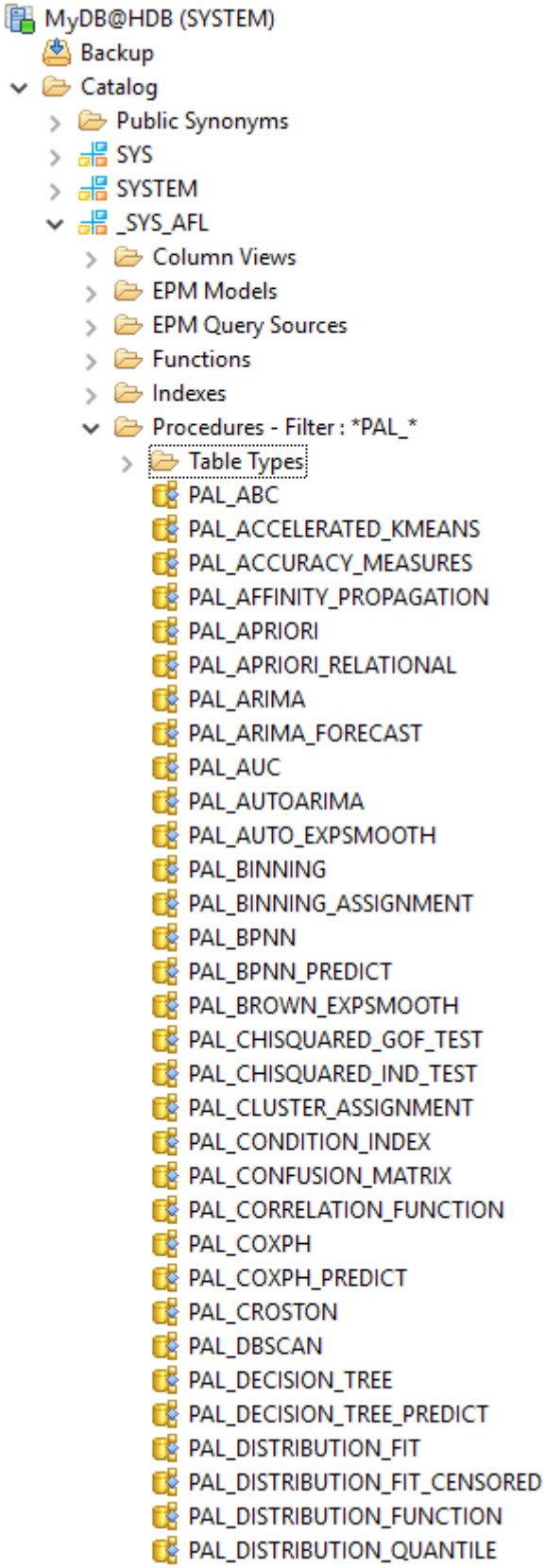
These views are granted to the `PUBLIC` role and can be accessed by anyone.

To check the views, run the following SQL statements:

```
SELECT * FROM "SYS"."AFL_AREAS" WHERE AREA_NAME = 'AFLPAL';
SELECT * FROM "SYS"."AFL_PACKAGES" WHERE AREA_NAME = 'AFLPAL';
SELECT * FROM "SYS"."AFL_FUNCTIONS" WHERE AREA_NAME = 'AFLPAL';
```

The result will tell you whether the PAL procedures were successfully installed on your system.

You can also open the SAP HANA Catalog and check if the PAL procedures are generated under the _SYS_AFL schema, as shown below:



## 2.5 Restricting CPU and Memory Usage

For best performance, some PAL functions (for example., random decision tree, model evaluation and hyper-parameter search) might make use of many concurrent threads and memory for quite some time. While you can use the THREAD_RATIO parameter in PAL function to limit the maximum concurrent threads, it is strongly recommended to set limits of resources for scriptserver, where PAL functions are running.

You can also set up Workload Management (WLM) classes and WLM admission control for PAL users. Workload management in SAP HANA allows you to balance and manage all workload types, including PAL function for optimal throughput and response time. The workload class can be found in the WORKLOAD_CLASS_NAME field of system view M_ACTIVE_STATEMENTS.

**Examples**

**Example 1: Managing workload with workload class**

```
SET SCHEMA DM_PAL;
DROP WORKLOAD CLASS "MY_WORKLOAD_CLASS_1";
CREATE WORKLOAD CLASS "MY_WORKLOAD_CLASS_1"
   SET 'PRIORITY' = '3', 'STATEMENT MEMORY LIMIT' = '100', 'STATEMENT TIMEOUT' =
'2';
CALL _SYS_AFL.PAL_RANDOM_DECISION_TREES
(PAL_TRAINING_DATA_TBL, PAL_RDT_PARAMETER_TBL, ?, ?, ?, ?) WITH
HINT( WORKLOAD_CLASS("MY_WORKLOAD_CLASS_1"));
```

**Example 2: Managing workload with workload mapping**

```
SET SCHEMA DM_PAL;
DROP WORKLOAD CLASS "MY_WORKLOAD_CLASS_2";
CREATE WORKLOAD CLASS "MY_WORKLOAD_CLASS_2"
   SET 'PRIORITY' = '3', 'STATEMENT MEMORY LIMIT' = '100', 'STATEMENT TIMEOUT' =
'2';
CREATE WORKLOAD MAPPING "MAPPING_PAL_RDT" WORKLOAD CLASS "MY_WORKLOAD_CLASS_2"
SET 'SCHEMA NAME' = '_SYS_AFL', 'OBJECT NAME' = 'PAL_RANDOM_DECISION_TREES';
CALL _SYS_AFL.PAL_RANDOM_DECISION_TREES (PAL_TRAINING_DATA_TBL,
PAL_RDT_PARAMETER_TBL, ?, ?, ?, ?);
```

# 3 Calling PAL Procedures

After SAP HANA AFL is installed, PAL procedures are generated under schema _SYS_AFL. In general, you pass input tables (such as data tables and parameter tables) to those procedures and the returned results are stored in output tables.

## PAL Procedure Name

PAL procedures are under the `_SYS_AFL` schema. All PAL procedure names start with `PAL_`.

## Input Tables

Unlike other SAP HANA stored procedures which are static-typed and require pre-defined input and output table signatures, PAL procedures allow you to pass input tables with different column structures, as long as the table types are accepted as specified in this document. For example, you can pass a 10-column training data table or a 100-column training data table to the `_SYS_AFL.PAL_LOGISTIC_REGRESSION` procedure to generate the wrapper procedures.

**Example**

Pass a 3-feature (4-column) table or 4-feature (5-column) table to the PAL k-means procedure.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_4_COLUMN_DATA_TBL;
CREATE COLUMN TABLE PAL_4_COLUMN_DATA_TBL(
    "ID" INTEGER,
    "V000" DOUBLE,
    "V001" VARCHAR(2),
    "V002" DOUBLE
);
INSERT INTO PAL_4_COLUMN_DATA_TBL VALUES (0, 0.5, 'A', 0.5);
INSERT INTO PAL_4_COLUMN_DATA_TBL VALUES (1, 1.5, 'A', 0.5);
INSERT INTO PAL_4_COLUMN_DATA_TBL VALUES (2, 1.5, 'A', 1.5);
INSERT INTO PAL_4_COLUMN_DATA_TBL VALUES (3, 0.5, 'A', 1.5);
INSERT INTO PAL_4_COLUMN_DATA_TBL VALUES (4, 1.1, 'B', 1.2);
INSERT INTO PAL_4_COLUMN_DATA_TBL VALUES (5, 0.5, 'B', 15.5);
INSERT INTO PAL_4_COLUMN_DATA_TBL VALUES (6, 1.5, 'B', 15.5);
INSERT INTO PAL_4_COLUMN_DATA_TBL VALUES (7, 1.5, 'B', 16.5);
INSERT INTO PAL_4_COLUMN_DATA_TBL VALUES (8, 0.5, 'B', 16.5);
INSERT INTO PAL_4_COLUMN_DATA_TBL VALUES (9, 1.2, 'C', 16.1);
INSERT INTO PAL_4_COLUMN_DATA_TBL VALUES (10, 15.5, 'C', 15.5);
INSERT INTO PAL_4_COLUMN_DATA_TBL VALUES (11, 16.5, 'C', 15.5);
INSERT INTO PAL_4_COLUMN_DATA_TBL VALUES (12, 16.5, 'C', 16.5);
INSERT INTO PAL_4_COLUMN_DATA_TBL VALUES (13, 15.5, 'C', 16.5);
INSERT INTO PAL_4_COLUMN_DATA_TBL VALUES (14, 15.6, 'D', 16.2);
INSERT INTO PAL_4_COLUMN_DATA_TBL VALUES (15, 15.5, 'D', 0.5);
INSERT INTO PAL_4_COLUMN_DATA_TBL VALUES (16, 16.5, 'D', 0.5);
INSERT INTO PAL_4_COLUMN_DATA_TBL VALUES (17, 16.5, 'D', 1.5);
INSERT INTO PAL_4_COLUMN_DATA_TBL VALUES (18, 15.5, 'D', 1.5);
INSERT INTO PAL_4_COLUMN_DATA_TBL VALUES (19, 15.7, 'A', 1.6);
```

```
DROP TABLE PAL_5_COLUMN_DATA_TBL;
CREATE COLUMN TABLE PAL_5_COLUMN_DATA_TBL(
    "ID" INTEGER,
    "V000" DOUBLE,
    "V001" VARCHAR(2),
    "V002" DOUBLE,
    "V003" INTEGER
);
INSERT INTO PAL_5_COLUMN_DATA_TBL VALUES (0, 0.5, 'A', 0.5, 3);
INSERT INTO PAL_5_COLUMN_DATA_TBL VALUES (1, 1.5, 'A', 0.5, 4);
INSERT INTO PAL_5_COLUMN_DATA_TBL VALUES (2, 1.5, 'A', 1.5, -1);
INSERT INTO PAL_5_COLUMN_DATA_TBL VALUES (3, 0.5, 'A', 1.5, 0);
INSERT INTO PAL_5_COLUMN_DATA_TBL VALUES (4, 1.1, 'B', 1.2, 8);
INSERT INTO PAL_5_COLUMN_DATA_TBL VALUES (5, 0.5, 'B', 15.5, 9);
INSERT INTO PAL_5_COLUMN_DATA_TBL VALUES (6, 1.5, 'B', 15.5, -17);
INSERT INTO PAL_5_COLUMN_DATA_TBL VALUES (7, 1.5, 'B', 16.5, 2);
INSERT INTO PAL_5_COLUMN_DATA_TBL VALUES (8, 0.5, 'B', 16.5, 6);
INSERT INTO PAL_5_COLUMN_DATA_TBL VALUES (9, 1.2, 'C', 16.1, 12);
INSERT INTO PAL_5_COLUMN_DATA_TBL VALUES (10, 15.5, 'C', 15.5, 19);
INSERT INTO PAL_5_COLUMN_DATA_TBL VALUES (11, 16.5, 'C', 15.5, -21);
INSERT INTO PAL_5_COLUMN_DATA_TBL VALUES (12, 16.5, 'C', 16.5, 11);
INSERT INTO PAL_5_COLUMN_DATA_TBL VALUES (13, 15.5, 'C', 16.5, 15);
INSERT INTO PAL_5_COLUMN_DATA_TBL VALUES (14, 15.6, 'D', 16.2, 3);
INSERT INTO PAL_5_COLUMN_DATA_TBL VALUES (15, 15.5, 'D', 0.5, 1);
INSERT INTO PAL_5_COLUMN_DATA_TBL VALUES (16, 16.5, 'D', 0.5, -15);
INSERT INTO PAL_5_COLUMN_DATA_TBL VALUES (17, 16.5, 'D', 1.5, -5);
INSERT INTO PAL_5_COLUMN_DATA_TBL VALUES (18, 15.5, 'D', 1.5, 5);
INSERT INTO PAL_5_COLUMN_DATA_TBL VALUES (19, 15.7, 'A', 1.6, 19);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL(
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR(1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 0.2, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('GROUP_NUMBER', 4, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('INIT_TYPE', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('DISTANCE_LEVEL',2, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MAX_ITERATION', 100, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('EXIT_THRESHOLD', NULL, 1.0E-6, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('CATEGORY_WEIGHTS', NULL, 0.5, NULL);
CALL _SYS_AFL.PAL_KMEANS(PAL_4_COLUMN_DATA_TBL,
"#PAL_PARAMETER_TBL", ?, ?, ?, ?, ?);
CALL _SYS_AFL.PAL_KMEANS(PAL_5_COLUMN_DATA_TBL,
"#PAL_PARAMETER_TBL", ?, ?, ?, ?, ?);
```

### Support for Categorical Attributes with One-hot Encoding

Some PAL functions allow for data with categorical types in different ways. One of them is one-hot encoding. That is, if an attribute is of category type, its value is converted to a binary vector and then used as a numerical attribute. For example, Gender in the following table is of category type.

| Customer ID | Age | Income | Gender |
|---|---|---|---|
| T1 | 31 | 10,000 | Female |
| T2 | 27 | 8,000 | Male |

Because Gender has two distinct values, it is converted into a binary vector with two dimensions:

| Customer ID | Age | Income | Gender 1 | Gender 2 |
|---|---|---|---|---|
| T1 | 31 | 10,000 | 0 | 1 |
| T2 | 27 | 8,000 | 1 | 0 |

One-hot encoding is used in the following functions.

- Accelerated K-Means
- DBSCAN
- Gaussian Mixture Model (GMM)
- Generalised Linear Models
- K-Means
- K-Medians
- K-Medoids
- KNN
- Logistic Regression (with Elastic Net Regularization)
- Multi-Class Logistic Regression
- Multilayer Perceptron
- Support Vector Machine
- Multiple Linear Regression

## Parameter Tables

PAL procedures use parameter tables to transfer parameter values for each function. Each PAL procedure has its own parameter table. To avoid a conflict of table names when several users call PAL procedures at the same time, it is recommended that the parameter table be created as a local temporary column table, so that each parameter table has its own unique scope per session.

The parameter table structure is as follows:

| Column Name | Data Type | Description |
|---|---|---|
| PARAM_NAME | VARCHAR | Parameter name |
| INT_VALUE | INTEGER | INTEGER parameter value |
| DOUBLE_VALUE | DOUBLE | DOUBLE parameter value |
| STRING_VALUE | VARCHAR | STRING parameter value |

In most cases, each row contains one parameter value, INTEGER, DOUBLE, or STRING. In some cases, two parameter values might be required. Refer to the parameter table description for each PAL procedure.

The following table is an example of a parameter table with three parameters. The first parameter, `GROUP_NUMBER`, is an integer parameter. Thus, in the `GROUP_NUMBER` row, you should fill the parameter value in the `INT_VALUE` column, and fill NULL in the `DOUBLE_VALUE` and `STRING_VALUE` columns.

| PARAM_NAME | INT_VALUE | DOUBLE_VALUE | STRING_VALUE |
|---|---|---|---|
| GROUP_NUMBER | 1 | NULL | NULL |
| SUPPORT | NULL | 0.2 | NULL |
| VAR_NAME | NULL | NULL | hello |

## General Parameters

There are some general parameters which are used by some PAL procedures. The following table introduces these parameters.

| Parameter Name | Description |
|---|---|
| THREAD_RATIO | The `THREAD_RATIO` (DOUBLE) parameter allows you to set the upper limit of thread usage in proportion of the current available threads.<br><br>The valid value range is `[0,1]`, where `0` indicates that a single thread shall be used and `1` means all the current available threads will be used if necessary. Invalid value input will be ignored. |
| CATEGORICAL_VARIABLE | By default, integer column is considered as continuous column for all PAL procedures.<br><br>The `CATEGORICAL_VARIABLE` (VARCHAR) parameter allows you to explicitly set integer column as categorical column by column name in string.<br><br>If the an integer column is set as `CATEGORICAL_VARIABLE`, this parameter is used to set classification or regression problem.<br><br>If a double column is set as `CATEGORICAL_VARIABLE`, it will be ignored. |
| DEPENDENT_VARIABLE | For classification/regression algorithms, the `DEPENDENT_VARIABLE` (VARCHAR) parameter allows you to explicitly set the column name in a string of dependent variables (i.e. “Y”, “response”, ...).<br><br>An error will be thrown if an ID column, if exists, is specified as `DEPENDENT_VARIABLE`. |

| Parameter Name | Description |
| --- | --- |
| HAS_ID | Some PAL algorithms require an ID column and some do not. This makes it difficult to switch between PAL algorithms with the same data table. The `HAS_ID` (INTEGER) parameter allows you to explicitly tell the PAL algorithm whether the first column is an ID column.<br>• For algorithms which require an ID column:<br>  • `HAS_ID = 0` indicates that there is no ID column in the data table. The algorithm will use all the columns to perform training or processing and return the results. However, any output table related to an ID column (e.g. fitted value for linear regression) will have no output information.<br>  • `HAS_ID =1` (default) indicates that the **first** column of the data table is an ID column. The algorithm will ignore that column when performing training or processing and returning the results. All the ID related information will also return.<br>• For algorithms which do not require an ID column:<br>  • `HAS_ID = 0` (default) indicates there is no ID column in the data table. The algorithm will use all the columns to perform training or processing and return the results.<br>  • `HAS_ID = 1` indicates that the **first** column of the data table is an ID column. The algorithm will ignore that column when performing training or processing and returning the results.<br>**Notes:**<br>• The ID column, if exist, is always the first column in the input table before any processing.<br>• A scoring procedure always requires an ID column and `HAS_ID` does not apply to those procedures.<br>• `HAS_ID` does not apply to clustering algorithms. |

## Output Tables

PAL procedures can determine the output table structure (i.e. number of columns, column name and type) without explicit definition. The output table structure is determined based on different rules as follows:

1. The number of columns, column name and type of the output table are fixed and pre-defined by PAL.
2. The column name and type in the output table inherit from the column in the input table. For example, the output column which includes the row ID information will have the same column name and type as the corresponding input column.

3. The output table structure inherits from the input table structure. For example, the `_SYS_AFL.PAL_KMEANS` procedure returns the center coordinates. The number of columns and column name inherit from the input data table which includes the data features.
4. The column type is "widened" to DOUBLE type from the input column with INTEGER type. For example, the `_SYS_AFL.PAL_KMEANS` procedure returns center coordinates, each of which has the same column name as in the input data column. If the input data column is of INTEGER type, the corresponding output column will be of DOUBLE type because it includes the mean value of the feature, which is usually not of INTEGER type.
5. The column name inherits from input column with additional prefix string. For example, the `_SYS_AFL.PAL_PCA` procedure returns loadings information and the column name inherits from the input column with the additional prefix "`LOADINGS`" because the column represents loading for the corresponding input column on the calculated component.
6. The column name is pre-defined by PAL with increment numbers. For example, the `_SYS_AFL.PAL_PCA` procedure returns scores on each component and the column is named as `COMPONENT_1`, `COMPONENT_2`, etc.

The output table structure for each PAL procedure is described in the *PAL Procedures* section of this document.

PAL procedure only requires matchable output column types. You can pass tables with the same column type but different column name as the output table for PAL procedures.

> **Note**
>
> The interfaces to call PAL procedures described in this document are introduced since SAP HANA 2.0 SPS 02. The previous wrapper procedure generation approach, using the table signature between procedures generated by `SYS.AFLLANG_WRAPPER_PROCEDURE_CREATE`, is still supported. Refer to the older versions (e.g. SAP HANA 2.0 SPS 01, SAP HANA 1.0 SPS 12) of this document for information on wrapper procedure generation.

## Calling PAL from machine learning clients for SAP HANA

You can use PAL procedures through the Python and R machine learning clients for SAP HANA. See the following guides for details:

- Python Machine Learning Client for SAP HANA
- R Machine Learning Client for SAP HANA

> **Note**
>
> Column names with single quotation marks ('') are not supported by the `AFLLANG` procedure. Therefore you cannot use such a column name in PAL.

## Related Information

SAP Note 2046767

## 3.1 Switching Between the PAL Algorithms with the Same Datatset

Below is an example to illustrate that given the same training data, how to switch between the linear regression and random decision trees for training.

```
SET SCHEMA DM_PAL;
-- Prepare training data table
DROP TABLE PAL_DATA_TBL;
CREATE COLUMN TABLE PAL_DATA_TBL
(
    "ID" VARCHAR(50),
    "Y" DOUBLE,
    "X1" DOUBLE,
    "X2" VARCHAR(100),
    "X3" INTEGER
);
INSERT INTO PAL_DATA_TBL VALUES (0, -6.879, 0.00, 'A', 1);
INSERT INTO PAL_DATA_TBL VALUES (1, -3.449, 0.50, 'A', 1);
INSERT INTO PAL_DATA_TBL VALUES (2,  6.635, 0.54, 'B', 1);
INSERT INTO PAL_DATA_TBL VALUES (3, 11.844, 1.04, 'B', 1);
INSERT INTO PAL_DATA_TBL VALUES (4,  2.786, 1.50, 'A', 1);
INSERT INTO PAL_DATA_TBL VALUES (5,  2.389, 0.04, 'B', 2);
INSERT INTO PAL_DATA_TBL VALUES (6, -0.011, 2.00, 'A', 2);
INSERT INTO PAL_DATA_TBL VALUES (7,  8.839, 2.04, 'B', 2);
INSERT INTO PAL_DATA_TBL VALUES (8,  4.689, 1.54, 'B', 1);
INSERT INTO PAL_DATA_TBL VALUES (9, -5.507, 1.00, 'A', 2);
-- Prepare parameter table for linear regression
DROP TABLE #PAL_MLR_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_MLR_PARAMETER_TBL ("NAME" VARCHAR(100),
"INTARGS" INT, "DOUBLEARGS" DOUBLE,"STRINGARGS" VARCHAR(100));
INSERT INTO #PAL_MLR_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 0.5, NULL);
INSERT INTO #PAL_MLR_PARAMETER_TBL VALUES ('PMML_EXPORT', 1, NULL, NULL);
-- INTEGER column is by default continuous column, explicitly set to categorical
variable
INSERT INTO #PAL_MLR_PARAMETER_TBL VALUES ('CATEGORICAL_VARIABLE', NULL,
NULL,'X3');
-- Call linear regression
CALL _SYS_AFL.PAL_LINEAR_REGRESSION(PAL_DATA_TBL,
"#PAL_MLR_PARAMETER_TBL", ?, ?, ?, ?, ?);
-- Prepare parameter table for random decision trees
DROP TABLE #PAL_RDT_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_RDT_PARAMETER_TBL ("NAME" VARCHAR(100),
"INTARGS" INT, "DOUBLEARGS" DOUBLE,"STRINGARGS" VARCHAR(100));
INSERT INTO #PAL_RDT_PARAMETER_TBL VALUES ('TREES_NUM', 10, NULL, NULL);
INSERT INTO #PAL_RDT_PARAMETER_TBL VALUES ('SPLIT_THRESHOLD', NULL, 1e-5, NULL);
INSERT INTO #PAL_RDT_PARAMETER_TBL VALUES ('NODE_SIZE', 1, NULL, NULL);
INSERT INTO #PAL_RDT_PARAMETER_TBL VALUES
('CATEGORICAL_VARIABLE',NULL,NULL,'X3');
-- The training data consist of ID column, explicitly set HAS_ID = 1 and
algorithm will not consider the first column as feature
INSERT INTO #PAL_RDT_PARAMETER_TBL VALUES ('HAS_ID', 1, NULL, NULL);
-- By default the last column of the input table is the DEPENDENT_VARIABLE,
explicitly set DEPENDENT_VARIABLE as column 'Y'
INSERT INTO #PAL_RDT_PARAMETER_TBL VALUES ('DEPENDENT_VARIABLE', NULL, NULL,'Y');
-- Call random decision trees with same training data
CALL _SYS_AFL.PAL_RANDOM_DECISION_TREES(PAL_DATA_TBL,
"#PAL_RDT_PARAMETER_TBL", ?, ?, ?, ?);
```

## 3.2 Calling PAL Procedures in Parallel with Hint PARALLEL_BY_PARAMETER_PARTITIONS

You can run parallel execution of selected PAL procedures with partition table as input from SAP HANA SQLScript using the `WITH HINT (PARALLEL_BY_PARAMETER_PARTITIONS ())` clause.

The main scenario is to run scoring procedure with a trained model from PAL supervised learning algorithms, such as decision trees and random decision trees. Given a partitioned data table, the parallel execution of the scoring procedure will be initiated on each data partition, sharing the same trained model and other procedure parameters from the other unpartitioned tables. This feature works on both single-node and multiple-node SAP HANA environment.

Among input tables, data partitioning is only valid for data table. As data table is the first input table for the PAL procedures that are enabled with this hint, you should always use `PARALLEL_BY_PARAMETER_PARTITIONS (p1)` when calling the procedures, where `p1` is the first input parameter of the PAL procedure.

The data table can be partitioned and distributed in a cluster of SAP HANA nodes. If SAP HANA Script Server is started in a specific node, the execution of PAL procedure for the data partition stored in that node will be done locally by this co-located Script Server. If no Script Server is started in a node, the location where the data partition stored in that node will be processed is decided either by an optimizer or by an additional hint.

The hint can be used to perform scoring for the following PAL procedures:

- Multilayer Perceptron
- Bi-variate geometric regression
- Bi-variate natural logarithmic regression
- Binning assignment
- Cluster assignment
- Exponential regression
- Generalized linear models
- Linear discriminant analysis
- Logistic regression (with elastic net regularization)
- Multi-class logistic regression
- Multiple linear regression
- Naive Bayes
- Polynomial regression
- Decision trees
- Principal component analysis
- Random decision trees
- Support vector machine

**Example**

```
SET SCHEMA DM_PAL;
DROP TABLE  PAL_C45_TRAINING_DATA_TBL;
CREATE COLUMN TABLE PAL_C45_TRAINING_DATA_TBL (
    "OUTLOOK" VARCHAR(20),
    "TEMP" INTEGER,
    "HUMIDITY" DOUBLE,
    "WINDY" VARCHAR(10),
```

```
    "CLASS" VARCHAR(20)
);
INSERT INTO PAL_C45_TRAINING_DATA_TBL VALUES ('Sunny',    75, 70.0, 'Yes',
'Play');
INSERT INTO PAL_C45_TRAINING_DATA_TBL VALUES ('Sunny',    80, 90.0, 'Yes', 'Do
not Play');
INSERT INTO PAL_C45_TRAINING_DATA_TBL VALUES ('Sunny',    85, 85.0, 'No',  'Do
not Play');
INSERT INTO PAL_C45_TRAINING_DATA_TBL VALUES ('Sunny',    72, 95.0, 'No',  'Do
not Play');
INSERT INTO PAL_C45_TRAINING_DATA_TBL VALUES ('Sunny',    69, 70.0, 'No',
'Play');
INSERT INTO PAL_C45_TRAINING_DATA_TBL VALUES ('Overcast', 72, 90.0, 'Yes',
'Play');
INSERT INTO PAL_C45_TRAINING_DATA_TBL VALUES ('Overcast', 83, 78.0, 'No',
'Play');
INSERT INTO PAL_C45_TRAINING_DATA_TBL VALUES ('Overcast', 64, 65.0, 'Yes',
'Play');
INSERT INTO PAL_C45_TRAINING_DATA_TBL VALUES ('Overcast', 81, 75.0, 'No',
'Play');
INSERT INTO PAL_C45_TRAINING_DATA_TBL VALUES ('Rain',     71, 80.0, 'Yes', 'Do
not Play');
INSERT INTO PAL_C45_TRAINING_DATA_TBL VALUES ('Rain',     65, 70.0, 'Yes', 'Do
not Play');
INSERT INTO PAL_C45_TRAINING_DATA_TBL VALUES ('Rain',     75, 80.0, 'No',
'Play');
INSERT INTO PAL_C45_TRAINING_DATA_TBL VALUES ('Rain',     68, 80.0, 'No',
'Play');
INSERT INTO PAL_C45_TRAINING_DATA_TBL VALUES ('Rain',     70, 96.0, 'No',
'Play');
DROP TABLE #PAL_TRAINING_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_TRAINING_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO #PAL_TRAINING_PARAMETER_TBL VALUES ('ALGORITHM',                1,
null, null);
INSERT INTO #PAL_TRAINING_PARAMETER_TBL VALUES ('SPLIT_THRESHOLD',       null,
1e-5, null);
INSERT INTO #PAL_TRAINING_PARAMETER_TBL VALUES ('MAX_DEPTH',                4,
null, null);
INSERT INTO #PAL_TRAINING_PARAMETER_TBL VALUES ('MIN_RECORDS_OF_PARENT',    2,
null, null);
INSERT INTO #PAL_TRAINING_PARAMETER_TBL VALUES ('MIN_RECORDS_OF_LEAF',      1,
null, null);
INSERT INTO #PAL_TRAINING_PARAMETER_TBL VALUES ('IS_OUTPUT_RULES',          1,
null, null);
DROP TABLE PAL_C45_TREEMODEL_TBL;
CREATE COLUMN TABLE PAL_C45_TREEMODEL_TBL (
    "ID" INTEGER,
    "TREEMODEL" VARCHAR(5000)
);
CALL "_SYS_AFL"."PAL_DECISION_TREE"(PAL_C45_TRAINING_DATA_TBL,
#PAL_TRAINING_PARAMETER_TBL, PAL_C45_TREEMODEL_TBL, ?, ?, ?, ?) WITH OVERVIEW;
DROP TABLE PAL_DT_SCORING_DATA_TBL;
CREATE COLUMN TABLE PAL_DT_SCORING_DATA_TBL (
    "ID" INTEGER,
    "OUTLOOK" VARCHAR(20),
    "TEMP" INTEGER,
    "HUMIDITY" DOUBLE,
    "WINDY" VARCHAR(10)
    ) GROUP TYPE "MULTI_NODE"
      GROUP NAME "NODE_ALL"
      PARTITION BY RANGE(ID) (
          PARTITION 0<=VALUES<2,
          PARTITION 2<=VALUES<4,
```

```
          PARTITION 4<=VALUES<6,
          PARTITION OTHERS);
INSERT INTO PAL_DT_SCORING_DATA_TBL VALUES (0, 'Overcast', 75, 70, 'Yes');
INSERT INTO PAL_DT_SCORING_DATA_TBL VALUES (1, 'Rain',     78, 70, 'Yes');
INSERT INTO PAL_DT_SCORING_DATA_TBL VALUES (2, 'Sunny',    66, 70, 'Yes');
INSERT INTO PAL_DT_SCORING_DATA_TBL VALUES (3, 'Sunny',    69, 70, 'Yes');
INSERT INTO PAL_DT_SCORING_DATA_TBL VALUES (4, 'Rain',   null, 70, 'Yes');
INSERT INTO PAL_DT_SCORING_DATA_TBL VALUES (5, null,       70, 70, 'Yes');
INSERT INTO PAL_DT_SCORING_DATA_TBL VALUES (6, '***',      70, 70, 'Yes');
DROP TABLE #PAL_SCORING_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_SCORING_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO #PAL_SCORING_PARAMETER_TBL VALUES ('IS_OUTPUT_PROBABILITY', 1, null,
null);
INSERT INTO #PAL_SCORING_PARAMETER_TBL VALUES ('MODEL_FORMAT', 0, null, null);
CALL "_SYS_AFL"."PAL_DECISION_TREE_PREDICT"(PAL_DT_SCORING_DATA_TBL,
PAL_C45_TREEMODEL_TBL, #PAL_SCORING_PARAMETER_TBL, ?)
    WITH HINT (PARALLEL_BY_PARAMETER_PARTITIONS(p1));
```

**Expected Result**

| | ID | SCORE | CONFIDENCE |
|---|---|---|---|
| 1 | 0 | Play | 1 |
| 2 | 1 | Do not Play | 1 |
| 3 | 2 | Play | 1 |
| 4 | 3 | Play | 1 |
| 5 | 4 | Play | 0.642857142... |
| 6 | 5 | Play | 0.692307692... |
| 7 | 6 | Play | 0.692307692... |

## 3.3 Calling PAL Procedures in Parallel with Hint PARALLEL_BY_PARAMETER_VALUES

The PARALLEL_BY_PARAMETER_VALUES hint enables parallel execution of PAL functions. Processing of data is performed in parallel by groups.

The following examples use PAL functions which have a data table and a parameter table as input and assume the data table has big amount of data and the parameter table has small amount of data. Here are some typical use cases:

**Example 1: Group specific parameters**

Both the data table and the parameter table have the group identifier column. Rows in the data table and the parameter table with the same group identifier are mapped and processed together. A set of result data will be generated for each group. The result from different groups of input data will be collected together to build the final result.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_SES_DATA_GRP_TBL;
```

```
CREATE COLUMN TABLE PAL_SES_DATA_GRP_TBL (
    "GROUP_ID" NVARCHAR(10),
    "ID" INT,
    "RAWDATA" DOUBLE
);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_1',  1, 200.0);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_1',  2, 135.0);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_1',  3, 195.0);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_1',  4, 197.5);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_1',  5, 310.0);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_1',  6, 175.0);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_1',  7, 155.0);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_1',  8, 130.0);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_1',  9, 220.0);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_1', 10, 277.5);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_1', 11, 235.0);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_2',  1, 200.0);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_2',  2, 135.0);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_2',  3, 195.0);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_2',  4, 197.5);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_2',  5, 310.0);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_2',  6, 175.0);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_2',  7, 155.0);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_2',  8, 130.0);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_2',  9, 220.0);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_2', 10, 277.5);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_2', 11, 235.0);
DROP TABLE PAL_SES_PARAMETER_GRP_TBL;
CREATE COLUMN TABLE PAL_SES_PARAMETER_GRP_TBL (
    "GROUP_ID" NVARCHAR(10),
    "PARAM_NAME" NVARCHAR(100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR(100)
);
INSERT INTO PAL_SES_PARAMETER_GRP_TBL VALUES ('GROUP_1', 'ADAPTIVE_METHOD',0,
NULL, NULL);
INSERT INTO PAL_SES_PARAMETER_GRP_TBL VALUES ('GROUP_1', 'MEASURE_NAME', NULL,
NULL, 'MSE');
INSERT INTO PAL_SES_PARAMETER_GRP_TBL VALUES ('GROUP_1', 'ALPHA', NULL,0.1,
NULL);
INSERT INTO PAL_SES_PARAMETER_GRP_TBL VALUES ('GROUP_1', 'DELTA', NULL,0.2,
NULL);
INSERT INTO PAL_SES_PARAMETER_GRP_TBL VALUES ('GROUP_1', 'FORECAST_NUM',12,
NULL,NULL);
INSERT INTO PAL_SES_PARAMETER_GRP_TBL VALUES ('GROUP_1', 'EXPOST_FLAG',1,
NULL,NULL);
INSERT INTO PAL_SES_PARAMETER_GRP_TBL VALUES ('GROUP_1',
'PREDICTION_CONFIDENCE_1', NULL, 0.8, NULL);
INSERT INTO PAL_SES_PARAMETER_GRP_TBL VALUES ('GROUP_1',
'PREDICTION_CONFIDENCE_2', NULL, 0.95, NULL);
INSERT INTO PAL_SES_PARAMETER_GRP_TBL VALUES ('GROUP_2', 'ADAPTIVE_METHOD',0,
NULL, NULL);
INSERT INTO PAL_SES_PARAMETER_GRP_TBL VALUES ('GROUP_2', 'MEASURE_NAME', NULL,
NULL, 'MSE');
INSERT INTO PAL_SES_PARAMETER_GRP_TBL VALUES ('GROUP_2', 'ALPHA', NULL,0.1,
NULL);
INSERT INTO PAL_SES_PARAMETER_GRP_TBL VALUES ('GROUP_2', 'DELTA', NULL,0.2,
NULL);
INSERT INTO PAL_SES_PARAMETER_GRP_TBL VALUES ('GROUP_2', 'FORECAST_NUM',12,
NULL,NULL);
INSERT INTO PAL_SES_PARAMETER_GRP_TBL VALUES ('GROUP_2', 'EXPOST_FLAG',1,
NULL,NULL);
INSERT INTO PAL_SES_PARAMETER_GRP_TBL VALUES ('GROUP_2',
'PREDICTION_CONFIDENCE_1', NULL, 0.75, NULL);
INSERT INTO PAL_SES_PARAMETER_GRP_TBL VALUES ('GROUP_2',
'PREDICTION_CONFIDENCE_2', NULL, 0.8, NULL);
DROP TABLE PAL_SES_RESULT_GRP_TBL;
```

```
CREATE COLUMN TABLE PAL_SES_RESULT_GRP_TBL (
    "GROUP_ID" NVARCHAR(10),
    "TIMESTAMP" INT,
    "VALUE" DOUBLE,
    "PI1_LOWER" DOUBLE,
    "PI1_UPPER" DOUBLE,
    "PI2_LOWER" DOUBLE,
    "PI2_UPPER" DOUBLE
);
DROP TABLE PAL_SES_STAT_GRP_TBL;
CREATE COLUMN TABLE PAL_SES_STAT_GRP_TBL (
    "GROUP_ID" NVARCHAR(10),
    "STAT_NAME" NVARCHAR(100),
    "STAT_VALUE" DOUBLE
);
CALL _SYS_AFL.PAL_SINGLE_EXPSMOOTH(PAL_SES_DATA_GRP_TBL,
PAL_SES_PARAMETER_GRP_TBL, PAL_SES_RESULT_GRP_TBL, PAL_SES_STAT_GRP_TBL)
    WITH OVERVIEW WITH HINT( PARALLEL_BY_PARAMETER_VALUES(p1.GROUP_ID,
p2.GROUP_ID) );
SELECT * FROM PAL_SES_RESULT_GRP_TBL;
SELECT * FROM PAL_SES_STAT_GRP_TBL;
```

Expected Result:

| | GROUP_ID | TIMESTAMP | VALUE | PI1_LOWER | PI1_UPPER | PI2_LOWER | PI2_UPPER |
|---|---|---|---|---|---|---|---|
| 1 | GROUP_2 | 2 | 200 | ? | ? | ? | ? |
| 2 | GROUP_2 | 3 | 193.5 | ? | ? | ? | ? |
| 3 | GROUP_2 | 4 | 193.65 | ? | ? | ? | ? |
| 4 | GROUP_2 | 5 | 194.035 | ? | ? | ? | ? |
| 5 | GROUP_2 | 6 | 205.6315 | ? | ? | ? | ? |
| 6 | GROUP_2 | 7 | 202.56834999999998 | ? | ? | ? | ? |
| 7 | GROUP_2 | 8 | 197.81151499999999 | ? | ? | ? | ? |
| 8 | GROUP_2 | 9 | 191.0303635 | ? | ? | ? | ? |
| 9 | GROUP_2 | 10 | 193.92732715 | ? | ? | ? | ? |
| 10 | GROUP_2 | 11 | 202.284594435 | ? | ? | ? | ? |
| 11 | GROUP_2 | 12 | 205.55613499150002 | 134.77394650773846 | 276.3383234752616 | 126.70094... | 284.41132977970204 |
| 12 | GROUP_2 | 13 | 205.55613499150002 | 134.4209159462461 | 276.6913540367539 | 126.30764... | 284.80462496156906 |
| 13 | GROUP_2 | 14 | 205.55613499150002 | 134.0696287775175 | 277.04264120548254 | 125.91629... | 285.19597790939497 |
| 14 | GROUP_2 | 15 | 205.55613499150002 | 133.7200594253385 | 277.39221055766154 | 125.52685... | 285.58541711646876 |
| 15 | GROUP_2 | 16 | 205.55613499150002 | 133.3721829327995 | 277.74008705020054 | 125.13929... | 285.97297038614045 |
| 16 | GROUP_2 | 17 | 205.55613499150002 | 133.0259749415023 | 278.0862950414977 | 124.75360... | 286.358664854986 |
| 17 | GROUP_2 | 18 | 205.55613499150002 | 132.68141167165618 | 278.43085831134385 | 124.36974... | 286.74252701498136 |
| 18 | GROUP_2 | 19 | 205.55613499150002 | 132.33846990301677 | 278.7738000799833 | 123.98768... | 287.12458273473743 |
| 19 | GROUP_2 | 20 | 205.55613499150002 | 131.99712695662504 | 279.115143026375 | 123.60741... | 287.50485727984386 |
| 20 | GROUP_2 | 21 | 205.55613499150002 | 131.65736067730512 | 279.4549093056949 | 123.22889... | 287.88337533236745 |
| 21 | GROUP_2 | 22 | 205.55613499150002 | 131.3191494168835 | 279.7931205661165 | 122.85210... | 288.2601610095471 |
| 22 | GROUP_2 | 23 | 205.55613499150002 | 130.98247201809278 | 280.12979796490725 | 122.47703... | 288.6352378817254 |
| 23 | GROUP_1 | 2 | 200 | ? | ? | ? | ? |
| 24 | GROUP_1 | 3 | 193.5 | ? | ? | ? | ? |
| 25 | GROUP_1 | 4 | 193.65 | ? | ? | ? | ? |
| 26 | GROUP_1 | 5 | 194.035 | ? | ? | ? | ? |
| 27 | GROUP_1 | 6 | 205.6315 | ? | ? | ? | ? |
| 28 | GROUP_1 | 7 | 202.56834999999998 | ? | ? | ? | ? |
| 29 | GROUP_1 | 8 | 197.81151499999999 | ? | ? | ? | ? |
| 30 | GROUP_1 | 9 | 191.0303635 | ? | ? | ? | ? |
| 31 | GROUP_1 | 10 | 193.92732715 | ? | ? | ? | ? |
| 32 | GROUP_1 | 11 | 202.284594435 | ? | ? | ? | ? |
| 33 | GROUP_1 | 12 | 205.55613499150002 | 126.700940203298 | 284.41132977970204 | 84.957521... | 326.15474837081297 |
| 34 | GROUP_1 | 13 | 205.55613499150002 | 126.30764502143097 | 284.80462496156906 | 84.356028... | 326.75624144567405 |
| 35 | GROUP_1 | 14 | 205.55613499150002 | 125.91629207360504 | 285.19597790939497 | 83.757505... | 327.35476412989897 |
| 36 | GROUP_1 | 15 | 205.55613499150002 | 125.52685286653129 | 285.58541711646876 | 83.161909... | 327.95036000021264 |
| 37 | GROUP_1 | 16 | 205.55613499150002 | 125.1392995968596 | 285.97297038614045 | 82.569198... | 328.5430715781697 |
| 38 | GROUP_1 | 17 | 205.55613499150002 | 124.75360512801403 | 286.358664854986 | 81.979329... | 329.1329403655816 |
| 39 | GROUP_1 | 18 | 205.55613499150002 | 124.36974296801866 | 286.74252701498136 | 81.392263... | 329.7200068784292 |
| 40 | GROUP_1 | 19 | 205.55613499150002 | 123.98768724826262 | 287.12458273473743 | 80.807959... | 330.30431067933887 |
| 41 | GROUP_1 | 20 | 205.55613499150002 | 123.60741270315617 | 287.50485727984386 | 80.226379... | 330.88589040869545 |
| 42 | GROUP_1 | 21 | 205.55613499150002 | 123.22889465063255 | 287.88337533236745 | 79.647486... | 331.46478381446275 |
| 43 | GROUP_1 | 22 | 205.55613499150002 | 122.85210897345293 | 288.2601610095471 | 79.071242... | 332.04102778077464 |
| 44 | GROUP_1 | 23 | 205.55613499150002 | 122.4770321012746 | 288.6352378817254 | 78.497611... | 332.6146583553595 |

| | GROUP_ID | STAT_NAME | STAT_VALUE |
|---|---|---|---|
| 1 | GROUP_1 | MSE | 3,438.3321253085405 |
| 2 | GROUP_2 | MSE | 3,438.3321253085405 |

Example 2: Parameters shared by groups of data

The data table has the group identifier column. The parameter table doesn't. Then the parameter values will be shared by all data groups and processed together with all data groups. This is suitable for the use case where different data groups will be processed with the same parameter values. Duplication of parameter table rows will cause more memory consumption. Since the parameter table contains only small amount of data, this should not be much extra memory consumption.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_SES_DATA_GRP_TBL;
CREATE COLUMN TABLE PAL_SES_DATA_GRP_TBL (
    "GROUP_ID" NVARCHAR(10),
    "ID" INT,
    "RAWDATA" DOUBLE
);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_1',  1, 200.0);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_1',  2, 135.0);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_1',  3, 195.0);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_1',  4, 197.5);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_1',  5, 310.0);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_1',  6, 175.0);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_1',  7, 155.0);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_1',  8, 130.0);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_1',  9, 220.0);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_1', 10, 277.5);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_1', 11, 235.0);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_2',  1, 200.0);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_2',  2, 135.0);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_2',  3, 195.0);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_2',  4, 197.5);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_2',  5, 310.0);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_2',  6, 175.0);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_2',  7, 155.0);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_2',  8, 130.0);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_2',  9, 220.0);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_2', 10, 277.5);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_2', 11, 235.0);
DROP TABLE PAL_SES_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_SES_PARAMETER_TBL (
    "PARAM_NAME" NVARCHAR(100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR(100)
);
INSERT INTO PAL_SES_PARAMETER_TBL VALUES ('ADAPTIVE_METHOD',0, NULL, NULL);
INSERT INTO PAL_SES_PARAMETER_TBL VALUES ('MEASURE_NAME', NULL, NULL, 'MSE');
INSERT INTO PAL_SES_PARAMETER_TBL VALUES ('ALPHA', NULL,0.1, NULL);
INSERT INTO PAL_SES_PARAMETER_TBL VALUES ('DELTA', NULL,0.2, NULL);
INSERT INTO PAL_SES_PARAMETER_TBL VALUES ('FORECAST_NUM',12, NULL,NULL);
INSERT INTO PAL_SES_PARAMETER_TBL VALUES ('EXPOST_FLAG',1, NULL,NULL);
INSERT INTO PAL_SES_PARAMETER_TBL VALUES ('PREDICTION_CONFIDENCE_1', NULL, 0.8,
NULL);
INSERT INTO PAL_SES_PARAMETER_TBL VALUES ('PREDICTION_CONFIDENCE_2', NULL, 0.95,
NULL);
DROP TABLE PAL_SES_RESULT_GRP_TBL;
CREATE COLUMN TABLE PAL_SES_RESULT_GRP_TBL (
    "GROUP_ID" NVARCHAR(10),
    "TIMESTAMP" INT,
    "VALUE" DOUBLE,
    "PI1_LOWER" DOUBLE,
    "PI1_UPPER" DOUBLE,
    "PI2_LOWER" DOUBLE,
    "PI2_UPPER" DOUBLE
);
DROP TABLE PAL_SES_STAT_GRP_TBL;
CREATE COLUMN TABLE PAL_SES_STAT_GRP_TBL (
    "GROUP_ID" NVARCHAR(10),
    "STAT_NAME" NVARCHAR(100),
```

```
    "STAT_VALUE" DOUBLE
);
CALL _SYS_AFL.PAL_SINGLE_EXPSMOOTH(PAL_SES_DATA_GRP_TBL, PAL_SES_PARAMETER_TBL,
PAL_SES_RESULT_GRP_TBL, PAL_SES_STAT_GRP_TBL)
    WITH OVERVIEW WITH HINT( PARALLEL_BY_PARAMETER_VALUES(p1.GROUP_ID) );
SELECT * FROM PAL_SES_RESULT_GRP_TBL;
SELECT * FROM PAL_SES_STAT_GRP_TBL;
```

**Expected Result:**

| | GROUP_ID | TIMESTAMP | VALUE | PI1_LOWER | PI1_UPPER | PI2_LOWER | PI2_UPPER |
|---|---|---|---|---|---|---|---|
| 1 | GROUP_1 | 2 | 200 | ? | ? | ? | ? |
| 2 | GROUP_1 | 3 | 193.5 | ? | ? | ? | ? |
| 3 | GROUP_1 | 4 | 193.65 | ? | ? | ? | ? |
| 4 | GROUP_1 | 5 | 194.035 | ? | ? | ? | ? |
| 5 | GROUP_1 | 6 | 205.6315 | ? | ? | ? | ? |
| 6 | GROUP_1 | 7 | 202.56834999999998 | ? | ? | ? | ? |
| 7 | GROUP_1 | 8 | 197.81151499999999 | ? | ? | ? | ? |
| 8 | GROUP_1 | 9 | 191.0303635 | ? | ? | ? | ? |
| 9 | GROUP_1 | 10 | 193.92732715 | ? | ? | ? | ? |
| 10 | GROUP_1 | 11 | 202.284594435 | ? | ? | ? | ? |
| 11 | GROUP_1 | 12 | 205.55613499150002 | 126.700940203298 | 284.41132977970204 | 84.9575216121871 | 326.15474837081297 |
| 12 | GROUP_1 | 13 | 205.55613499150002 | 126.30764502143097 | 284.80462496156906 | 84.35602853732595 | 326.75624144567405 |
| 13 | GROUP_1 | 14 | 205.55613499150002 | 125.91629207360504 | 285.19597790939497 | 83.75750585310104 | 327.35476412989897 |
| 14 | GROUP_1 | 15 | 205.55613499150002 | 125.52685286653129 | 285.58541711646876 | 83.16190998278736 | 327.95036000021264 |
| 15 | GROUP_1 | 16 | 205.55613499150002 | 125.1392995968596 | 285.97297038614045 | 82.56919840483033 | 328.5430715781697 |
| 16 | GROUP_1 | 17 | 205.55613499150002 | 124.75360512801403 | 286.358664854986 | 81.97932961741841 | 329.1329403655816 |
| 17 | GROUP_1 | 18 | 205.55613499150002 | 124.36974296801866 | 286.74252701498136 | 81.39226310457079 | 329.7200068784292 |
| 18 | GROUP_1 | 19 | 205.55613499150002 | 123.98768724826262 | 287.12458273473743 | 80.80795930366119 | 330.30431067933887 |
| 19 | GROUP_1 | 20 | 205.55613499150002 | 123.60741270315617 | 287.50485727984386 | 80.22637957430457 | 330.88589040869545 |
| 20 | GROUP_1 | 21 | 205.55613499150002 | 123.22889465063255 | 287.88337533236745 | 79.64748616853727 | 331.46478381446275 |
| 21 | GROUP_1 | 22 | 205.55613499150002 | 122.85210897345293 | 288.2601610095471 | 79.07124220222539 | 332.04102778077464 |
| 22 | GROUP_1 | 23 | 205.55613499150002 | 122.4770321012746 | 288.6352378817254 | 78.49761162764055 | 332.6146583553595 |
| 23 | GROUP_2 | 2 | 200 | ? | ? | ? | ? |
| 24 | GROUP_2 | 3 | 193.5 | ? | ? | ? | ? |
| 25 | GROUP_2 | 4 | 193.65 | ? | ? | ? | ? |
| 26 | GROUP_2 | 5 | 194.035 | ? | ? | ? | ? |
| 27 | GROUP_2 | 6 | 205.6315 | ? | ? | ? | ? |
| 28 | GROUP_2 | 7 | 202.56834999999998 | ? | ? | ? | ? |
| 29 | GROUP_2 | 8 | 197.81151499999999 | ? | ? | ? | ? |
| 30 | GROUP_2 | 9 | 191.0303635 | ? | ? | ? | ? |
| 31 | GROUP_2 | 10 | 193.92732715 | ? | ? | ? | ? |
| 32 | GROUP_2 | 11 | 202.284594435 | ? | ? | ? | ? |
| 33 | GROUP_2 | 12 | 205.55613499150002 | 126.700940203298 | 284.41132977970204 | 84.9575216121871 | 326.15474837081297 |
| 34 | GROUP_2 | 13 | 205.55613499150002 | 126.30764502143097 | 284.80462496156906 | 84.35602853732595 | 326.75624144567405 |
| 35 | GROUP_2 | 14 | 205.55613499150002 | 125.91629207360504 | 285.19597790939497 | 83.75750585310104 | 327.35476412989897 |
| 36 | GROUP_2 | 15 | 205.55613499150002 | 125.52685286653129 | 285.58541711646876 | 83.16190998278736 | 327.95036000021264 |
| 37 | GROUP_2 | 16 | 205.55613499150002 | 125.1392995968596 | 285.97297038614045 | 82.56919840483033 | 328.5430715781697 |
| 38 | GROUP_2 | 17 | 205.55613499150002 | 124.75360512801403 | 286.358664854986 | 81.97932961741841 | 329.1329403655816 |
| 39 | GROUP_2 | 18 | 205.55613499150002 | 124.36974296801866 | 286.74252701498136 | 81.39226310457079 | 329.7200068784292 |
| 40 | GROUP_2 | 19 | 205.55613499150002 | 123.98768724826262 | 287.12458273473743 | 80.80795930366119 | 330.30431067933887 |
| 41 | GROUP_2 | 20 | 205.55613499150002 | 123.60741270315617 | 287.50485727984386 | 80.22637957430457 | 330.88589040869545 |
| 42 | GROUP_2 | 21 | 205.55613499150002 | 123.22889465063255 | 287.88337533236745 | 79.64748616853727 | 331.46478381446275 |
| 43 | GROUP_2 | 22 | 205.55613499150002 | 122.85210897345293 | 288.2601610095471 | 79.07124220222539 | 332.04102778077464 |
| 44 | GROUP_2 | 23 | 205.55613499150002 | 122.4770321012746 | 288.6352378817254 | 78.49761162764055 | 332.6146583553595 |

| | GROUP_ID | STAT_NAME | STAT_VALUE |
|---|---|---|---|
| 1 | GROUP_2 | MSE | 3,438.3321253085405 |
| 2 | GROUP_1 | MSE | 3,438.3321253085405 |

**Example 3 Data shared by groups of parameters**

The data table doesn't have the group identifier column. The parameter table has one. Then all the data rows will be mapped to each parameter group. As the data table has big volume, duplication of its data is expected to have higher memory consumption.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_SES_DATA_GRP_TBL;
CREATE COLUMN TABLE PAL_SES_DATA_GRP_TBL (
    "ID" INT,
    "RAWDATA" DOUBLE
);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ( 1, 200.0);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ( 2, 135.0);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ( 3, 195.0);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ( 4, 197.5);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ( 5, 310.0);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ( 6, 175.0);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ( 7, 155.0);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ( 8, 130.0);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ( 9, 220.0);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES (10, 277.5);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES (11, 235.0);
DROP TABLE PAL_SES_PARAMETER_GRP_TBL;
CREATE COLUMN TABLE PAL_SES_PARAMETER_GRP_TBL (
    "GROUP_ID" NVARCHAR(10),
    "PARAM_NAME" NVARCHAR(100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR(100)
);
INSERT INTO PAL_SES_PARAMETER_GRP_TBL VALUES ('GROUP_1', 'ADAPTIVE_METHOD',0,
NULL, NULL);
INSERT INTO PAL_SES_PARAMETER_GRP_TBL VALUES ('GROUP_1', 'MEASURE_NAME', NULL,
NULL, 'MSE');
INSERT INTO PAL_SES_PARAMETER_GRP_TBL VALUES ('GROUP_1', 'ALPHA', NULL,0.1,
NULL);
INSERT INTO PAL_SES_PARAMETER_GRP_TBL VALUES ('GROUP_1', 'DELTA', NULL,0.2,
NULL);
INSERT INTO PAL_SES_PARAMETER_GRP_TBL VALUES ('GROUP_1', 'FORECAST_NUM',12,
NULL,NULL);
INSERT INTO PAL_SES_PARAMETER_GRP_TBL VALUES ('GROUP_1', 'EXPOST_FLAG',1,
NULL,NULL);
INSERT INTO PAL_SES_PARAMETER_GRP_TBL VALUES ('GROUP_1',
'PREDICTION_CONFIDENCE_1', NULL, 0.8, NULL);
INSERT INTO PAL_SES_PARAMETER_GRP_TBL VALUES ('GROUP_1',
'PREDICTION_CONFIDENCE_2', NULL, 0.95, NULL);
INSERT INTO PAL_SES_PARAMETER_GRP_TBL VALUES ('GROUP_2', 'ADAPTIVE_METHOD',0,
NULL, NULL);
INSERT INTO PAL_SES_PARAMETER_GRP_TBL VALUES ('GROUP_2', 'MEASURE_NAME', NULL,
NULL, 'MSE');
INSERT INTO PAL_SES_PARAMETER_GRP_TBL VALUES ('GROUP_2', 'ALPHA', NULL,0.1,
NULL);
INSERT INTO PAL_SES_PARAMETER_GRP_TBL VALUES ('GROUP_2', 'DELTA', NULL,0.2,
NULL);
INSERT INTO PAL_SES_PARAMETER_GRP_TBL VALUES ('GROUP_2', 'FORECAST_NUM',12,
NULL,NULL);
INSERT INTO PAL_SES_PARAMETER_GRP_TBL VALUES ('GROUP_2', 'EXPOST_FLAG',1,
NULL,NULL);
```

```
INSERT INTO PAL_SES_PARAMETER_GRP_TBL VALUES ('GROUP_2',
'PREDICTION_CONFIDENCE_1', NULL, 0.75, NULL);
INSERT INTO PAL_SES_PARAMETER_GRP_TBL VALUES ('GROUP_2',
'PREDICTION_CONFIDENCE_2', NULL, 0.8, NULL);
DROP TABLE PAL_SES_RESULT_GRP_TBL;
CREATE COLUMN TABLE PAL_SES_RESULT_GRP_TBL (
    "GROUP_ID" NVARCHAR(10),
    "TIMESTAMP" INT,
    "VALUE" DOUBLE,
    "PI1_LOWER" DOUBLE,
    "PI1_UPPER" DOUBLE,
    "PI2_LOWER" DOUBLE,
    "PI2_UPPER" DOUBLE
);
DROP TABLE PAL_SES_STAT_GRP_TBL;
CREATE COLUMN TABLE PAL_SES_STAT_GRP_TBL (
    "GROUP_ID" NVARCHAR(10),
    "STAT_NAME" NVARCHAR(100),
    "STAT_VALUE" DOUBLE
);
CALL _SYS_AFL.PAL_SINGLE_EXPSMOOTH(PAL_SES_DATA_GRP_TBL,
PAL_SES_PARAMETER_GRP_TBL, PAL_SES_RESULT_GRP_TBL, PAL_SES_STAT_GRP_TBL)
    WITH OVERVIEW WITH HINT( PARALLEL_BY_PARAMETER_VALUES(p2.GROUP_ID) );
SELECT * FROM PAL_SES_RESULT_GRP_TBL;
SELECT * FROM PAL_SES_STAT_GRP_TBL;
```

### Expected Result

| | GROUP_ID | TIMESTAMP | VALUE | PI1_LOWER | PI1_UPPER | PI2_LOWER | PI2_UPPER |
|---|---|---|---|---|---|---|---|
| 1 | GROUP_2 | 2 | 200 | ? | ? | ? | ? |
| 2 | GROUP_2 | 3 | 193.5 | ? | ? | ? | ? |
| 3 | GROUP_2 | 4 | 193.65 | ? | ? | ? | ? |
| 4 | GROUP_2 | 5 | 194.035 | ? | ? | ? | ? |
| 5 | GROUP_2 | 6 | 205.6315 | ? | ? | ? | ? |
| 6 | GROUP_2 | 7 | 202.56834999999998 | ? | ? | ? | ? |
| 7 | GROUP_2 | 8 | 197.81151499999999 | ? | ? | ? | ? |
| 8 | GROUP_2 | 9 | 191.0303635 | ? | ? | ? | ? |
| 9 | GROUP_2 | 10 | 193.92732715 | ? | ? | ? | ? |
| 10 | GROUP_2 | 11 | 202.284594435 | ? | ? | ? | ? |
| 11 | GROUP_2 | 12 | 205.55613499150002 | 134.77394650773846 | 276.3383234752616 | 126.700940203298 | 284.41132977970204 |
| 12 | GROUP_2 | 13 | 205.55613499150002 | 134.4209159462461 | 276.6913540367539 | 126.30764502143097 | 284.80462496156906 |
| 13 | GROUP_2 | 14 | 205.55613499150002 | 134.0696287775175 | 277.04264120548254 | 125.91629207360504 | 285.19597790939497 |
| 14 | GROUP_2 | 15 | 205.55613499150002 | 133.7200594253385 | 277.39221055766154 | 125.52685286653129 | 285.58541711646876 |
| 15 | GROUP_2 | 16 | 205.55613499150002 | 133.3721829327995 | 277.74008705020054 | 125.1392995968596 | 285.97297038614045 |
| 16 | GROUP_2 | 17 | 205.55613499150002 | 133.0259749415023 | 278.0862950414977 | 124.75360512801403 | 286.358664854986 |
| 17 | GROUP_2 | 18 | 205.55613499150002 | 132.68141167165618 | 278.43085831134385 | 124.36974296801866 | 286.74252701498136 |
| 18 | GROUP_2 | 19 | 205.55613499150002 | 132.33846990301677 | 278.7738000799833 | 123.98768724826262 | 287.12458273473743 |
| 19 | GROUP_2 | 20 | 205.55613499150002 | 131.99712695662504 | 279.115143026375 | 123.60741270315617 | 287.50485727984386 |
| 20 | GROUP_2 | 21 | 205.55613499150002 | 131.65736067730512 | 279.4549093056949 | 123.22889465063255 | 287.88337533236745 |
| 21 | GROUP_2 | 22 | 205.55613499150002 | 131.3191494168835 | 279.7931205661165 | 122.85210897345293 | 288.2601610095471 |
| 22 | GROUP_2 | 23 | 205.55613499150002 | 130.98247201809278 | 280.12979796490725 | 122.4770321012746 | 288.6352378817254 |
| 23 | GROUP_1 | 2 | 200 | ? | ? | ? | ? |
| 24 | GROUP_1 | 3 | 193.5 | ? | ? | ? | ? |
| 25 | GROUP_1 | 4 | 193.65 | ? | ? | ? | ? |
| 26 | GROUP_1 | 5 | 194.035 | ? | ? | ? | ? |
| 27 | GROUP_1 | 6 | 205.6315 | ? | ? | ? | ? |
| 28 | GROUP_1 | 7 | 202.56834999999998 | ? | ? | ? | ? |
| 29 | GROUP_1 | 8 | 197.81151499999999 | ? | ? | ? | ? |
| 30 | GROUP_1 | 9 | 191.0303635 | ? | ? | ? | ? |
| 31 | GROUP_1 | 10 | 193.92732715 | ? | ? | ? | ? |
| 32 | GROUP_1 | 11 | 202.284594435 | ? | ? | ? | ? |
| 33 | GROUP_1 | 12 | 205.55613499150002 | 126.700940203298 | 284.41132977970204 | 84.9575216121871 | 326.15474837081297 |
| 34 | GROUP_1 | 13 | 205.55613499150002 | 126.30764502143097 | 284.80462496156906 | 84.35602853732595 | 326.75624144567405 |
| 35 | GROUP_1 | 14 | 205.55613499150002 | 125.91629207360504 | 285.19597790939497 | 83.75750585310104 | 327.35476412989897 |
| 36 | GROUP_1 | 15 | 205.55613499150002 | 125.52685286653129 | 285.58541711646876 | 83.16190998278736 | 327.95036000021264 |
| 37 | GROUP_1 | 16 | 205.55613499150002 | 125.1392995968596 | 285.97297038614045 | 82.56919840483033 | 328.5430715781697 |
| 38 | GROUP_1 | 17 | 205.55613499150002 | 124.75360512801403 | 286.358664854986 | 81.97932961741841 | 329.1329403655816 |
| 39 | GROUP_1 | 18 | 205.55613499150002 | 124.36974296801866 | 286.74252701498136 | 81.39226310457079 | 329.7200068784292 |
| 40 | GROUP_1 | 19 | 205.55613499150002 | 123.98768724826262 | 287.12458273473743 | 80.80795930366119 | 330.30431067933887 |
| 41 | GROUP_1 | 20 | 205.55613499150002 | 123.60741270315617 | 287.50485727984386 | 80.22637957430457 | 330.88589040869545 |
| 42 | GROUP_1 | 21 | 205.55613499150002 | 123.22889465063255 | 287.88337533236745 | 79.64748616853727 | 331.46478381446275 |
| 43 | GROUP_1 | 22 | 205.55613499150002 | 122.85210897345293 | 288.2601610095471 | 79.07124220222539 | 332.04102778077464 |
| 44 | GROUP_1 | 23 | 205.55613499150002 | 122.4770321012746 | 288.6352378817254 | 78.49761162764055 | 332.6146583553595 |

| | GROUP_ID | STAT_NAME | STAT_VALUE |
|---|---|---|---|
| 1 | GROUP_1 | MSE | 3,438.3321253085405 |
| 2 | GROUP_2 | MSE | 3,438.3321253085405 |

To use this hint, there should be one or more group identifier columns in at least one input table. And all output tables should have the corresponding group identifier columns. The types of the columns in each table should be identical. The group identifier columns in each input table must be specified inside the hint. Data in different tables and with the same group identifier will be processed together.

Some tables can have no group identifier column. These tables should not appear in the hint. All rows of these tables are mapped to each group. A copy of all the data is processed together with each group. As data duplication is needed, it's possible to consume higher volume of memory.

## 3.4 Calling PAL Procedures in Parallel with MAP_REDUCE

MAP_REDUCE allows you to embed procedures directly without table functions. It has a built-in grouping algorithm for parallelization.

**Example 1: Shared parameters**

```
SET SCHEMA DM_PAL;
DROP TYPE PAL_SES_DATA_T;
CREATE TYPE PAL_SES_DATA_T AS TABLE (
    "ID" INT,
    "RAWDATA" DOUBLE
);
DROP TYPE PAL_SES_PARAMETER_T;
CREATE TYPE PAL_SES_PARAMETER_T AS TABLE (
    "PARAM_NAME" NVARCHAR(100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR(100)
);
DROP TYPE PAL_SES_DATA_GRP_T;
CREATE TYPE PAL_SES_DATA_GRP_T AS TABLE (
    "GROUP_ID" NVARCHAR(10),
    "ID" INT,
    "RAWDATA" DOUBLE
);
DROP TYPE PAL_SES_RESULT_GRP_T;
CREATE TYPE PAL_SES_RESULT_GRP_T AS TABLE (
    "GROUP_ID" NVARCHAR(10),
    "TIMESTAMP" INT,
    "VALUE" DOUBLE,
    "PI1_LOWER" DOUBLE,
    "PI1_UPPER" DOUBLE,
    "PI2_LOWER" DOUBLE,
    "PI2_UPPER" DOUBLE
);
DROP TYPE PAL_SES_STAT_GRP_T;
CREATE TYPE PAL_SES_STAT_GRP_T AS TABLE (
    "GROUP_ID" NVARCHAR(10),
    "STAT_NAME" NVARCHAR(100),
    "STAT_VALUE" DOUBLE
);
DROP TABLE PAL_SES_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_SES_PARAMETER_TBL LIKE PAL_SES_PARAMETER_T;
INSERT INTO PAL_SES_PARAMETER_TBL VALUES ('ADAPTIVE_METHOD',0, NULL, NULL);
INSERT INTO PAL_SES_PARAMETER_TBL VALUES ('MEASURE_NAME', NULL, NULL, 'MSE');
INSERT INTO PAL_SES_PARAMETER_TBL VALUES ('ALPHA', NULL,0.1, NULL);
INSERT INTO PAL_SES_PARAMETER_TBL VALUES ('DELTA', NULL,0.2, NULL);
INSERT INTO PAL_SES_PARAMETER_TBL VALUES ('FORECAST_NUM',12, NULL,NULL);
INSERT INTO PAL_SES_PARAMETER_TBL VALUES ('EXPOST_FLAG',1, NULL,NULL);
INSERT INTO PAL_SES_PARAMETER_TBL VALUES ('PREDICTION_CONFIDENCE_1', NULL, 0.8,
NULL);
INSERT INTO PAL_SES_PARAMETER_TBL VALUES ('PREDICTION_CONFIDENCE_2', NULL, 0.95,
NULL);
DROP TABLE PAL_SES_DATA_GRP_TBL;
CREATE COLUMN TABLE PAL_SES_DATA_GRP_TBL LIKE PAL_SES_DATA_GRP_T;
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_1',  1, 200.0);
```

```
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_1',  2, 135.0);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_1',  3, 195.0);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_1',  4, 197.5);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_1',  5, 310.0);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_1',  6, 175.0);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_1',  7, 155.0);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_1',  8, 130.0);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_1',  9, 220.0);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_1', 10, 277.5);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_1', 11, 235.0);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_2',  1, 200.0);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_2',  2, 135.0);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_2',  3, 195.0);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_2',  4, 197.5);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_2',  5, 310.0);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_2',  6, 175.0);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_2',  7, 155.0);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_2',  8, 130.0);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_2',  9, 220.0);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_2', 10, 277.5);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_2', 11, 235.0);
-- Mapper
DROP FUNCTION SINGLE_EXPSMOOTH_MAPPER;
CREATE FUNCTION SINGLE_EXPSMOOTH_MAPPER(
  dummy_value INT,
  it_data     PAL_SES_DATA_GRP_T
) RETURNS PAL_SES_DATA_GRP_T AS
BEGIN
  RETURN :it_data;
END;
-- Reducer
DROP PROCEDURE SINGLE_EXPSMOOTH_REDUCER;
CREATE PROCEDURE SINGLE_EXPSMOOTH_REDUCER(
  IN  iv_group_id   NVARCHAR(10),
  IN  it_data       PAL_SES_DATA_T,
  IN  it_param      PAL_SES_PARAMETER_T,
  OUT ot_result     PAL_SES_RESULT_GRP_T,
  OUT ot_stat       PAL_SES_STAT_GRP_T
) READS SQL DATA AS
BEGIN
  CALL _SYS_AFL.PAL_SINGLE_EXPSMOOTH(:it_data, :it_param, lt_result, lt_stat);
  ot_result = SELECT :iv_group_id AS "GROUP_ID", "TIMESTAMP", "VALUE",
"PI1_LOWER", "PI1_UPPER", "PI2_LOWER", "PI2_UPPER"FROM :lt_result;
  ot_stat   = SELECT :iv_group_id AS "GROUP_ID", "STAT_NAME", "STAT_VALUE"
FROM :lt_stat;
END;
-- Main procedure
DROP PROCEDURE SINGLE_EXPSMOOTH_MAIN;
CREATE PROCEDURE SINGLE_EXPSMOOTH_MAIN(
  IN  it_group_data   PAL_SES_DATA_GRP_T,
  OUT ot_group_result PAL_SES_RESULT_GRP_T,
  OUT ot_group_stat   PAL_SES_STAT_GRP_T
) AS
BEGIN
  DECLARE lt_group_result PAL_SES_RESULT_GRP_T;
  DECLARE lt_group_stat   PAL_SES_STAT_GRP_T;
  lt_dummy = SELECT 1 a FROM dummy;
  lt_param = SELECT "PARAM_NAME", "INT_VALUE", "DOUBLE_VALUE", "STRING_VALUE"
FROM PAL_SES_PARAMETER_TBL;
  MAP_REDUCE(:lt_dummy, SINGLE_EXPSMOOTH_MAPPER(:lt_dummy.a, :it_group_data)
GROUP BY "GROUP_ID" AS "GROUP_DATA",
                        SINGLE_EXPSMOOTH_REDUCER("GROUP_DATA"."GROUP_ID",
"GROUP_DATA", :lt_param, lt_group_result, lt_group_stat) );
  ot_group_result = SELECT * FROM :lt_group_result;
  ot_group_stat   = SELECT * FROM :lt_group_stat;
END;
CALL SINGLE_EXPSMOOTH_MAIN(PAL_SES_DATA_GRP_TBL, ?, ?);
```

### Expected Result:

| | GROUP_ID | TIMESTAMP | VALUE | PI1_LOWER | PI1_UPPER | PI2_LOWER | PI2_UPPER |
|---|---|---|---|---|---|---|---|
| 1 | GROUP_2 | 2 | 200 | ? | ? | ? | ? |
| 2 | GROUP_2 | 3 | 193.5 | ? | ? | ? | ? |
| 3 | GROUP_2 | 4 | 193.65 | ? | ? | ? | ? |
| 4 | GROUP_2 | 5 | 194.035 | ? | ? | ? | ? |
| 5 | GROUP_2 | 6 | 205.6315 | ? | ? | ? | ? |
| 6 | GROUP_2 | 7 | 202.56834999999998 | ? | ? | ? | ? |
| 7 | GROUP_2 | 8 | 197.81151499999999 | ? | ? | ? | ? |
| 8 | GROUP_2 | 9 | 191.0303635 | ? | ? | ? | ? |
| 9 | GROUP_2 | 10 | 193.92732715 | ? | ? | ? | ? |
| 10 | GROUP_2 | 11 | 202.284594435 | ? | ? | ? | ? |
| 11 | GROUP_2 | 12 | 205.55613499150002 | 126.700940203298 | 284.41132977970204 | 84.9575216121871 | 326.15474837081297 |
| 12 | GROUP_2 | 13 | 205.55613499150002 | 126.30764502143097 | 284.80462496156906 | 84.35602853732595 | 326.75624144567405 |
| 13 | GROUP_2 | 14 | 205.55613499150002 | 125.91629207360504 | 285.19597790939497 | 83.75750585310104 | 327.35476412989897 |
| 14 | GROUP_2 | 15 | 205.55613499150002 | 125.52685286653129 | 285.58541711646876 | 83.16190998278736 | 327.95036000021264 |
| 15 | GROUP_2 | 16 | 205.55613499150002 | 125.1392995968596 | 285.97297038614045 | 82.56919840483033 | 328.5430715781697 |
| 16 | GROUP_2 | 17 | 205.55613499150002 | 124.75360512801403 | 286.358664854986 | 81.97932961741841 | 329.1329403655816 |
| 17 | GROUP_2 | 18 | 205.55613499150002 | 124.36974296801866 | 286.74252701498136 | 81.39226310457079 | 329.7200068784292 |
| 18 | GROUP_2 | 19 | 205.55613499150002 | 123.98768724826262 | 287.12458273473743 | 80.80795930366119 | 330.30431067933887 |
| 19 | GROUP_2 | 20 | 205.55613499150002 | 123.60741270315617 | 287.50485727984386 | 80.22637957430457 | 330.88589040869545 |
| 20 | GROUP_2 | 21 | 205.55613499150002 | 123.22889465063255 | 287.88337533236745 | 79.64748616853727 | 331.46478381446275 |
| 21 | GROUP_2 | 22 | 205.55613499150002 | 122.85210897345293 | 288.2601610095471 | 79.07124220222539 | 332.04102778077464 |
| 22 | GROUP_2 | 23 | 205.55613499150002 | 122.4770321012746 | 288.6352378817254 | 78.49761162764055 | 332.6146583553595 |
| 23 | GROUP_1 | 2 | 200 | ? | ? | ? | ? |
| 24 | GROUP_1 | 3 | 193.5 | ? | ? | ? | ? |
| 25 | GROUP_1 | 4 | 193.65 | ? | ? | ? | ? |
| 26 | GROUP_1 | 5 | 194.035 | ? | ? | ? | ? |
| 27 | GROUP_1 | 6 | 205.6315 | ? | ? | ? | ? |
| 28 | GROUP_1 | 7 | 202.56834999999998 | ? | ? | ? | ? |
| 29 | GROUP_1 | 8 | 197.81151499999999 | ? | ? | ? | ? |
| 30 | GROUP_1 | 9 | 191.0303635 | ? | ? | ? | ? |
| 31 | GROUP_1 | 10 | 193.92732715 | ? | ? | ? | ? |
| 32 | GROUP_1 | 11 | 202.284594435 | ? | ? | ? | ? |
| 33 | GROUP_1 | 12 | 205.55613499150002 | 126.700940203298 | 284.41132977970204 | 84.9575216121871 | 326.15474837081297 |
| 34 | GROUP_1 | 13 | 205.55613499150002 | 126.30764502143097 | 284.80462496156906 | 84.35602853732595 | 326.75624144567405 |
| 35 | GROUP_1 | 14 | 205.55613499150002 | 125.91629207360504 | 285.19597790939497 | 83.75750585310104 | 327.35476412989897 |
| 36 | GROUP_1 | 15 | 205.55613499150002 | 125.52685286653129 | 285.58541711646876 | 83.16190998278736 | 327.95036000021264 |
| 37 | GROUP_1 | 16 | 205.55613499150002 | 125.1392995968596 | 285.97297038614045 | 82.56919840483033 | 328.5430715781697 |
| 38 | GROUP_1 | 17 | 205.55613499150002 | 124.75360512801403 | 286.358664854986 | 81.97932961741841 | 329.1329403655816 |
| 39 | GROUP_1 | 18 | 205.55613499150002 | 124.36974296801866 | 286.74252701498136 | 81.39226310457079 | 329.7200068784292 |
| 40 | GROUP_1 | 19 | 205.55613499150002 | 123.98768724826262 | 287.12458273473743 | 80.80795930366119 | 330.30431067933887 |
| 41 | GROUP_1 | 20 | 205.55613499150002 | 123.60741270315617 | 287.50485727984386 | 80.22637957430457 | 330.88589040869545 |
| 42 | GROUP_1 | 21 | 205.55613499150002 | 123.22889465063255 | 287.88337533236745 | 79.64748616853727 | 331.46478381446275 |
| 43 | GROUP_1 | 22 | 205.55613499150002 | 122.85210897345293 | 288.2601610095471 | 79.07124220222539 | 332.04102778077464 |
| 44 | GROUP_1 | 23 | 205.55613499150002 | 122.4770321012746 | 288.6352378817254 | 78.49761162764055 | 332.6146583553595 |

| | GROUP_ID | STAT_NAME | STAT_VALUE |
|---|---|---|---|
| 1 | GROUP_2 | MSE | 3,438.3321253085405 |
| 2 | GROUP_1 | MSE | 3,438.3321253085405 |

### Example 2: Group specific parameters

```
SET SCHEMA DM_PAL;
DROP TYPE PAL_SES_DATA_T;
CREATE TYPE PAL_SES_DATA_T AS TABLE (
```

```
    "ID" INT,
    "RAWDATA" DOUBLE
);
DROP TYPE PAL_SES_PARAMETER_T;
CREATE TYPE PAL_SES_PARAMETER_T AS TABLE (
    "PARAM_NAME" NVARCHAR(100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR(100)
);
DROP TYPE PAL_SES_DATA_GRP_T;
CREATE TYPE PAL_SES_DATA_GRP_T AS TABLE (
    "GROUP_ID" NVARCHAR(10),
    "ID" INT,
    "RAWDATA" DOUBLE
);
DROP TYPE PAL_SES_PARAMETER_GRP_T;
CREATE TYPE PAL_SES_PARAMETER_GRP_T AS TABLE (
    "GROUP_ID" NVARCHAR(10),
    "PARAM_NAME" NVARCHAR(100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR(100)
);
DROP TYPE PAL_SES_RESULT_GRP_T;
CREATE TYPE PAL_SES_RESULT_GRP_T AS TABLE (
    "GROUP_ID" NVARCHAR(10),
    "TIMESTAMP" INT,
    "VALUE" DOUBLE,
    "PI1_LOWER" DOUBLE,
    "PI1_UPPER" DOUBLE,
    "PI2_LOWER" DOUBLE,
    "PI2_UPPER" DOUBLE
);
DROP TYPE PAL_SES_STAT_GRP_T;
CREATE TYPE PAL_SES_STAT_GRP_T AS TABLE (
    "GROUP_ID" NVARCHAR(10),
    "STAT_NAME" NVARCHAR(100),
    "STAT_VALUE" DOUBLE
);
DROP TABLE PAL_SES_PARAMETER_GRP_TBL;
CREATE COLUMN TABLE PAL_SES_PARAMETER_GRP_TBL LIKE PAL_SES_PARAMETER_GRP_T;
INSERT INTO PAL_SES_PARAMETER_GRP_TBL VALUES ('GROUP_1', 'ADAPTIVE_METHOD',0,
NULL, NULL);
INSERT INTO PAL_SES_PARAMETER_GRP_TBL VALUES ('GROUP_1', 'MEASURE_NAME', NULL,
NULL, 'MSE');
INSERT INTO PAL_SES_PARAMETER_GRP_TBL VALUES ('GROUP_1', 'ALPHA', NULL,0.1,
NULL);
INSERT INTO PAL_SES_PARAMETER_GRP_TBL VALUES ('GROUP_1', 'DELTA', NULL,0.2,
NULL);
INSERT INTO PAL_SES_PARAMETER_GRP_TBL VALUES ('GROUP_1', 'FORECAST_NUM',12,
NULL,NULL);
INSERT INTO PAL_SES_PARAMETER_GRP_TBL VALUES ('GROUP_1', 'EXPOST_FLAG',1,
NULL,NULL);
INSERT INTO PAL_SES_PARAMETER_GRP_TBL VALUES ('GROUP_1',
'PREDICTION_CONFIDENCE_1', NULL, 0.8, NULL);
INSERT INTO PAL_SES_PARAMETER_GRP_TBL VALUES ('GROUP_1',
'PREDICTION_CONFIDENCE_2', NULL, 0.95, NULL);
INSERT INTO PAL_SES_PARAMETER_GRP_TBL VALUES ('GROUP_2', 'ADAPTIVE_METHOD',0,
NULL, NULL);
INSERT INTO PAL_SES_PARAMETER_GRP_TBL VALUES ('GROUP_2', 'MEASURE_NAME', NULL,
NULL, 'MSE');
INSERT INTO PAL_SES_PARAMETER_GRP_TBL VALUES ('GROUP_2', 'ALPHA', NULL,0.1,
NULL);
INSERT INTO PAL_SES_PARAMETER_GRP_TBL VALUES ('GROUP_2', 'DELTA', NULL,0.2,
NULL);
INSERT INTO PAL_SES_PARAMETER_GRP_TBL VALUES ('GROUP_2', 'FORECAST_NUM',12,
NULL,NULL);
```

```
INSERT INTO PAL_SES_PARAMETER_GRP_TBL VALUES ('GROUP_2', 'EXPOST_FLAG',1,
NULL,NULL);
INSERT INTO PAL_SES_PARAMETER_GRP_TBL VALUES ('GROUP_2',
'PREDICTION_CONFIDENCE_1', NULL, 0.75, NULL);
INSERT INTO PAL_SES_PARAMETER_GRP_TBL VALUES ('GROUP_2',
'PREDICTION_CONFIDENCE_2', NULL, 0.8, NULL);
DROP TABLE PAL_SES_DATA_GRP_TBL;
CREATE COLUMN TABLE PAL_SES_DATA_GRP_TBL LIKE PAL_SES_DATA_GRP_T;
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_1',  1, 200.0);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_1',  2, 135.0);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_1',  3, 195.0);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_1',  4, 197.5);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_1',  5, 310.0);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_1',  6, 175.0);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_1',  7, 155.0);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_1',  8, 130.0);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_1',  9, 220.0);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_1', 10, 277.5);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_1', 11, 235.0);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_2',  1, 200.0);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_2',  2, 135.0);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_2',  3, 195.0);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_2',  4, 197.5);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_2',  5, 310.0);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_2',  6, 175.0);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_2',  7, 155.0);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_2',  8, 130.0);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_2',  9, 220.0);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_2', 10, 277.5);
INSERT INTO PAL_SES_DATA_GRP_TBL VALUES ('GROUP_2', 11, 235.0);
-- Mapper
DROP FUNCTION SINGLE_EXPSMOOTH_MAPPER;
CREATE FUNCTION SINGLE_EXPSMOOTH_MAPPER(
  dummy_value INT,
  it_data     PAL_SES_DATA_GRP_T
) RETURNS PAL_SES_DATA_GRP_T AS
BEGIN
  RETURN :it_data;
END;
-- Reducer
DROP PROCEDURE SINGLE_EXPSMOOTH_REDUCER;
CREATE PROCEDURE SINGLE_EXPSMOOTH_REDUCER(
  IN  iv_group_id   NVARCHAR(10),
  IN  it_data       PAL_SES_DATA_T,
  OUT ot_result     PAL_SES_RESULT_GRP_T,
  OUT ot_stat       PAL_SES_STAT_GRP_T
) READS SQL DATA AS
BEGIN
  lt_param = SELECT "PARAM_NAME", "INT_VALUE", "DOUBLE_VALUE", "STRING_VALUE"
FROM PAL_SES_PARAMETER_GRP_TBL WHERE "GROUP_ID" = :iv_group_id;
  CALL _SYS_AFL.PAL_SINGLE_EXPSMOOTH(:it_data, :lt_param, lt_result, lt_stat);
  ot_result = SELECT :iv_group_id AS "GROUP_ID", "TIMESTAMP", "VALUE",
"PI1_LOWER", "PI1_UPPER", "PI2_LOWER", "PI2_UPPER"FROM :lt_result;
  ot_stat   = SELECT :iv_group_id AS "GROUP_ID", "STAT_NAME", "STAT_VALUE"
FROM :lt_stat;
END;
-- Main procedure
DROP PROCEDURE SINGLE_EXPSMOOTH_MAIN;
CREATE PROCEDURE SINGLE_EXPSMOOTH_MAIN(
  IN  it_group_data   PAL_SES_DATA_GRP_T,
  OUT ot_group_result PAL_SES_RESULT_GRP_T,
  OUT ot_group_stat   PAL_SES_STAT_GRP_T
) AS
BEGIN
  DECLARE lt_group_result PAL_SES_RESULT_GRP_T;
  DECLARE lt_group_stat   PAL_SES_STAT_GRP_T;
  lt_dummy = SELECT 1 a FROM dummy;
```

```
  MAP_REDUCE(:lt_dummy, SINGLE_EXPSMOOTH_MAPPER(:lt_dummy.a, :it_group_data)
GROUP BY "GROUP_ID" AS "GROUP_DATA",
                        SINGLE_EXPSMOOTH_REDUCER("GROUP_DATA"."GROUP_ID",
"GROUP_DATA", lt_group_result, lt_group_stat) );
  ot_group_result = SELECT * FROM :lt_group_result;
  ot_group_stat   = SELECT * FROM :lt_group_stat;
END;
CALL SINGLE_EXPSMOOTH_MAIN(PAL_SES_DATA_GRP_TBL, ?, ?);
```

**Expected Result:**

| | GROUP_ID | TIMESTAMP | VALUE | PI1_LOWER | PI1_UPPER | PI2_LOWER | PI2_UPPER |
|---|---|---|---|---|---|---|---|
| 1 | GROUP_2 | 2 | 200 | ? | ? | ? | ? |
| 2 | GROUP_2 | 3 | 193.5 | ? | ? | ? | ? |
| 3 | GROUP_2 | 4 | 193.65 | ? | ? | ? | ? |
| 4 | GROUP_2 | 5 | 194.035 | ? | ? | ? | ? |
| 5 | GROUP_2 | 6 | 205.6315 | ? | ? | ? | ? |
| 6 | GROUP_2 | 7 | 202.56834999999998 | ? | ? | ? | ? |
| 7 | GROUP_2 | 8 | 197.81151499999999 | ? | ? | ? | ? |
| 8 | GROUP_2 | 9 | 191.0303635 | ? | ? | ? | ? |
| 9 | GROUP_2 | 10 | 193.92732715 | ? | ? | ? | ? |
| 10 | GROUP_2 | 11 | 202.284594435 | ? | ? | ? | ? |
| 11 | GROUP_2 | 12 | 205.55613499150002 | 134.77394650773846 | 276.3383... | 126.700940203298 | 284.41132977970204 |
| 12 | GROUP_2 | 13 | 205.55613499150002 | 134.4209159462461 | 276.6913... | 126.30764502143097 | 284.80462496156906 |
| 13 | GROUP_2 | 14 | 205.55613499150002 | 134.0696287775175 | 277.0426... | 125.91629207360504 | 285.19597790939497 |
| 14 | GROUP_2 | 15 | 205.55613499150002 | 133.7200594253385 | 277.3922... | 125.52685286653129 | 285.58541711646876 |
| 15 | GROUP_2 | 16 | 205.55613499150002 | 133.3721829327995 | 277.7400... | 125.1392995968596 | 285.97297038614045 |
| 16 | GROUP_2 | 17 | 205.55613499150002 | 133.0259749415023 | 278.0862... | 124.75360512801403 | 286.358664854986 |
| 17 | GROUP_2 | 18 | 205.55613499150002 | 132.68141167165618 | 278.4308... | 124.36974296801866 | 286.74252701498136 |
| 18 | GROUP_2 | 19 | 205.55613499150002 | 132.33846990301677 | 278.7738... | 123.98768724826262 | 287.12458273473743 |
| 19 | GROUP_2 | 20 | 205.55613499150002 | 131.99712695662504 | 279.1151... | 123.60741270315617 | 287.50485727984386 |
| 20 | GROUP_2 | 21 | 205.55613499150002 | 131.65736067730512 | 279.4549... | 123.22889465063255 | 287.88337533236745 |
| 21 | GROUP_2 | 22 | 205.55613499150002 | 131.3191494168835 | 279.7931... | 122.85210897345293 | 288.2601610095471 |
| 22 | GROUP_2 | 23 | 205.55613499150002 | 130.98247201809278 | 280.1297... | 122.4770321012746 | 288.6352378817254 |
| 23 | GROUP_1 | 2 | 200 | ? | ? | ? | ? |
| 24 | GROUP_1 | 3 | 193.5 | ? | ? | ? | ? |
| 25 | GROUP_1 | 4 | 193.65 | ? | ? | ? | ? |
| 26 | GROUP_1 | 5 | 194.035 | ? | ? | ? | ? |
| 27 | GROUP_1 | 6 | 205.6315 | ? | ? | ? | ? |
| 28 | GROUP_1 | 7 | 202.56834999999998 | ? | ? | ? | ? |
| 29 | GROUP_1 | 8 | 197.81151499999999 | ? | ? | ? | ? |
| 30 | GROUP_1 | 9 | 191.0303635 | ? | ? | ? | ? |
| 31 | GROUP_1 | 10 | 193.92732715 | ? | ? | ? | ? |
| 32 | GROUP_1 | 11 | 202.284594435 | ? | ? | ? | ? |
| 33 | GROUP_1 | 12 | 205.55613499150002 | 126.700940203298 | 284.4113... | 84.9575216121871 | 326.15474837081297 |
| 34 | GROUP_1 | 13 | 205.55613499150002 | 126.30764502143097 | 284.8046... | 84.35602853732595 | 326.75624144567405 |
| 35 | GROUP_1 | 14 | 205.55613499150002 | 125.91629207360504 | 285.1959... | 83.75750585310104 | 327.35476412989897 |
| 36 | GROUP_1 | 15 | 205.55613499150002 | 125.52685286653129 | 285.5854... | 83.16190998278736 | 327.95036000021264 |
| 37 | GROUP_1 | 16 | 205.55613499150002 | 125.1392995968596 | 285.9729... | 82.56919840483033 | 328.5430715781697 |
| 38 | GROUP_1 | 17 | 205.55613499150002 | 124.75360512801403 | 286.3586... | 81.97932961741841 | 329.1329403655816 |
| 39 | GROUP_1 | 18 | 205.55613499150002 | 124.36974296801866 | 286.7425... | 81.39226310457079 | 329.7200068784292 |
| 40 | GROUP_1 | 19 | 205.55613499150002 | 123.98768724826262 | 287.1245... | 80.80795930366119 | 330.30431067933887 |
| 41 | GROUP_1 | 20 | 205.55613499150002 | 123.60741270315617 | 287.5048... | 80.22637957430457 | 330.88589040869545 |
| 42 | GROUP_1 | 21 | 205.55613499150002 | 123.22889465063255 | 287.8833... | 79.64748616853727 | 331.46478381446275 |
| 43 | GROUP_1 | 22 | 205.55613499150002 | 122.85210897345293 | 288.2601... | 79.07124220222539 | 332.04102778077464 |
| 44 | GROUP_1 | 23 | 205.55613499150002 | 122.4770321012746 | 288.6352... | 78.49761162764055 | 332.6146583553595 |

| | GROUP_ID | STAT_NAME | STAT_VALUE |
|---|---|---|---|
| 1 | GROUP_2 | MSE | 3,438.3321253085405 |
| 2 | GROUP_1 | MSE | 3,438.3321253085405 |

## 3.5 Calling PAL Procedures in Parallel with Operator MAP_MERGE

The MAP_MERGE operator is used to apply each row of the input table to the mapper procedure and unite all intermediate result tables. The purpose of the operator is to replace sequential FOR-loops and union patterns with a parallel operator.

You can run parallel execution of selected PAL procedures from SAP HANA SQLScript. A group identifier should be assigned in advance to input data. The column containing group identifier is specified so that the input data can be separated and PAL procedures are executed on those groups in parallel.

**Example 1**

```
SET SCHEMA DM_PAL;
DROP TYPE PAL_SINGLESMOOTH_DATA_T;
CREATE TYPE PAL_SINGLESMOOTH_DATA_T AS TABLE(
    "GROUP_ID" VARCHAR (100),
    "ID" INT,
    "RAWDATA" DOUBLE
);
DROP TYPE PAL_PARAMETER_T;
CREATE TYPE PAL_PARAMETER_T AS TABLE (
    "GROUP_ID" VARCHAR (100),
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
DROP TYPE PAL_SINGLE_RESULT_T;
CREATE TYPE PAL_SINGLE_RESULT_T AS TABLE (
    "TIMESTAMP" INT,
    "VALUE" DOUBLE,
    "PI1_LOWER" DOUBLE,
    "PI1_UPPER" DOUBLE,
    "PI2_LOWER" DOUBLE,
    "PI2_UPPER" DOUBLE
);
DROP TYPE PAL_MULTIPLE_RESULT_T;
CREATE TYPE PAL_MULTIPLE_RESULT_T AS TABLE (
    "GROUP_ID" VARCHAR(100),
    "TIMESTAMP" INT,
    "VALUE" DOUBLE,
    "PI1_LOWER" DOUBLE,
    "PI1_UPPER" DOUBLE,
    "PI2_LOWER" DOUBLE,
    "PI2_UPPER" DOUBLE
);
DROP FUNCTION SES_SINGLE_THREAD;
CREATE FUNCTION SES_SINGLE_THREAD(
    IN i_group_id VARCHAR(100),
    IN it_data    PAL_SINGLESMOOTH_DATA_T,
    IN it_param   PAL_PARAMETER_T
) RETURNS PAL_MULTIPLE_RESULT_T AS
```

```
BEGIN
DECLARE lt_result PAL_SINGLE_RESULT_T;
lt_data = SELECT ID, RAWDATA FROM :it_data WHERE group_id = :i_group_id;
lt_param = SELECT PARAM_NAME, INT_VALUE, DOUBLE_VALUE, STRING_VALUE
FROM :it_param WHERE group_id = :i_group_id;
CALL "_SYS_AFL"."PAL_SINGLE_EXPSMOOTH"(:lt_data, :lt_param, lt_result, lt_stat);
RETURN SELECT i_group_id AS GROUP_ID, * FROM :lt_result;
END;
DROP PROCEDURE SES_PARALLEL_WRAPPER;
CREATE PROCEDURE SES_PARALLEL_WRAPPER(
    IN it_data    PAL_SINGLESMOOTH_DATA_T,
    IN it_param   PAL_PARAMETER_T,
    OUT et_result PAL_MULTIPLE_RESULT_T
) AS
BEGIN
  lt_groups = SELECT DISTINCT group_id AS group_id FROM :it_data;
  et_result = MAP_MERGE(:lt_groups,
SES_SINGLE_THREAD(:lt_groups.group_id, :it_data, :it_param));
END;
DROP TABLE PAL_SINGLESMOOTH_DATA_TBL;
CREATE COLUMN TABLE PAL_SINGLESMOOTH_DATA_TBL LIKE PAL_SINGLESMOOTH_DATA_T;
INSERT INTO PAL_SINGLESMOOTH_DATA_TBL VALUES ('A', 1,  200.0);
INSERT INTO PAL_SINGLESMOOTH_DATA_TBL VALUES ('A', 2,  135.0);
INSERT INTO PAL_SINGLESMOOTH_DATA_TBL VALUES ('A', 3,  195.0);
INSERT INTO PAL_SINGLESMOOTH_DATA_TBL VALUES ('A', 4,  197.5);
INSERT INTO PAL_SINGLESMOOTH_DATA_TBL VALUES ('A', 5,  310.0);
INSERT INTO PAL_SINGLESMOOTH_DATA_TBL VALUES ('A', 6,  175.0);
INSERT INTO PAL_SINGLESMOOTH_DATA_TBL VALUES ('A', 7,  155.0);
INSERT INTO PAL_SINGLESMOOTH_DATA_TBL VALUES ('A', 8,  130.0);
INSERT INTO PAL_SINGLESMOOTH_DATA_TBL VALUES ('A', 9,  220.0);
INSERT INTO PAL_SINGLESMOOTH_DATA_TBL VALUES ('A', 10, 277.5);
INSERT INTO PAL_SINGLESMOOTH_DATA_TBL VALUES ('A', 11, 235.0);
INSERT INTO PAL_SINGLESMOOTH_DATA_TBL VALUES ('B', 1,  200.0);
INSERT INTO PAL_SINGLESMOOTH_DATA_TBL VALUES ('B', 2,  135.0);
INSERT INTO PAL_SINGLESMOOTH_DATA_TBL VALUES ('B', 3,  195.0);
INSERT INTO PAL_SINGLESMOOTH_DATA_TBL VALUES ('B', 4,  197.5);
INSERT INTO PAL_SINGLESMOOTH_DATA_TBL VALUES ('B', 5,  310.0);
INSERT INTO PAL_SINGLESMOOTH_DATA_TBL VALUES ('B', 6,  175.0);
INSERT INTO PAL_SINGLESMOOTH_DATA_TBL VALUES ('B', 7,  155.0);
INSERT INTO PAL_SINGLESMOOTH_DATA_TBL VALUES ('B', 8,  130.0);
INSERT INTO PAL_SINGLESMOOTH_DATA_TBL VALUES ('B', 9,  220.0);
INSERT INTO PAL_SINGLESMOOTH_DATA_TBL VALUES ('B', 10, 277.5);
INSERT INTO PAL_SINGLESMOOTH_DATA_TBL VALUES ('B', 11, 235.0);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL LIKE PAL_PARAMETER_T;
INSERT INTO #PAL_PARAMETER_TBL VALUES ('A', 'ADAPTIVE_METHOD',            0,
NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('A', 'MEASURE_NAME',            NULL,
NULL, 'MSE');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('A', 'ALPHA',                   NULL,
0.1, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('A', 'DELTA',                   NULL,
0.2, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('A', 'FORECAST_NUM',               6,
NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('A', 'EXPOST_FLAG',                1,
NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('A', 'PREDICTION_CONFIDENCE_1', NULL,
0.8, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('A', 'PREDICTION_CONFIDENCE_2', NULL,
0.95, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('B', 'ADAPTIVE_METHOD',            0,
NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('B', 'MEASURE_NAME',            NULL,
NULL, 'MSE');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('B', 'ALPHA',                   NULL,
0.1, NULL);
```

```
INSERT INTO #PAL_PARAMETER_TBL VALUES ('B', 'DELTA',                    NULL,
0.2, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('B', 'FORECAST_NUM',                6,
NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('B', 'EXPOST_FLAG',                 0,
NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('B', 'PREDICTION_CONFIDENCE_1', NULL,
0.8, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('B', 'PREDICTION_CONFIDENCE_2', NULL,
0.95, NULL);
CALL SES_PARALLEL_WRAPPER(PAL_SINGLESMOOTH_DATA_TBL, #PAL_PARAMETER_TBL, ?);
```

**Expected Result**

| | GROUP_ID | TIMESTAMP | VALUE | PI1_LOWER | PI1_UPPER | PI2_LOWER | PI2_UPPER |
|---|---|---|---|---|---|---|---|
| 1 | A | 2 | 200 | ? | ? | ? | ? |
| 2 | A | 3 | 193.5 | ? | ? | ? | ? |
| 3 | A | 4 | 193.65 | ? | ? | ? | ? |
| 4 | A | 5 | 194.0... | ? | ? | ? | ? |
| 5 | A | 6 | 205.6... | ? | ? | ? | ? |
| 6 | A | 7 | 202.5... | ? | ? | ? | ? |
| 7 | A | 8 | 197.8... | ? | ? | ? | ? |
| 8 | A | 9 | 191.0... | ? | ? | ? | ? |
| 9 | A | 10 | 193.9... | ? | ? | ? | ? |
| 10 | A | 11 | 202.2... | ? | ? | ? | ? |
| 11 | A | 12 | 205.5... | 126.70094... | 284.4113... | 84.957521... | 326.1547... |
| 12 | A | 13 | 205.5... | 126.30764... | 284.8046... | 84.356028... | 326.7562... |
| 13 | A | 14 | 205.5... | 125.91629... | 285.1959... | 83.757505... | 327.3547... |
| 14 | A | 15 | 205.5... | 125.52685... | 285.5854... | 83.161909... | 327.9503... |
| 15 | A | 16 | 205.5... | 125.13929... | 285.9729... | 82.569198... | 328.5430... |
| 16 | A | 17 | 205.5... | 124.75360... | 286.3586... | 81.979329... | 329.1329... |
| 17 | B | 12 | 205.5... | 126.70094... | 284.4113... | 84.957521... | 326.1547... |
| 18 | B | 13 | 205.5... | 126.30764... | 284.8046... | 84.356028... | 326.7562... |
| 19 | B | 14 | 205.5... | 125.91629... | 285.1959... | 83.757505... | 327.3547... |
| 20 | B | 15 | 205.5... | 125.52685... | 285.5854... | 83.161909... | 327.9503... |
| 21 | B | 16 | 205.5... | 125.13929... | 285.9729... | 82.569198... | 328.5430... |
| 22 | B | 17 | 205.5... | 124.75360... | 286.3586... | 81.979329... | 329.1329... |

**Example 2**

It is also possible to use MAP_MERGE operator for PAL procedures with dynamic output structure. Below is an example.

```
SET SCHEMA DM_PAL;
DROP TYPE PAL_KMEANS_DATA_T;
CREATE TYPE PAL_KMEANS_DATA_T AS TABLE (
    "GROUP_ID" VARCHAR (100),
    "ID" INTEGER,
    "V000" DOUBLE,
    "V001" VARCHAR (2),
    "V002" DOUBLE
);
DROP TYPE PAL_PARAMETER_T;
CREATE TYPE PAL_PARAMETER_T AS TABLE (
```

```
    "GROUP_ID" VARCHAR (100),
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
DROP TYPE PAL_KMEANS_CENTERS_SINGLEGROUP_T;
CREATE TYPE PAL_KMEANS_CENTERS_SINGLEGROUP_T AS TABLE(
    "CLUSTER_ID" INTEGER,
    "V000" DOUBLE,
    "V001" VARCHAR(2),
    "V002" DOUBLE
);
DROP TYPE PAL_KMEANS_CENTERS_MULTIGROUP_T;
CREATE TYPE PAL_KMEANS_CENTERS_MULTIGROUP_T AS TABLE(
    "GROUP_ID" VARCHAR (100),
    "CLUSTER_ID" INTEGER,
    "V000" DOUBLE,
    "V001" VARCHAR(2),
    "V002" DOUBLE
);
DROP FUNCTION KMEANS_SINGLE_THREAD;
CREATE FUNCTION KMEANS_SINGLE_THREAD(
    IN i_group_id VARCHAR(100),
    IN it_data    PAL_KMEANS_DATA_T,
    IN it_param   PAL_PARAMETER_T
) RETURNS PAL_KMEANS_CENTERS_MULTIGROUP_T AS
BEGIN
DECLARE lt_centers TABLE ("CLUSTER_ID" INTEGER, "V000" DOUBLE, "V001"
VARCHAR(2), "V002" DOUBLE);
lt_data = SELECT ID, V000, V001, V002 FROM :it_data WHERE group_id = :i_group_id;
lt_param = SELECT PARAM_NAME, INT_VALUE, DOUBLE_VALUE, STRING_VALUE
FROM :it_param WHERE group_id = :i_group_id;
CALL "_SYS_AFL"."PAL_KMEANS"(:lt_data, :lt_param, lt_result, lt_centers,
lt_model, lt_stat, lt_selfparams);
RETURN SELECT i_group_id AS GROUP_ID, * FROM :lt_centers;
END;
DROP PROCEDURE KMEANS_PARALLEL_WRAPPER;
CREATE PROCEDURE KMEANS_PARALLEL_WRAPPER(
    IN it_data    PAL_KMEANS_DATA_T,
    IN it_param   PAL_PARAMETER_T,
    OUT et_result PAL_KMEANS_CENTERS_MULTIGROUP_T
) AS
BEGIN
  lt_groups = SELECT DISTINCT group_id AS group_id FROM :it_data;
  et_result = MAP_MERGE(:lt_groups,
KMEANS_SINGLE_THREAD(:lt_groups.group_id, :it_data, :it_param));
END;
DROP TABLE PAL_KMEANS_DATA_TBL;
CREATE COLUMN TABLE PAL_KMEANS_DATA_TBL LIKE PAL_KMEANS_DATA_T;
INSERT INTO PAL_KMEANS_DATA_TBL VALUES ('GROUP1',  0,  0.5, 'A',  0.5);
INSERT INTO PAL_KMEANS_DATA_TBL VALUES ('GROUP1',  1,  1.5, 'A',  0.5);
INSERT INTO PAL_KMEANS_DATA_TBL VALUES ('GROUP1',  2,  1.5, 'A',  1.5);
INSERT INTO PAL_KMEANS_DATA_TBL VALUES ('GROUP1',  3,  0.5, 'A',  1.5);
INSERT INTO PAL_KMEANS_DATA_TBL VALUES ('GROUP1',  4,  1.1, 'B',  1.2);
INSERT INTO PAL_KMEANS_DATA_TBL VALUES ('GROUP1',  5,  0.5, 'B', 15.5);
INSERT INTO PAL_KMEANS_DATA_TBL VALUES ('GROUP1',  6,  1.5, 'B', 15.5);
INSERT INTO PAL_KMEANS_DATA_TBL VALUES ('GROUP1',  7,  1.5, 'B', 16.5);
INSERT INTO PAL_KMEANS_DATA_TBL VALUES ('GROUP1',  8,  0.5, 'B', 16.5);
INSERT INTO PAL_KMEANS_DATA_TBL VALUES ('GROUP1',  9,  1.2, 'C', 16.1);
INSERT INTO PAL_KMEANS_DATA_TBL VALUES ('GROUP1', 10, 15.5, 'C', 15.5);
INSERT INTO PAL_KMEANS_DATA_TBL VALUES ('GROUP1', 11, 16.5, 'C', 15.5);
INSERT INTO PAL_KMEANS_DATA_TBL VALUES ('GROUP1', 12, 16.5, 'C', 16.5);
INSERT INTO PAL_KMEANS_DATA_TBL VALUES ('GROUP1', 13, 15.5, 'C', 16.5);
INSERT INTO PAL_KMEANS_DATA_TBL VALUES ('GROUP1', 14, 15.6, 'D', 16.2);
INSERT INTO PAL_KMEANS_DATA_TBL VALUES ('GROUP1', 15, 15.5, 'D',  0.5);
INSERT INTO PAL_KMEANS_DATA_TBL VALUES ('GROUP1', 16, 16.5, 'D',  0.5);
INSERT INTO PAL_KMEANS_DATA_TBL VALUES ('GROUP1', 17, 16.5, 'D',  1.5);
```

```
INSERT INTO PAL_KMEANS_DATA_TBL VALUES ('GROUP1', 18, 15.5, 'D',  1.5);
INSERT INTO PAL_KMEANS_DATA_TBL VALUES ('GROUP1', 19, 15.7, 'A',  1.6);
INSERT INTO PAL_KMEANS_DATA_TBL VALUES ('GROUP2',  0,  0.5, 'A',  0.5);
INSERT INTO PAL_KMEANS_DATA_TBL VALUES ('GROUP2',  1,  1.5, 'A',  0.5);
INSERT INTO PAL_KMEANS_DATA_TBL VALUES ('GROUP2',  2,  1.5, 'A',  1.5);
INSERT INTO PAL_KMEANS_DATA_TBL VALUES ('GROUP2',  3,  0.5, 'A',  1.5);
INSERT INTO PAL_KMEANS_DATA_TBL VALUES ('GROUP2',  4,  1.1, 'B',  1.2);
INSERT INTO PAL_KMEANS_DATA_TBL VALUES ('GROUP2',  5,  0.5, 'B', 15.5);
INSERT INTO PAL_KMEANS_DATA_TBL VALUES ('GROUP2',  6,  1.5, 'B', 15.5);
INSERT INTO PAL_KMEANS_DATA_TBL VALUES ('GROUP2',  7,  1.5, 'B', 16.5);
INSERT INTO PAL_KMEANS_DATA_TBL VALUES ('GROUP2',  8,  0.5, 'B', 16.5);
INSERT INTO PAL_KMEANS_DATA_TBL VALUES ('GROUP2',  9,  1.2, 'C', 16.1);
INSERT INTO PAL_KMEANS_DATA_TBL VALUES ('GROUP2', 10, 15.5, 'C', 15.5);
INSERT INTO PAL_KMEANS_DATA_TBL VALUES ('GROUP2', 11, 16.5, 'C', 15.5);
INSERT INTO PAL_KMEANS_DATA_TBL VALUES ('GROUP2', 12, 16.5, 'C', 16.5);
INSERT INTO PAL_KMEANS_DATA_TBL VALUES ('GROUP2', 13, 15.5, 'C', 16.5);
INSERT INTO PAL_KMEANS_DATA_TBL VALUES ('GROUP2', 14, 15.6, 'D', 16.2);
INSERT INTO PAL_KMEANS_DATA_TBL VALUES ('GROUP2', 15, 15.5, 'D',  0.5);
INSERT INTO PAL_KMEANS_DATA_TBL VALUES ('GROUP2', 16, 16.5, 'D',  0.5);
INSERT INTO PAL_KMEANS_DATA_TBL VALUES ('GROUP2', 17, 16.5, 'D',  1.5);
INSERT INTO PAL_KMEANS_DATA_TBL VALUES ('GROUP2', 18, 15.5, 'D',  1.5);
INSERT INTO PAL_KMEANS_DATA_TBL VALUES ('GROUP2', 19, 15.7, 'A',  1.6);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL LIKE PAL_PARAMETER_T;
INSERT INTO #PAL_PARAMETER_TBL VALUES ('GROUP1', 'GROUP_NUMBER', 4, null, null);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('GROUP1', 'INIT_TYPE', 1, null, null);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('GROUP1', 'DISTANCE_LEVEL',2, null, null);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('GROUP1', 'MAX_ITERATION', 100, null,
null);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('GROUP1', 'EXIT_THRESHOLD', null, 1.0E-6,
null);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('GROUP1', 'CATEGORY_WEIGHTS', null, 0.5,
null);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('GROUP2', 'GROUP_NUMBER', 3, null, null);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('GROUP2', 'INIT_TYPE', 1, null, null);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('GROUP2', 'DISTANCE_LEVEL',2, null, null);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('GROUP2', 'MAX_ITERATION', 100, null,
null);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('GROUP2', 'EXIT_THRESHOLD', null, 1.0E-6,
null);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('GROUP2', 'CATEGORY_WEIGHTS', null, 0.5,
null);
CALL KMEANS_PARALLEL_WRAPPER(PAL_KMEANS_DATA_TBL, #PAL_PARAMETER_TBL, ?);
```

### Expected Result

| | GROUP_ID | CLUSTER_ID | V000 | V001 | V002 |
|---|---|---|---|---|---|
| 1 | GROUP1 | 0 | 1.02 | A | 1.04 |
| 2 | GROUP1 | 1 | 15.940000000000001 | D | 1.1199999999999999 |
| 3 | GROUP1 | 2 | 15.919999999999998 | C | 16.04 |
| 4 | GROUP1 | 3 | 1.04 | B | 16.02 |
| 5 | GROUP2 | 0 | 1.02 | A | 1.04 |
| 6 | GROUP2 | 1 | 15.940000000000001 | D | 1.1199999999999999 |
| 7 | GROUP2 | 2 | 8.48 | C | 16.029999999999998 |

## 3.6 State Enabled Real-Time Scoring Functions

Scoring functions are composed of two steps: parsing a model and applying the model on data.

In PAL, these two steps are performed in a single function. It is also possible to run them separately. In this way, the model parsing (de-serializing content from database table, or converting text-format content into model object) is executed only once, and then the model is kept and repeatedly applied to data. This is especially beneficial for the scenarios where the model is complex and the same model is applied by scoring functions multiple times. In such scenarios, parsing the model takes significant amount of time compared with the whole execution time of scoring functions. It is wise to avoid parsing the model over and over again.

Splitting parsing models and executing scoring can be achieved by a family of PAL functions called stated enabled functions. As the name suggests, after being parsed, the model is kept in a container called state. The life cycle of a PAL function state is longer than that of a PAL scoring function. There are four types of operations on the state:

- Creation of state: Trained models are read from database tables and fed to the _SYS_AFL.PAL_CREATE_PAL_MODEL_STATE function. The function generates states which contain the parsed models and returns the identifiers of the states.
- List of states: This can be done by SQL statement:
  `SELECT * FROM SYS.M_AFL_STATES;`
- Scoring with state: Together with the data to be scored on, the state is given to the corresponding PAL predict procedure as parameters in the parameter table.
- Clearing of state: State, as a container of a parsed model, consumes memory. It is necessary to clear them when they are obsolete. This is done by the _SYS_AFL.PAL_DELETE_MODEL_STATE function.

State enabled functions include:

- Support Vector Machine
- Random Decision Trees
- Decision Trees
- Cluster Assignment
- Latent Dirichlet Allocation
- Binning Assignment
- Naive Bayes
- Principal Component Analysis (PCA)
- Multilayer Perceptron
- Factorized Polynomial Regression Models
- Multiple Linear Regression
- Logistic Regression (with Elastic Net Regularization)
- Alternating Least Squares
- KNN
- Conditional Random Field
- Hybrid Gradient Boosting Tree
- Categorical Principal Component Analysis
- Unified Regression
  - Support Vector Machine

    - Multilayer Perceptron
    - Decision Trees
    - Random Decision Trees
    - Hybrid Gradient Boosting Tree
    - Multiple Linear Regression
- Unified Classification
    - Support Vector Machine
    - Logistic Regression (with Elastic Net Regularization)
    - Multilayer Perceptron
    - Decision Trees
    - Random Decision Trees
    - Hybrid Gradient Boosting Tree
    - Naive Bayes

See the corresponding topics (Compatibility, Local Interpretability...) for details about the feature of the state of the unified classification & regression.

Note that the model created from the unified classification & regression is not compatible with the original prediction method.

## _SYS_AFL.PAL_CREATE_MODEL_STATE

This procedure creates a state for a trained model.

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<Model Table 1>` | IN |
| 2 | `<Model Table 2>` | IN |
| 3 | `<Model Table 3>` | IN |
| 4 | `<Model Table 4>` | IN |
| 5 | `<Model Table 5>` | IN |
| 6 | `<PARAMETER table>` | IN |
| 7 | `<State Identifier OUTPUT table>` | OUT |

### Input Table

| Table | Column | Data Type | Description |
|---|---|---|---|
| Model Table 1 | All columns | INTEGER, DOUBLE, VARCHAR, or NVARCHAR | Content of the model. The format varies among different algorithms. Refer to the documentation of the individual algorithm. |
| Model Table 2 | All columns | INTEGER, DOUBLE, VARCHAR, or NVARCHAR | Content of the model. The format varies among different algorithms. Refer to the documentation of the individual algorithm. |
| Model Table 3 | All columns | INTEGER, DOUBLE, VARCHAR, or NVARCHAR | Content of the model. The format varies among different algorithms. Refer to the documentation of the individual algorithm. |
| Model Table 4 | All columns | INTEGER, DOUBLE, VARCHAR, or NVARCHAR | Content of the model. The format varies among different algorithms. Refer to the documentation of the individual algorithm. |
| Model Table 5 | All columns | INTEGER, DOUBLE, VARCHAR, or NVARCHAR | Content of the model. The format varies among different algorithms. Refer to the documentation of the individual algorithm. |

### Parameter Table

Mandatory Parameters

The following parameters are mandatory and must be given a value.

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| ALGORITHM | INTEGER | 1 | • 1: Support Vector Machine<br>• 2: Random Decision Trees<br>• 3: Decision Trees<br>• 4: Cluster Assignment<br>• 5: Latent Dirichlet Allocation<br>• 6: Binning Assignment<br>• 7: Naive Bayes<br>• 8: Principal Component Analysis (PCA)<br>• 9: Multilayer Perceptron<br>• 10: KNN<br>• 11: Factorized Polynomial Regression Models<br>• 16: Multiple Linear Regression<br>• 20: Logistic Regression (with Elastic Net Regularization)<br>• 24: Alternating Least Squares<br>• 26: Conditional Random Field<br>• 27: Hybrid Gradient Boosting Tree<br>• 28: Categorical Principal Component Analysis<br>• 30: Unified Regression<br>• 31: Unified Classification |

For some algorithms, extra mandatory parameters are required when creating the state model. For unpivoted unified regression and unified classification, the `FUNCTION` parameter is mandatory.

Optional Parameters

The following parameter is optional. If it is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| STATE_DESCRIPTION | VARCHAR or NVARCHAR | State of PAL model | Description of the state as model container. |

### Output Table

| Table | Column | Data Type | Description |
|---|---|---|---|
| State | 1st column | VARCHAR or NVARCHAR | State identifier key |
| | 2nd column | VARCHAR or NVARCHAR | State identifier value |

> **Note**
>
> The output State table of _SYS_AFL.PAL_CREATE_MODEL_STATE contains keys STATE_ID, HINT, HOST, and PORT. STATE_ID is the identifier of the AFL State. HOST and PORT indicate the host and the port of the script server process. HINT points to the volume ID of the index server where the procedure _SYS_AFL.PAL_CREATE_MODEL_STATE was started.
>
> For a landscape with a single index server, HINT is not mandatory for the subsequent call because the subsequent call is automatically routed to this index server. For a landscape with multiple index servers, use the syntax `WITH HINT(ROUTE_TO(<hint_value>))`. One example is CALL `_SYS_AFL.PAL_DELETE_MODEL_STATE(…)WITH HINT(ROUTE_TO(<hint_value>))`. In case that multiple script servers exist, there's uncertainty in finding the state. For details of the hint-related syntax, see the topic ROUTE_TO Hint in the SAP HANA SQL Reference Guide for SAP HANA Platform.

### Examples

#### Example 1: Create state for random decision trees

```
SET SCHEMA DM_PAL;
DROP TABLE  PAL_RDT_DATA_TBL;
CREATE COLUMN TABLE PAL_RDT_DATA_TBL (
    "OUTLOOK" VARCHAR(20),
    "TEMP" INTEGER,
    "HUMIDITY" DOUBLE,
    "WINDY" VARCHAR(10),
    "CLASS" VARCHAR(20)
);
INSERT INTO PAL_RDT_DATA_TBL VALUES ('Sunny', 75, 70.0, 'Yes', 'Play');
INSERT INTO PAL_RDT_DATA_TBL VALUES ('Sunny', NULL, 90.0, 'Yes', 'Do not Play');
INSERT INTO PAL_RDT_DATA_TBL VALUES ('Sunny', 85, NULL, 'No', 'Do not Play');
INSERT INTO PAL_RDT_DATA_TBL VALUES ('Sunny', 72, 95.0, 'No', 'Do not Play');
INSERT INTO PAL_RDT_DATA_TBL VALUES (NULL, NULL, 70.0, NULL, 'Play');
INSERT INTO PAL_RDT_DATA_TBL VALUES ('Overcast', 72.0, 90, 'Yes', 'Play');
INSERT INTO PAL_RDT_DATA_TBL VALUES ('Overcast', 83.0, 78, 'No', 'Play');
INSERT INTO PAL_RDT_DATA_TBL VALUES ('Overcast', 64.0, 65, 'Yes', 'Play');
INSERT INTO PAL_RDT_DATA_TBL VALUES ('Overcast', 81.0, 75, 'No', 'Play');
INSERT INTO PAL_RDT_DATA_TBL VALUES (NULL, 71, 80.0, 'Yes', 'Do not Play');
INSERT INTO PAL_RDT_DATA_TBL VALUES ('Rain', 65, 70.0, 'Yes', 'Do not Play');
INSERT INTO PAL_RDT_DATA_TBL VALUES ('Rain', 75, 80.0, 'No', 'Play');
INSERT INTO PAL_RDT_DATA_TBL VALUES ('Rain', 68, 80.0, 'No', 'Play');
INSERT INTO PAL_RDT_DATA_TBL VALUES ('Rain', 70, 96.0, 'No', 'Play');
DROP TABLE PAL_RDT_TRAINING_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_RDT_TRAINING_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (100)
);
INSERT INTO PAL_RDT_TRAINING_PARAMETER_TBL VALUES ('TREES_NUM', 300, NULL, NULL);
INSERT INTO PAL_RDT_TRAINING_PARAMETER_TBL VALUES ('TRY_NUM', 3, NULL, NULL);
INSERT INTO PAL_RDT_TRAINING_PARAMETER_TBL VALUES ('SEED', 2, NULL, NULL);
INSERT INTO PAL_RDT_TRAINING_PARAMETER_TBL VALUES ('SPLIT_THRESHOLD', NULL,
1e-5, NULL);
```

```
INSERT INTO PAL_RDT_TRAINING_PARAMETER_TBL VALUES ('CALCULATE_OOB', 1, NULL,
NULL);
INSERT INTO PAL_RDT_TRAINING_PARAMETER_TBL VALUES ('NODE_SIZE', 1, NULL, NULL);
INSERT INTO PAL_RDT_TRAINING_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 1.0,
NULL);
DROP TABLE PAL_RDT_MODEL_TBL;
CREATE COLUMN TABLE PAL_RDT_MODEL_TBL (
    "ROW_INDEX" INTEGER,
    "TREE_INDEX" INTEGER,
    "MODEL_CONTENT" NVARCHAR(5000)
);
DO BEGIN
  lt_data = SELECT * FROM PAL_RDT_DATA_TBL;
  lt_param = SELECT * FROM PAL_RDT_TRAINING_PARAMETER_TBL;
  CALL _SYS_AFL.PAL_RANDOM_DECISION_TREES (:lt_data, :lt_param, lt_model,
lt_imp, lt_error, lt_cm);
  INSERT INTO PAL_RDT_MODEL_TBL SELECT * FROM :lt_model;
END;
/* create state */
DROP TABLE PAL_SET_STATE_PARAMETERS_TBL;
CREATE COLUMN TABLE PAL_SET_STATE_PARAMETERS_TBL(
    "PARAM_NAME" VARCHAR (100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (100)
);
INSERT INTO PAL_SET_STATE_PARAMETERS_TBL VALUES('ALGORITHM', 2, NULL, NULL);
INSERT INTO PAL_SET_STATE_PARAMETERS_TBL VALUES('STATE_DESCRIPTION', NULL, NULL,
'PAL Random Decision Tree State');
DROP TABLE PAL_EMPTY_TBL;
CREATE TABLE PAL_EMPTY_TBL(
    ID double
);
DROP TABLE PAL_STATE_TBL;
CREATE TABLE PAL_STATE_TBL (
    S_KEY VARCHAR(50),
    S_VALUE VARCHAR(100)
);
DO BEGIN
  lt_model = SELECT * FROM PAL_RDT_MODEL_TBL;
  lt_empty = SELECT * FROM PAL_EMPTY_TBL;
  lt_param = SELECT * FROM PAL_SET_STATE_PARAMETERS_TBL;
  CALL
_SYS_AFL.PAL_CREATE_MODEL_STATE(:lt_model, :lt_empty, :lt_empty, :lt_empty, :lt_e
mpty, :lt_param, lt_state);
  INSERT INTO PAL_STATE_TBL SELECT * FROM :lt_state;
END;
SELECT * FROM PAL_STATE_TBL;
```

### Expected Result

Values may be different.

| | S_KEY | S_VALUE |
|---|---|---|
| 1 | STATE_ID | 599C8144C697384D9F98BBB63A7F7870 |
| 2 | HINT | 3 |
| 3 | HOST | cnpvg50860233v1 |
| 4 | PORT | 31104 |

### Example 2: Create state for LDA inference (with more than one model table)

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_LATENT_DIRICHLET_ALLOCATION_DATA_TBL;
```

```
CREATE COLUMN TABLE PAL_LATENT_DIRICHLET_ALLOCATION_DATA_TBL ("DOCUMENT_ID"
INTEGER, "TEXT" NVARCHAR (5000));
INSERT INTO PAL_LATENT_DIRICHLET_ALLOCATION_DATA_TBL VALUES (10 , 'cpu harddisk
graphiccard cpu monitor keyboard cpu memory memory');
INSERT INTO PAL_LATENT_DIRICHLET_ALLOCATION_DATA_TBL VALUES (20 , 'tires
mountainbike wheels valve helmet mountainbike rearfender tires mountainbike
mountainbike');
INSERT INTO PAL_LATENT_DIRICHLET_ALLOCATION_DATA_TBL VALUES (30 , 'carseat toy
strollers toy toy spoon toy strollers toy carseat');
INSERT INTO PAL_LATENT_DIRICHLET_ALLOCATION_DATA_TBL VALUES (40 , 'sweaters
sweaters sweaters boots sweaters rings vest vest shoe sweaters');
DROP TABLE PAL_LDT_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_LDT_PARAMETER_TBL ("PARAM_NAME" VARCHAR(256),
"INT_VALUE" INTEGER, "DOUBLE_VALUE" DOUBLE, "STRING_VALUE" VARCHAR(1000));
INSERT INTO PAL_LDT_PARAMETER_TBL VALUES ('TOPICS', 6, NULL, NULL);
INSERT INTO PAL_LDT_PARAMETER_TBL VALUES ('BURNIN', 50, NULL, NULL);
INSERT INTO PAL_LDT_PARAMETER_TBL VALUES ('THIN', 10, NULL, NULL);
INSERT INTO PAL_LDT_PARAMETER_TBL VALUES ('ITERATION', 100, NULL, NULL);
INSERT INTO PAL_LDT_PARAMETER_TBL VALUES ('SEED', 1, NULL, NULL);
INSERT INTO PAL_LDT_PARAMETER_TBL VALUES ('ALPHA', NULL, 0.1, NULL);
INSERT INTO PAL_LDT_PARAMETER_TBL VALUES ('MAX_TOP_WORDS', 5, NULL, NULL);
INSERT INTO PAL_LDT_PARAMETER_TBL VALUES ('OUTPUT_WORD_ASSIGNMENT', 1, NULL,
NULL);
INSERT INTO PAL_LDT_PARAMETER_TBL VALUES ('DELIMIT', NULL, NULL, ' '||char(13)||
char(10));
DROP TABLE PAL_LDA_TOPIC_WORD_DISTRIBUTION_TBL;
DROP TABLE PAL_LDA_DICTIONARY_TBL;
DROP TABLE PAL_LDA_CV_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_LDA_TOPIC_WORD_DISTRIBUTION_TBL ("TOPIC_ID" INTEGER,
"WORD_ID" INTEGER, "PROBABILITY" DOUBLE);
CREATE COLUMN TABLE PAL_LDA_DICTIONARY_TBL ("WORD_ID" INTEGER, "WORD"
NVARCHAR(5000));
CREATE COLUMN TABLE PAL_LDA_CV_PARAMETER_TBL ("PARAM_NAME" VARCHAR(256),
"INT_VALUE" INTEGER, "DOUBLE_VALUE" DOUBLE, "STRING_VALUE" VARCHAR(1000));
DO BEGIN
  lt_data = SELECT * FROM PAL_LATENT_DIRICHLET_ALLOCATION_DATA_TBL;
  lt_param = SELECT * FROM PAL_LDT_PARAMETER_TBL;
  CALL _SYS_AFL.PAL_LATENT_DIRICHLET_ALLOCATION (:lt_data, :lt_param,
lt_doc_topic_dist, lt_word_topic_asgn, lt_topic_top_words, lt_topic_words_dist,
lt_dict, lt_stat, lt_cv_param);
  INSERT INTO PAL_LDA_TOPIC_WORD_DISTRIBUTION_TBL SELECT *
FROM :lt_topic_words_dist;
  INSERT INTO PAL_LDA_DICTIONARY_TBL SELECT * FROM :lt_dict;
  INSERT INTO PAL_LDA_CV_PARAMETER_TBL SELECT * FROM :lt_cv_param;
END;
DROP TABLE PAL_SET_STATE_PARAMETERS_TBL;
CREATE COLUMN TABLE PAL_SET_STATE_PARAMETERS_TBL(
    NAME VARCHAR(50),
    INTARGS INTEGER,
    DOUBLEARGS DOUBLE,
    STRINGARGS VARCHAR(100)
);
INSERT INTO PAL_SET_STATE_PARAMETERS_TBL VALUES('ALGORITHM', 5, NULL, NULL);
INSERT INTO PAL_SET_STATE_PARAMETERS_TBL VALUES('STATE_DESCRIPTION', NULL, NULL,
'PAL LDAInference');
DROP TABLE PAL_EMPTY_TBL;
CREATE TABLE PAL_EMPTY_TBL(
    ID double
);
DROP TABLE PAL_STATE_TBL;
CREATE TABLE PAL_STATE_TBL (
    S_KEY VARCHAR(50),
    S_VALUE VARCHAR(100)
);
DO BEGIN
  lt_topic_words_dist = SELECT * FROM PAL_LDA_TOPIC_WORD_DISTRIBUTION_TBL;
  lt_dict = SELECT * FROM PAL_LDA_DICTIONARY_TBL;
  lt_cv_param = SELECT * FROM PAL_LDA_CV_PARAMETER_TBL;
```

```
  lt_empty = SELECT * FROM PAL_EMPTY_TBL;
  lt_param = SELECT * FROM PAL_SET_STATE_PARAMETERS_TBL;
  CALL
_SYS_AFL.PAL_CREATE_MODEL_STATE(:lt_topic_words_dist, :lt_dict, :lt_cv_param, :lt
_empty, :lt_empty, :lt_param, lt_state);
  INSERT INTO PAL_STATE_TBL SELECT * FROM :lt_state;
END;
SELECT * FROM PAL_STATE_TBL;
```

**Expected Result**

Values may be different.

| | S_KEY | S_VALUE |
|---|---|---|
| 1 | STATE_ID | 11855BAEA0156D439FE81AD0311F39D5 |
| 2 | HINT | 3 |
| 3 | HOST | cnpvg50860233v1 |
| 4 | PORT | 31104 |

**Example 3: Create state for Unified regession**

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_DATA_TBL;
CREATE COLUMN TABLE PAL_DATA_TBL(
     "ID" INTEGER,
     "X1" DOUBLE,
     "X2" VARCHAR (100),
     "X3" INTEGER,
     "Y" DOUBLE
);
INSERT INTO PAL_DATA_TBL VALUES (0, 0.0, 'A', 1, -6.879);
INSERT INTO PAL_DATA_TBL VALUES (1, 0.50, 'A', 1, -3.449);
INSERT INTO PAL_DATA_TBL VALUES (2, 0.54, 'B', 1, 6.635);
INSERT INTO PAL_DATA_TBL VALUES (3, 1.04, 'B', 1, 11.844);
INSERT INTO PAL_DATA_TBL VALUES (4, 1.50, 'A', 1, 2.786);
INSERT INTO PAL_DATA_TBL VALUES (5, 0.04, 'B', 2, 2.389);
INSERT INTO PAL_DATA_TBL VALUES (6, 2.00, 'A', 2, -0.011);
INSERT INTO PAL_DATA_TBL VALUES (7, 2.04, 'B', 2, 8.839);
INSERT INTO PAL_DATA_TBL VALUES (8, 1.54, 'B', 1, 4.689);
INSERT INTO PAL_DATA_TBL VALUES (9, 1.00, 'A', 2, -5.507);
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO PAL_PARAMETER_TBL VALUES ('FUNCTION',null,null,'RDT');
--INSERT INTO PAL_PARAMETER_TBL VALUES ('PARTITION_METHOD',2,null,null);
--INSERT INTO PAL_PARAMETER_TBL VALUES ('PARTITION_TRAINING_PERCENT',
null,0.7,null);
--INSERT INTO PAL_PARAMETER_TBL VALUES ('OUTPUT_PARTITION_RESULT', 1,null, null);
INSERT INTO PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', null,0.5,null);
INSERT INTO PAL_PARAMETER_TBL VALUES ('LABEL', null,null,'X3');
INSERT INTO PAL_PARAMETER_TBL VALUES ('BACKGROUND_SIZE', 4,null,null);
INSERT INTO PAL_PARAMETER_TBL VALUES ('KEY', 1,null,null);
DROP TABLE PAL_DATA_MODEL_TBL;
CREATE COLUMN TABLE PAL_DATA_MODEL_TBL(
     "ROW_INDEX" INTEGER,
     "PART_INDEX" INTEGER,
     "MODEL_CONTENT" VARCHAR (5000)
);
DO BEGIN
  data_tab = SELECT * FROM PAL_DATA_TBL;
```

```
  control_tab = SELECT * FROM PAL_PARAMETER_TBL;
  CALL _SYS_AFL.PAL_UNIFIED_REGRESSION(:data_tab, :control_tab, model_tab,
stats_tab, optimal_parameter_tab, partition_result, place_holder_1_tab,
place_holder_2_tab);
  INSERT into PAL_DATA_MODEL_TBL SELECT * FROM :model_tab;
END;
/* create state */
DROP TABLE PAL_SET_STATE_PARAMETERS_TBL;
CREATE COLUMN TABLE PAL_SET_STATE_PARAMETERS_TBL(
    "PARAM_NAME" VARCHAR (100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (100)
);
INSERT INTO PAL_SET_STATE_PARAMETERS_TBL VALUES('ALGORITHM', 30, NULL, NULL);
INSERT INTO PAL_SET_STATE_PARAMETERS_TBL VALUES('STATE_DESCRIPTION', NULL, NULL,
'PAL Unified Regression Unpv State');
INSERT INTO PAL_SET_STATE_PARAMETERS_TBL VALUES('FUNCTION', NULL, NULL, 'RDT');
DROP TABLE PAL_EMPTY_TBL;
CREATE TABLE PAL_EMPTY_TBL(
    ID double
);
DROP TABLE PAL_STATE_TBL;
CREATE TABLE PAL_STATE_TBL (
    S_KEY VARCHAR(50),
    S_VALUE VARCHAR(100)
);
DO BEGIN
  lt_model = SELECT * FROM PAL_DATA_MODEL_TBL;
  lt_empty = SELECT * FROM PAL_EMPTY_TBL;
  lt_param = SELECT * FROM PAL_SET_STATE_PARAMETERS_TBL;
  CALL
_SYS_AFL.PAL_CREATE_MODEL_STATE(:lt_model, :lt_empty, :lt_empty, :lt_empty, :lt_e
mpty, :lt_param, lt_state);
  INSERT INTO PAL_STATE_TBL SELECT * FROM :lt_state;
END;
```

**Expected Result**

Values may be different.

Rows (4)

| | S_KEY | S_VALUE |
|---|---|---|
| 1 | STATE_ID | 565BAE57E5F09841A2CAFAD9798CB6D0 |
| 2 | HINT | 2 |
| 3 | HOST | pvgl34899110a |
| 4 | PORT | 30040 |

## Predict with State

This function executes scoring function with the model contained in a state. For the detailed usage, please refer to the documentation of each function. In the call to the prediction function, the model table should be empty.

If both model content and state ID are provided, the corresponding state is created in addition to the execution of prediction. In such case, if a state with that ID already exists, it is used for the prediction and no state creation is performed.

**Examples**

**Example 1: Predict with random decision trees model using state**

This example proceeds with the Create state for random decision trees example.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_RDT_SCORING_DATA_TBL;
CREATE COLUMN TABLE PAL_RDT_SCORING_DATA_TBL (
    "ID" INTEGER,
    "OUTLOOK" VARCHAR(20),
    "TEMP" INTEGER,
    "HUMIDITY" DOUBLE,
    "WINDY" VARCHAR(10)
);
INSERT INTO PAL_RDT_SCORING_DATA_TBL VALUES (0, 'Overcast', 75, -10000.0, 'Yes');
INSERT INTO PAL_RDT_SCORING_DATA_TBL VALUES (1, 'Rain', 78, 70.0, 'Yes');
INSERT INTO PAL_RDT_SCORING_DATA_TBL VALUES (2, 'Sunny', -10000.0, NULL, 'Yes');
INSERT INTO PAL_RDT_SCORING_DATA_TBL VALUES (3, 'Sunny', 69, 70.0, 'Yes');
INSERT INTO PAL_RDT_SCORING_DATA_TBL VALUES (4, 'Rain', NULL, 70.0, 'Yes');
INSERT INTO PAL_RDT_SCORING_DATA_TBL VALUES (5, NULL, 70, 70.0, 'Yes');
INSERT INTO PAL_RDT_SCORING_DATA_TBL VALUES (6, '***', 70, 70.0, 'Yes');
DROP TABLE  #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR(100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR(100)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 0.5, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('VERBOSE', 0, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL(PARAM_NAME, STRING_VALUE)
    SELECT S_KEY, S_VALUE from PAL_STATE_TBL;
DROP TABLE PAL_EMPTY_RDT_MODEL_TBL;  -- for predict followed
CREATE COLUMN TABLE PAL_EMPTY_RDT_MODEL_TBL (
    "ROW_INDEX" INTEGER,
    "TREE_INDEX" INTEGER,
    "MODEL_CONTENT" NVARCHAR(5000)
);
CALL _SYS_AFL.PAL_RANDOM_DECISION_TREES_PREDICT (PAL_RDT_SCORING_DATA_TBL,
PAL_EMPTY_RDT_MODEL_TBL, #PAL_PARAMETER_TBL, ?);
```

**Expected Result**

| | ID | SCORE | CONFIDENCE |
|---|---|---|---|
| 1 | 0 | Play | 0.96 |
| 2 | 1 | Play | 0.85 |
| 3 | 2 | Do not Play | 0.9033333333333333 |
| 4 | 3 | Do not Play | 0.61 |
| 5 | 4 | Play | 0.68 |
| 6 | 5 | Do not Play | 0.63 |
| 7 | 6 | Play | 0.8266666666666667 |

### Example 2: LDA inference using state

This example proceeds with the Create state for LDA inference (with more than one model table) example.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_LATENT_DIRICHLET_ALLOCATION_INFERENCE_DATA_TBL;
CREATE COLUMN TABLE PAL_LATENT_DIRICHLET_ALLOCATION_INFERENCE_DATA_TBL
("DOCUMENT_ID" INTEGER, "TEXT" NVARCHAR (5000));
INSERT INTO PAL_LATENT_DIRICHLET_ALLOCATION_INFERENCE_DATA_TBL VALUES (10, 'toy
toy spoon cpu');
DROP TABLE PAL_EMPTY_TOPIC_WORD_DISTRIB;
CREATE TABLE PAL_EMPTY_TOPIC_WORD_DISTRIB (TOPID_ID integer, WORD_ID integer,
PROB double);
```

```
DROP TABLE PAL_EMPTY_DICT;
CREATE TABLE PAL_EMPTY_DICT (WORD_ID integer, WORD_TEXT NVARCHAR(100));
DROP TABLE PAL_EMPTY_CV_PARM;
CREATE TABLE PAL_EMPTY_CV_PARM ("PARAM_NAME" VARCHAR(256), "INT_VALUE" INTEGER,
"DOUBLE_VALUE" DOUBLE, "STRING_VALUE" VARCHAR(1000));
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL ("PARAM_NAME"
VARCHAR(256), "INT_VALUE" INTEGER, "DOUBLE_VALUE" DOUBLE, "STRING_VALUE"
VARCHAR(1000));
INSERT INTO #PAL_PARAMETER_TBL VALUES ('BURNIN', 2000, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('THIN', 100, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('ITERATION', 1000, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SEED', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('OUTPUT_WORD_ASSIGNMENT', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL(PARAM_NAME, STRING_VALUE)
    SELECT S_KEY, S_VALUE from PAL_STATE_TBL;
CALL _SYS_AFL.PAL_LATENT_DIRICHLET_ALLOCATION_INFERENCE
(PAL_LATENT_DIRICHLET_ALLOCATION_INFERENCE_DATA_TBL,
PAL_EMPTY_TOPIC_WORD_DISTRIB, PAL_EMPTY_DICT, PAL_EMPTY_CV_PARM,
#PAL_PARAMETER_TBL, ?, ?, ?);
```

**Expected Result**

| | DOCUMENT_ID | TOPIC_ID | PROBABILITY |
|---|---|---|---|
| 1 | 10 | 0 | 0.23913043478260873 |
| 2 | 10 | 1 | 0.4565217391304348 |
| 3 | 10 | 2 | 0.02173913043478261 |
| 4 | 10 | 3 | 0.02173913043478261 |
| 5 | 10 | 4 | 0.23913043478260873 |
| 6 | 10 | 5 | 0.02173913043478261 |

| | DOCUMENT_ID | WORD_ID | TOPIC_ID |
|---|---|---|---|
| 1 | 10 | 13 | 1 |
| 2 | 10 | 13 | 1 |
| 3 | 10 | 15 | 0 |
| 4 | 10 | 0 | 4 |

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | DOCUMENTS | 1 |
| 2 | VOCABULARY_SIZE | 21 |
| 3 | LOG_LIKELIHOOD | -7.925092991875363 |
| 4 | PERPLEXITY | 7.251970666272191 |

**Example 3: Unified Regression**

The AFL state of unified interface (classification and regression) supports both prediction and scoring.

```
-------------------------------Prediction and
Scoring------------------------------
DROP TABLE PAL_DATA_TBL_SCORE;
CREATE COLUMN TABLE PAL_DATA_TBL_SCORE(
     "ID" INTEGER,
     "X1" DOUBLE,
     "X2" VARCHAR (100),
     "X3" INTEGER,
     "Y" DOUBLE
```

```
);
INSERT INTO PAL_DATA_TBL_SCORE VALUES (0, 1.69, 'B', 1, 1.2);
INSERT INTO PAL_DATA_TBL_SCORE VALUES (1, 0.054, 'B', 2, 2.1);
INSERT INTO PAL_DATA_TBL_SCORE VALUES (2, 0.123, 'A', 2, 2.4);
INSERT INTO PAL_DATA_TBL_SCORE VALUES (3, 1.980, 'A', 1, 1.8);
INSERT INTO PAL_DATA_TBL_SCORE VALUES (4, 0.563, 'A', 1, 1.0);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('LABEL', null,null,'X3');
/* insert state model*/
INSERT INTO #PAL_PARAMETER_TBL(PARAM_NAME, STRING_VALUE)
    SELECT S_KEY, S_VALUE from PAL_STATE_TBL;
DROP TABLE PAL_EMPTY_RDT_MODEL_TBL;  -- for predict followed
CREATE COLUMN TABLE PAL_EMPTY_RDT_MODEL_TBL (
    "ROW_INDEX" INTEGER,
    "PART_INDEX" INTEGER,
    "MODEL_CONTENT" NVARCHAR(5000)
);
CALL _SYS_AFL.PAL_UNIFIED_REGRESSION_SCORE(PAL_DATA_TBL_SCORE,
PAL_EMPTY_RDT_MODEL_TBL, #PAL_PARAMETER_TBL, ?,?);
```

### Expected Result

RESULT:

Rows (5)

| | ID | SCORE | UPPER_BOUND | LOWER_BOUND | REASON |
|---|---|---|---|---|---|
| 1 | 0 | 1.46295 | 0.440427001786986 | *NULL* | [{"attr":"Y","pct":56.0,"val":0.1372837451948853 1},{"attr":"X1","pct":39.0,"val":-0.09676931163315703 7},{"attr":"X2","pct":5.0,"val":0.011437766438271608}] |
| 2 | 1 | 1.54428 | 0.46676629016382076 | *NULL* | [{"attr":"Y","pct":58.0,"val":0.10337775844444444},{"attr":"X2","pct":29.0,"val":0.05301059066666669},{"attr":"X1","pct":13.0,"val":-0.023102749111111109}] |
| 3 | 2 | 1.34071 | 0.42885109939238913 | *NULL* | [{"attr":"X1","pct":54.0,"val":-0.05081966069841271 2},{"attr":"X2","pct":33.0,"val":-0.03156148568253966 4},{"attr":"Y","pct":13.0,"val":0.012095746380952415}] |
| 4 | 3 | 1.58571 | 0.45377136902745485 | *NULL* | [{"attr":"X1","pct":88.0,"val":0.19273722505291005},{"attr":"X2","pct":10.0,"val":-0.022345880939153427},{"attr":"Y","pct":2.0,"val":0.0043232558862434049}] |
| 5 | 4 | 1.39604 | 0.417969062280012 | *NULL* | [{"attr":"X1","pct":49.0,"val":-0.12966764707407405},{"attr":"Y","pct":47.0,"val":0.12461689512433857},{"attr":"X2","pct":4.0,"val":-0.009901448050264542 7}] |

STATISTICS:

Rows (8)

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | EVAR | -0.11235959289999986 |
| 2 | MAE | 0.511942 |
| 3 | MAPE | 0.400441 |
| 4 | MAX_ERROR | 0.6592899999999999 |
| 5 | MSE | 0.2713141221399999 |
| 6 | R2 | -0.13047550891666626 |
| 7 | RMSE | 0.520878221986675 |
| 8 | WMAPE | 0.36567285714285713 |

## _SYS_AFL.PAL_DELETE_MODEL_STATE

This function clears a state and releases the resources used by it.

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<INPUT State Identifier table>` | IN |

| Position | Table Name | Parameter Type |
|---|---|---|
| 2 | `<PARAMETER table>` | IN |
| 3 | `<Message OUTPUT table>` | OUT |

### Input Table

| Table | Column | Column Data Type | Description |
|---|---|---|---|
| State | 1st column | VARCHAR or NVARCHAR | State identifier key |
| | 2nd column | VARCHAR or NVARCHAR | State identifier value |

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

None.

Output Table

| Table | Column | Column Data Type | Description |
|---|---|---|---|
| Message | 1st column | VARCHAR or NVARCHAR | ID |
| | 2nd column | VARCHAR or NVARCHAR | Timestamp |
| | 3rd column | VARCHAR or NVARCHAR | Error code |
| | 4th column | VARCHAR or NVARCHAR | Message |

## End-to-End Examples

### Example 1: Support vector classification

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_SVM_DATA_TBL;
CREATE COLUMN TABLE PAL_SVM_DATA_TBL (
    ID INTEGER,
    ATTRIBUTE1 DOUBLE,
    ATTRIBUTE2 DOUBLE,
    ATTRIBUTE3 DOUBLE,
    ATTRIBUTE4 VARCHAR(10),
    LABEL INTEGER
);
INSERT INTO PAL_SVM_DATA_TBL VALUES (0,1,10,100,'A',1);
INSERT INTO PAL_SVM_DATA_TBL VALUES (1,1.1,10.1,100,'A',1);
INSERT INTO PAL_SVM_DATA_TBL VALUES (2,1.2,10.2,100,'A',1);
```

```
INSERT INTO PAL_SVM_DATA_TBL VALUES (3,1.3,10.4,100,'A',1);
INSERT INTO PAL_SVM_DATA_TBL VALUES (4,1.2,10.3,100,'AB',1);
INSERT INTO PAL_SVM_DATA_TBL VALUES (5,4,40,400,'AB',2);
INSERT INTO PAL_SVM_DATA_TBL VALUES (6,4.1,40.1,400,'AB',2);
INSERT INTO PAL_SVM_DATA_TBL VALUES (7,4.2,40.2,400,'AB',2);
INSERT INTO PAL_SVM_DATA_TBL VALUES (8,4.3,40.4,400,'AB',2);
INSERT INTO PAL_SVM_DATA_TBL VALUES (9,4.2,40.3,400,'AB',2);
INSERT INTO PAL_SVM_DATA_TBL VALUES (10,9,90,900,'B',3);
INSERT INTO PAL_SVM_DATA_TBL VALUES (11,9.1,90.1,900,'A',3);
INSERT INTO PAL_SVM_DATA_TBL VALUES (12,9.2,90.2,900,'B',3);
INSERT INTO PAL_SVM_DATA_TBL VALUES (13,9.3,90.4,900,'A',3);
INSERT INTO PAL_SVM_DATA_TBL VALUES (14,9.2,90.3,900,'A',3);
DROP TABLE PAL_SVM_TRAINING_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_SVM_TRAINING_PARAMETER_TBL (
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR (1000)
);
INSERT INTO PAL_SVM_TRAINING_PARAMETER_TBL VALUES ('KERNEL_TYPE',2,NULL,NULL);
INSERT INTO PAL_SVM_TRAINING_PARAMETER_TBL VALUES ('TYPE',1,NULL,NULL);
INSERT INTO PAL_SVM_TRAINING_PARAMETER_TBL VALUES ('RBF_GAMMA',NULL,0.005,NULL);
INSERT INTO PAL_SVM_TRAINING_PARAMETER_TBL VALUES ('HAS_ID',1,NULL,NULL);
DROP TABLE PAL_SVM_MODEL_TBL_EX1;
CREATE COLUMN TABLE PAL_SVM_MODEL_TBL_EX1 (ROW_INDEX INTEGER, MODEL_CONTENT
NVARCHAR(5000));
DO BEGIN
  lt_data = SELECT * FROM PAL_SVM_DATA_TBL;
  lt_param = SELECT * FROM PAL_SVM_TRAINING_PARAMETER_TBL;
  CALL _SYS_AFL.PAL_SVM(:lt_data, :lt_param, lt_model, lt_stat, lt_cv);
  INSERT INTO PAL_SVM_MODEL_TBL_EX1 SELECT * FROM :lt_model;
END;
SELECT * FROM PAL_SVM_MODEL_TBL_EX1;
/* Create state */
DROP TABLE PAL_SET_STATE_PARAMETERS_TBL;
CREATE COLUMN TABLE PAL_SET_STATE_PARAMETERS_TBL (
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR (1000)
);
INSERT INTO PAL_SET_STATE_PARAMETERS_TBL VALUES('ALGORITHM', 1, NULL, NULL);
INSERT INTO PAL_SET_STATE_PARAMETERS_TBL VALUES('STATE_DESCRIPTION', NULL, NULL,
'PAL Support Vector Machine');
DROP TABLE PAL_EMPTY_TBL;
CREATE TABLE PAL_EMPTY_TBL(
    ID double
);
DROP TABLE PAL_STATE_TBL;
CREATE TABLE PAL_STATE_TBL (
    S_KEY VARCHAR(50),
    S_VALUE VARCHAR(100)
);
DO BEGIN
  lt_model = SELECT * FROM PAL_SVM_MODEL_TBL_EX1;
  lt_empty = SELECT * FROM PAL_EMPTY_TBL;
  lt_param = SELECT * FROM PAL_SET_STATE_PARAMETERS_TBL;
  CALL
_SYS_AFL.PAL_CREATE_MODEL_STATE(:lt_model, :lt_empty, :lt_empty, :lt_empty, :lt_e
mpty, :lt_param, lt_state);
  INSERT INTO PAL_STATE_TBL SELECT * FROM :lt_state;
END;
SELECT * FROM PAL_STATE_TBL;
-- Predict with state
DROP TABLE PAL_SVM_PREDICT_DATA_TBL;
CREATE COLUMN TABLE PAL_SVM_PREDICT_DATA_TBL(ID INTEGER, ATTRIBUTE1 DOUBLE,
ATTRIBUTE2 DOUBLE, ATTRIBUTE3 DOUBLE, ATTRIBUTE4 VARCHAR(10));
INSERT INTO PAL_SVM_PREDICT_DATA_TBL VALUES(0,1,10,100,'A');
```

```
INSERT INTO PAL_SVM_PREDICT_DATA_TBL VALUES(1,1.2,10.2,100,'A');
INSERT INTO PAL_SVM_PREDICT_DATA_TBL VALUES(2,4.1,40.1,400,'AB');
INSERT INTO PAL_SVM_PREDICT_DATA_TBL VALUES(3,4.2,40.3,400,'AB');
INSERT INTO PAL_SVM_PREDICT_DATA_TBL VALUES(4,9.1,90.1,900,'A');
INSERT INTO PAL_SVM_PREDICT_DATA_TBL VALUES(5,9.2,90.2,900,'A');
INSERT INTO PAL_SVM_PREDICT_DATA_TBL VALUES(6,4,40,400,'A');
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL(
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES('THREAD_RATIO',NULL,0.1,NULL);
INSERT INTO #PAL_PARAMETER_TBL(PARAM_NAME, STRING_VALUE)
    SELECT S_KEY, S_VALUE from PAL_STATE_TBL;
SELECT * FROM #PAL_PARAMETER_TBL;
DROP TABLE PAL_EMPTY_SVM_MODEL_TBL_EX1;
CREATE COLUMN TABLE PAL_EMPTY_SVM_MODEL_TBL_EX1 (ROW_INDEX INTEGER,
MODEL_CONTENT NVARCHAR(5000));
CALL
_SYS_AFL.PAL_SVM_PREDICT(PAL_SVM_PREDICT_DATA_TBL,PAL_EMPTY_SVM_MODEL_TBL_EX1,#PA
L_PARAMETER_TBL,?);
-- Delete state
DROP TABLE #PAL_DELETE_STATE_CONTROL_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_DELETE_STATE_CONTROL_TBL(
    NAME VARCHAR(50),
    INTARGS INTEGER,
    DOUBLEARGS DOUBLE,
    STRINGARGS VARCHAR(100)
);
CALL _SYS_AFL.PAL_DELETE_MODEL_STATE(PAL_STATE_TBL,
#PAL_DELETE_STATE_CONTROL_TBL, ?);
```

### Expected Result

| | ROW_INDEX | MODEL_CONTENT |
|---|---|---|
| 1 | 0 | {"C":100.0,"attr_number":4,"bias":[-0.03010865044869730,0.01637144277035301,0.2088123843101358],"categorical_attr_number":1,"class_numl |

| | STATE_ID | AREA_NAME | DESCRIPTION | CREATE_TIMESTAMP | LAST_ACCESS_TIMESTAMP | HOST | PORT |
|---|---|---|---|---|---|---|---|
| 1 | D525755AEEA383458717CD3A54FC706B | AFLPAL | PAL Support Vector Machine | Mar 8, 2018 12:43:19.882 PM | Mar 8, 2018 12:43:19.882 PM | cnpvg50860233v1 | 31,104 |

| | S_KEY | S_VALUE |
|---|---|---|
| 1 | STATE_ID | D525755AEEA383458717CD3A54FC706B |
| 2 | HINT | 3 |
| 3 | HOST | cnpvg50860233v1 |
| 4 | PORT | 31104 |

| | PARAM_NAME | INT_VALUE | DOUBLE_VALUE | STRING_VALUE |
|---|---|---|---|---|
| 1 | THREAD_RATIO | ? | 0.1 | ? |
| 2 | STATE_ID | ? | ? | D525755AEEA383458717CD3A54FC706B |
| 3 | HINT | ? | ? | 3 |
| 4 | HOST | ? | ? | cnpvg50860233v1 |
| 5 | PORT | ? | ? | 31104 |

| | ID | SCORE | PROBABILITY |
|---|---|---|---|
| 1 | 0 | 1 | ? |
| 2 | 1 | 1 | ? |
| 3 | 2 | 2 | ? |
| 4 | 3 | 2 | ? |
| 5 | 4 | 3 | ? |
| 6 | 5 | 3 | ? |
| 7 | 6 | 2 | ? |

| | ID | TIMESTAMP | ERROR_CODE | MESSAGE |
|---|---|---|---|---|
| | | | | |
| | | | | |

### Example 2: KNN

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_KNN_DATA_TBL;
CREATE COLUMN TABLE PAL_KNN_DATA_TBL (
    "ID" INTEGER,
    "TYPE" NVARCHAR(1),
    "X1" DOUBLE,
    "X2" DOUBLE
);
INSERT INTO PAL_KNN_DATA_TBL VALUES (0, 'B', 1, 1);
INSERT INTO PAL_KNN_DATA_TBL VALUES (1, 'C', 10, 10);
INSERT INTO PAL_KNN_DATA_TBL VALUES (2, 'C', 10, 11);
INSERT INTO PAL_KNN_DATA_TBL VALUES (3, 'C', 10, 10);
INSERT INTO PAL_KNN_DATA_TBL VALUES (4, 'A', 1000, 1000);
INSERT INTO PAL_KNN_DATA_TBL VALUES (5, 'A', 1000, 1001);
INSERT INTO PAL_KNN_DATA_TBL VALUES (6, 'A', 1000, 999);
INSERT INTO PAL_KNN_DATA_TBL VALUES (7, 'A', 999, 999);
INSERT INTO PAL_KNN_DATA_TBL VALUES (8, 'A', 999, 1000);
INSERT INTO PAL_KNN_DATA_TBL VALUES (9, 'A', 1000, 1000);
DROP TABLE PAL_KNN_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_KNN_PARAMETER_TBL (
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR (1000)
);
INSERT INTO PAL_KNN_PARAMETER_TBL VALUES ('K_NEAREST_NEIGHBOURS', 3, NULL, NULL);
INSERT INTO PAL_KNN_PARAMETER_TBL VALUES ('ATTRIBUTE_NUM', 2, NULL, NULL);
INSERT INTO PAL_KNN_PARAMETER_TBL VALUES ('VOTING_TYPE', 0, NULL, NULL);
INSERT INTO PAL_KNN_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 0.1, NULL);
INSERT INTO PAL_KNN_PARAMETER_TBL VALUES ('STAT_INFO', 0, NULL, NULL);
/* Create state */
DROP TABLE PAL_SET_STATE_PARAMETERS_TBL;
CREATE COLUMN TABLE PAL_SET_STATE_PARAMETERS_TBL (
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR (1000)
);
INSERT INTO PAL_SET_STATE_PARAMETERS_TBL VALUES('ALGORITHM', 10, NULL, NULL);
INSERT INTO PAL_SET_STATE_PARAMETERS_TBL VALUES('STATE_DESCRIPTION', NULL, NULL,
'PAL KNN');
INSERT INTO PAL_SET_STATE_PARAMETERS_TBL
  SELECT * FROM PAL_KNN_PARAMETER_TBL;
DROP TABLE PAL_EMPTY_TBL;
CREATE TABLE PAL_EMPTY_TBL(
```

```
    ID double
);
DROP TABLE PAL_STATE_TBL;
CREATE TABLE PAL_STATE_TBL (
    S_KEY VARCHAR(50),
    S_VALUE VARCHAR(100)
);
DO BEGIN
  lt_data = SELECT * FROM PAL_KNN_DATA_TBL;
  lt_empty = SELECT * FROM PAL_EMPTY_TBL;
  lt_param = SELECT * FROM PAL_SET_STATE_PARAMETERS_TBL;
  CALL
_SYS_AFL.PAL_CREATE_MODEL_STATE(:lt_data, :lt_empty, :lt_empty, :lt_empty, :lt_em
pty, :lt_param, lt_state);
  INSERT INTO PAL_STATE_TBL SELECT * FROM :lt_state;
END;
SELECT * FROM PAL_STATE_TBL;
-- Predict with state
DROP TABLE PAL_KNN_CLASSDATA_TBL;
CREATE COLUMN TABLE PAL_KNN_CLASSDATA_TBL(
    "ID" INTEGER,
    "X1" DOUBLE,
    "X2" DOUBLE
);
INSERT INTO PAL_KNN_CLASSDATA_TBL VALUES (0, 2, 1);
INSERT INTO PAL_KNN_CLASSDATA_TBL VALUES (1, 9, 10);
INSERT INTO PAL_KNN_CLASSDATA_TBL VALUES (2, 9, 11);
INSERT INTO PAL_KNN_CLASSDATA_TBL VALUES (3, 15000, 15000);
INSERT INTO PAL_KNN_CLASSDATA_TBL VALUES (4, 1000, 1000);
INSERT INTO PAL_KNN_CLASSDATA_TBL VALUES (5, 500, 1001);
INSERT INTO PAL_KNN_CLASSDATA_TBL VALUES (6, 500, 999);
INSERT INTO PAL_KNN_CLASSDATA_TBL VALUES (7, 199, 999);
DROP TABLE PAL_KNN_PREDICT_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_KNN_PREDICT_PARAMETER_TBL ("PARAM_NAME" VARCHAR(256),
"INT_VALUE" INTEGER, "DOUBLE_VALUE" DOUBLE, "STRING_VALUE" VARCHAR(1000));
INSERT INTO PAL_KNN_PREDICT_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 0.1,
NULL);
INSERT INTO PAL_KNN_PREDICT_PARAMETER_TBL (PARAM_NAME, STRING_VALUE)
    SELECT S_KEY, S_VALUE from PAL_STATE_TBL;
DROP TABLE PAL_EMPTY_KNN_MODEL_TBL;
CREATE COLUMN TABLE PAL_EMPTY_KNN_MODEL_TBL (
    "ID" INTEGER,
    "TYPE" NVARCHAR(1),
    "X1" DOUBLE,
    "X2" DOUBLE
);
CALL _SYS_AFL.PAL_KNN (PAL_EMPTY_KNN_MODEL_TBL, PAL_KNN_CLASSDATA_TBL,
PAL_KNN_PREDICT_PARAMETER_TBL, ?, ?);
-- Delete state
DROP TABLE #PAL_DELETE_STATE_CONTROL_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_DELETE_STATE_CONTROL_TBL(
    NAME VARCHAR(50),
    INTARGS INTEGER,
    DOUBLEARGS DOUBLE,
    STRINGARGS VARCHAR(100)
);
CALL _SYS_AFL.PAL_DELETE_MODEL_STATE(PAL_STATE_TBL,
#PAL_DELETE_STATE_CONTROL_TBL, ?);
```

### Expected Result

| | STATE_ID | AREA_NAME | DESCRIPTION | CREATE_TIMESTAMP | LAST_ACCESS_TIMESTAMP | HOST | PORT |
|---|---|---|---|---|---|---|---|
| 1 | 84B8F95C3776BC478E36DD92ED949C0D | AFLPAL | PAL KNN | Feb 27, 2019 3:27:36.285 PM | Feb 27, 2019 3:27:36.285 PM | cnpvg50860233v1 | 31,204 |

| | S_KEY | S_VALUE |
|---|---|---|
| 1 | STATE_ID | 84B8F95C3776BC478E36DD92ED949C0D |
| 2 | HINT | 3 |
| 3 | HOST | cnpvg50860233v1 |
| 4 | PORT | 31204 |

| | ID | TARGET |
|---|---|---|
| 1 | 0 | C |
| 2 | 1 | C |
| 3 | 2 | C |
| 4 | 3 | A |
| 5 | 4 | A |
| 6 | 5 | A |
| 7 | 6 | A |
| 8 | 7 | A |

| | TEST_ID | K | TRAIN_ID | DISTANCE |
|---|---|---|---|---|
| 1 | 0 | 1 | 0 | 1 |
| 2 | 0 | 2 | 1 | 12.041594578792296 |
| 3 | 0 | 3 | 3 | 12.041594578792296 |
| 4 | 1 | 1 | 1 | 1 |
| 5 | 1 | 2 | 3 | 1 |
| 6 | 1 | 3 | 2 | 1.4142135623730951 |
| 7 | 2 | 1 | 2 | 1 |
| 8 | 2 | 2 | 1 | 1.4142135623730951 |
| 9 | 2 | 3 | 3 | 1.4142135623730951 |
| 10 | 3 | 1 | 5 | 19,798.2827790695 |
| 11 | 3 | 2 | 4 | 19,798.98987322333 |
| 12 | 3 | 3 | 9 | 19,798.98987322333 |
| 13 | 4 | 1 | 4 | 0 |
| 14 | 4 | 2 | 9 | 0 |
| 15 | 4 | 3 | 6 | 1 |
| 16 | 5 | 1 | 8 | 499.001002003002 |
| 17 | 5 | 2 | 7 | 499.0040079999358 |
| 18 | 5 | 3 | 5 | 500 |
| 19 | 6 | 1 | 7 | 499 |
| 20 | 6 | 2 | 8 | 499.001002003002 |
| 21 | 6 | 3 | 6 | 500 |
| 22 | 7 | 1 | 7 | 800 |
| 23 | 7 | 2 | 8 | 800.0006249997558 |
| 24 | 7 | 3 | 6 | 801 |

| | ID | TIMESTAMP | ERROR_CODE | MESSAGE |
|---|---|---|---|---|
| | | | | |
| | | | | |

## Related Information

SAP HANA SQLScript Reference for SAP HANA Platform

# 3.7 Exception Handling

Exceptions thrown by a PAL procedure can be caught by the exception handler in a SQLScript procedure with AFL error code 423.

The following shows an example of handling the exceptions thrown by an ARIMA function.
In the example, you are generating the `"DM_PAL".PAL_ARIMAXTRAIN_PROC` procedure to call an ARIMA function, whose input data table is empty. You create a SQLScript procedure `"DM_PAL".PAL_ARIIMAX_NON_EXCEPTION_PROC` which calls the `"_SYS_AFL"."PAL_ARIMA` procedure. When you call the `"DM_PAL".PAL_ARIIMAX_NON_EXCEPTION_PROC` procedure, the exception handler in the procedure will catch the errors thrown by the procedure.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_ARIMAX_DATA_TBL;
CREATE COLUMN TABLE PAL_ARIMAX_DATA_TBL (
    "TIMESTAMP" INTEGER,
    "Y" DOUBLE,
    "X1" DOUBLE
);
DROP TYPE PAL_PARAMETER_T;
CREATE TYPE PAL_PARAMETER_T AS TABLE (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL LIKE PAL_PARAMETER_T;
INSERT INTO #PAL_PARAMETER_TBL VALUES ('P', 1,null,null);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('Q', 1,null,null);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('D', 0,null,null);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('METHOD', 1,null,null);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('STATIONARY', 1,null,null);
DROP TABLE PAL_ARIMAX_MODEL_TBL;
CREATE COLUMN TABLE PAL_ARIMAX_MODEL_TBL (
    "KEY" VARCHAR (100),
    "VALUE" VARCHAR (5000)
);
DROP PROCEDURE "DM_PAL".PAL_ARIIMAX_NON_EXCEPTION_PROC;
CREATE PROCEDURE PAL_ARIIMAX_NON_EXCEPTION_PROC(IN training_data
PAL_ARIMAX_DATA_TBL, IN para_args PAL_PARAMETER_T, OUT result_model
PAL_ARIMAX_MODEL_TBL)
LANGUAGE SQLSCRIPT AS
BEGIN
```

```
    -- used to catch exceptions
    DECLARE EXIT HANDLER FOR SQLEXCEPTION
SELECT ::SQL_ERROR_CODE, ::SQL_ERROR_MESSAGE FROM DUMMY;
    CALL "_SYS_AFL"."PAL_ARIMA"(:training_data, :para_args, result_model,
result_fit);
END;
CALL PAL_ARIIMAX_NON_EXCEPTION_PROC(PAL_ARIMAX_DATA_TBL, #PAL_PARAMETER_TBL, ?);
```

### Expected Result

| | :1 | :2 |
|---|---|---|
| 1 | 423 | AFL error: "DM_PAL"."PAL_ARIIMAX_NON_EXCEPTION_PROC": line 6 col 5 (at pos 323): AFL error: search table error: _SYS_AFL.AFLPAL:ARIMA_ANY: [423] (range 3) AFL error exception: exception 73001021: PAL error[73001021]:Input table  is empty |

# 4 Using PAL in SAP HANA AFM

The SAP HANA Application Function Modeler (AFM) in SAP HANA Studio supports functions from PAL in flowgraph models. With the AFM, you can easily add PAL function nodes to your flowgraph, specify its parameters and input/output table types, and generate the procedure, all without writing any SQLScript code.

You can also execute the procedure to get the output result of the function, and save the auto-generated SQLScript code for future use.

The main procedure is as follows:

1. Create a new flowgraph or open an existing flowgraph in the *Project Explorer* view.

   > **Note**
   >
   > For details on how to create a flowgraph, see the section on creating a flowgraph in *SAP HANA Developer Guide for SAP HANA Studio*.

2. Specify the target schema by selecting the flowgraph container and editing *Target Schema* in the *Properties* view.
3. Add the input(s) for the flowgraph by doing the following:
   1. Right-click the input anchor region on the left side of the flowgraph container and choose *Add Input*.
   2. Edit the table types of the input by editing the signature the *Properties* view.

   > **Note**
   >
   > You can also drag a table from the catalog in the *Systems* view to the input anchor region of the flowgraph container.

4. Add a PAL function to the flowgraph by doing the following:
   1. Drag the function node from the *Predictive Analysis Library* compartment of the *Palette* to the flowgraph editing area.
   2. Specify the input table types of the function by selecting the input anchor and editing its signature in the *Properties* view.
   3. Specify the parameters of the function by selecting the parameter input anchor and editing its signature and fixed content in the *Properties* view.

      > **Note**
      >
      > A parameter table usually has fixed table content. If you want to supply the parameter values later when you execute the procedure, clear the *Fixed Content* option.

   4. Specify the output table types of the function by selecting the output anchor and editing the signature in the *Properties* view.
5. (Optional) You can add more PAL nodes to the flowgraph if needed and connect them by holding the *Connect* button from the source anchor and dragging a connection to the destination anchor.
6. Connect the input(s) in the input anchor region of the flowgraph container to the required input anchor(s) of the PAL function node.
7. For the output tables that you want to see the output result after procedure execution, add them to the output anchor region on the right side of the flowgraph container. To do that, move your mouse cursor over

the output anchor of the function node, hold the *Connect* button, and drag a connection to the output anchor region.

8. Save the flowgraph by choosing *File* > *Save* in the HANA Studio main menu.
9. Activate the flowgraph by right-clicking the flowgraph in the *Project Explorer* view and choosing *Team* > *Activate*.
   A new procedure is generated in the target schema which is specified in Step 2.

   > **Note**
   >
   > To activate the flowgraph, the database user _SYS_REPO needs SELECT object privileges for objects that are used as data sources.

10. Select the black downward triangle next to the *Execute* button in the top right corner of the AFM.
    A context menu appears. It shows the options *Execute in SQL Editor* and *Open in SQL Editor* as well as the option *Execute and Explore* for every output of the flowgraph. In addition, the context menu shows the option *Edit Input Bindings*.
11. (Optional) If the flowgraph has input tables without fixed content, choose the option *Edit Input Bindings*.
    A wizard appears that allows you to bind all inputs of the flowgraph to data sources in the catalog.

    > **Note**
    >
    > If you do not bind the inputs, AFM will automatically open this wizard when executing the procedure.

12. Choose one of the options *Execute in SQL Editor*, *Open in SQL Editor*, or *Execute and Explore* for one of the outputs of the flowgraph.
    The behavior of the AFM depends on the execution mode.
    - *Open in SQL Editor*: Opens a SQL console containing the SQL code to execute the runtime object.
    - *Execute in SQL Editor*: Opens a SQL console containing the SQL code to execute the runtime object and runs this SQL code.
    - *Execute and Explore*: Executes the runtime object and opens the *Data Explorer* view for the chosen output of the flowgraph.
13. Close the flowgraph by choosing *File* > *Close* in the HANA Studio main menu.

For more information on how to use AFM, see the section on transforming data using SAP HANA Application Function Modeler in *SAP HANA Developer Guide for SAP HANA Studio*.

## Related Information

SAP HANA Developer Guide for SAP HANA Studio

# 5 PAL Procedures

The following are the available algorithms and functions in the Predictive Analysis Library.

> **Note**
>
> The **Built-in Function Name** column in the table shows the old interfaces used prior to SAP HANA Platform 2.0 SPS 02. Starting from SPS 02, the procedures shown in the **PAL Procedure Name** column are used.

| Category | PAL Algorithm | PAL Procedure Name | Built-in Function Name |
|---|---|---|---|
| Clustering | Accelerated K-Means [page 75] | PAL_ACCELERATED_KMEANS | ACCELERATEDKMEANS |
| | Affinity Propagation [page 84] | PAL_AFFINITY_PROPAGATION | AP |
| | Agglomerate Hierarchical Clustering [page 90] | PAL_HIERARCHICAL_CLUSTERING | HCAGGLOMERATE |
| | Anomaly Detection [page 99] | PAL_ANOMALY_DETECTION | |
| | Cluster Assignment [page 106] | PAL_CLUSTER_ASSIGNMENT | CLUSTERASSIGNMENT |
| | DBSCAN [page 111] | PAL_DBSCAN | DBSCAN |
| | Gaussian Mixture Model (GMM) [page 117] | PAL_GMM | GMM |
| | Geometry DBSCAN [page 125] | PAL_GEO_DBSCAN | |
| | K-Means [page 131] | PAL_KMEANS<br>PAL_VALIDATE_KMEANS | KMEANS<br>VALIDATEKMEANS |
| | K-Medians [page 143] | PAL_KMEDIANS | KMEDIANS |
| | K-Medoids [page 150] | PAL_KMEDOIDS | KMEDOIDS |
| | Self-Organizing Maps [page 157] | PAL_SOM | SELFORGMAP |
| | Slight Silhouette [page 163] | PAL_SLIGHT_SILHOUETTE | |

| Category | PAL Algorithm | PAL Procedure Name | Built-in Function Name |
|---|---|---|---|
| | Spectral Clustering [page 167] | PAL_SPECTRAL_CLUSTERING | |
| Classification | Area Under Curve (AUC) [page 174] | PAL_AUC<br>PAL_MULTICLASS_AUC<br>PAL_ROC_AUC | AUC |
| | C4.5 Decision Tree<br>(See *Decision Trees*) | PAL_DECISION_TREE<br>PAL_DECISION_TREE_PREDICT | CREATEDTWITHC45 |
| | CART Decision Tree<br>(See *Decision Trees*) | PAL_DECISION_TREE<br>PAL_DECISION_TREE_PREDICT | CART |
| | CHAID Decision Tree<br>(See *Decision Trees*) | PAL_DECISION_TREE<br>PAL_DECISION_TREE_PREDICT | CREATEDTWITHCHAID |
| | Confusion Matrix [page 183] | PAL_CONFUSION_MATRIX | CONFUSIONMATRIX |
| | Decision Trees [page 187] | PAL_DECISION_TREE<br>PAL_DECISION_TREE_PREDICT | DECISIONTREE |
| | Hybrid Gradient Boosting Tree [page 203] | PAL_HGBT<br>PAL_HGBT_PREDICT | |
| | KNN [page 236] | PAL_KNN<br>PAL_KNN_CV | KNN |
| | Linear Discriminant Analysis [page 265] | PAL_LINEAR_DISCRIMINANT_ANALYSIS<br>PAL_LINEAR_DISCRIMINANT_ANALYSIS_CLASSIFY<br>PAL_LINEAR_DISCRIMINANT_ANALYSIS_PROJECT | LDAFIT<br>LDACLASSIFY<br>LDAPROJECT |
| | Logistic Regression (with Elastic Net Regularization) [page 279] | PAL_LOGISTIC_REGRESSION<br>PAL_LOGISTIC_REGRESSION_PREDICT | LOGISTICREGRESSION<br>FORECASTWITHLOGISTICR |

| Category | PAL Algorithm | PAL Procedure Name | Built-in Function Name |
|---|---|---|---|
| | Multi-Class Logistic Regression [page 304] | PAL_MULTICLASS_LOGISTIC_REGRESSION<br>PAL_MULTICLASS_LOGISTIC_REGRESSION_PREDICT | LRMCTR<br>LRMCTE |
| | Multilayer Perceptron [page 330] | PAL_MULTILAYER_PERCEPTRON<br>PAL_MULTILAYER_PERCEPTRON_PREDICT | CREATEBPNN<br>PREDICTWITHBPNN |
| | Naive Bayes [page 352] | PAL_NAIVE_BAYES<br>PAL_NAIVE_BAYES_PREDICT | NBCTRAIN<br>NBCPREDICT |
| | Online Multi-Class Logistic Regression [page 364] | PAL_INIT_ONLINE_MULTICLASS_LOGISTIC_REGRESSION | |
| | Random Decision Trees [page 381] | PAL_RANDOM_DECISION_TREES<br>PAL_RANDOM_DECISION_TREES_PREDICT | |
| | Support Vector Machine [page 399] | PAL_SVM<br>PAL_SVM_PREDICT | SVMTRAIN<br>SVMPREDICT |
| Regression | Bi-Variate Geometric Regression [page 424] | PAL_GEOMETRIC_REGRESSION<br>PAL_GEOMETRIC_REGRESSION_PREDICT | GEOREGRESSION<br>FORECASTWITHGEOR |
| | Bi-Variate Natural Logarithmic Regression [page 431] | PAL_LOGARITHMIC_REGRESSION<br>PAL_LOGARITHMIC_REGRESSION_PREDICT | LNREGRESSION<br>FORECASTWITHLNR |
| | Cox Proportional Hazard Model [page 438] | PAL_COXPH<br>PAL_COXPH_PREDICT | COXPHESTIMATE<br>COXPHPREDICT |
| | Exponential Regression [page 448] | PAL_EXPONENTIAL_REGRESSION<br>PAL_EXPONENTIAL_REGRESSION_PREDICT | EXPREGRESSION<br>FORECASTWITHEXPR |

| Category | PAL Algorithm | PAL Procedure Name | Built-in Function Name |
|---|---|---|---|
| | Generalised Linear Models [page 456] | PAL_GLM<br>PAL_GLM_PREDICT | GLMESTIMATE<br>GLMPREDICT |
| | Multiple Linear Regression [page 478] | PAL_LINEAR_REGRESSION<br>PAL_LINEAR_REGRES-SION_PREDICT | LRREGRESSION<br>FORECASTWITHLR |
| | Online Linear Regression (Stateless) [page 507] | PAL_CREATE_ONLINE_LIN-EAR_REGRESSION_STATE<br>PAL_ONLINE_LINEAR_RE-GRESSION<br>PAL_DELETE_ONLINE_LIN-EAR_REGRESSION_STATE | |
| | Polynomial Regression [page 513] | PAL_POLYNOMIAL_REGRES-SION<br>PAL_POLYNOMIAL_REGRES-SION_PREDICT | POLYNOMIALREGRESSION<br>FORECASTWITHPOLYNO-MIALR |
| Association | Apriori [page 524] | PAL_APRIORI<br>PAL_LITE_APRIORI<br>PAL_APRIORI_RELATIONAL | APRIORIRULE<br>LITEAPRIORIRULE<br>APRIORIRULE2 |
| | FP-Growth [page 542] | PAL_FPGROWTH<br>PAL_FPGROWTH_RELA-TIONAL | FPGROWTH |
| | K-Optimal Rule Discovery (KORD) [page 552] | PAL_KORD | KORD |
| | Sequential Pattern Mining [page 557] | PAL_SPM<br>PAL_SPM_RELATIONAL | SPM |

| Category | PAL Algorithm | PAL Procedure Name | Built-in Function Name |
|---|---|---|---|
| Time Series | Additive Model Time Series Analysis [page 566] | PAL_ADDITIVE_MODEL_ANALYSIS<br>PAL_ADDITIVE_MODEL_PREDICT<br>PAL_ADDITIVE_MODEL_EXPLAIN<br>PAL_MASSIVE_ADDITIVE_MODEL_ANALYSIS<br>PAL_MASSIVE_ADDITIVE_MODEL_PREDICT | |
| | ARIMA [page 593] | PAL_ARIMA<br>PAL_ARIMA_FORECAST<br>PAL_ARIMA_EXPLAIN<br>PAL_MASSIVE_ARIMA<br>PAL_MASSIVE_ARIMA_FORECAST<br>PAL_MASSIVE_ARIMA_EXPLAIN | ARIMATRAIN<br>ARIMAFORECAST |
| | Auto ARIMA [page 627] | PAL_AUTOARIMA<br>PAL_MASSIVE_AUTOARIMA | AUTOARIMA |
| | Auto Exponential Smoothing [page 651] | PAL_AUTO_EXPSMOOTH | FORECASTSMOOTHING |
| | Bayesian Change Point Detection [page 671] | PAL_BCPD | |
| | Bayesian Structural Time Series [page 679] | PAL_BSTS_TRAIN<br>PAL_BSTS_PREDICT | |
| | Brown Exponential Smoothing [page 689] | PAL_BROWN_EXPSMOOTH | BROWNEXPSMOOTH |
| | Change-Point Detection [page 695] | PAL_CPDETECTION | |
| | Croston's Method [page 702] | PAL_CROSTON | CROSTON |

| Category | PAL Algorithm | PAL Procedure Name | Built-in Function Name |
|---|---|---|---|
| | Croston TSB Method [page 707] | PAL_CROSTONTSB<br>PAL_MASSIVE_CROSTONTSB | |
| | Discrete Wavelet Packet Transform [page 722] | PAL_WAVELET_PACKET_TRANSFORM<br>PAL_WAVELET_PACKET_TRANSFORM_INVERSE | |
| | Discrete Wavelet Transform [page 732] | PAL_DISCRETE_WAVELET_TRANSFORM<br>PAL_DISCRETE_WAVELET_TRANSFORM_INVERSE | |
| | Fast Dynamic Time Warping [page 750] | PAL_FAST_DTW | |
| | Fast Fourier Transform [page 758] | PAL_FFT | FFT |
| | Forecast Accuracy Measures [page 762] | PAL_ACCURACY_MEASURES | ACCURACYMEASURES |
| | GARCH [page 766] | PAL_GARCH<br>PAL_GARCH_FORECAST<br>PAL_GARCH_PREDICT | |
| | Hierarchical Forecast [page 776] | PAL_HIERARCHICAL_FORECAST | |
| | Intermittent Time Series Forecast (ITSF) [page 783] | PAL_ITSF | |
| | Linear Regression with Damped Trend and Seasonal Adjust [page 788] | PAL_LR_SEASONAL_ADJUST | LRWITHSEASONALADJUST |
| | Long-Term Series Forecasting [page 794] | PAL_LTSF_TRAIN<br>PAL_LTSF_TRAIN_CONTINUE<br>PAL_LTSF_PREDICT | |

| Category | PAL Algorithm | PAL Procedure Name | Built-in Function Name |
|---|---|---|---|
| | Online ARIMA [page 806] | PAL_ONLINE_ARIMA<br>PAL_ONLINE_ARIMA_FORE-CAST | |
| | Online Bayesian Change Point Detection [page 815] | PAL_ONLINE_BCPD | |
| | Outlier Detection [page 821] | PAL_OUTLIER_DETEC-TION_FOR_TIME_SERIES | |
| | Periodogram [page 888] | PAL_PERIODOGRAM | |
| | ROCKET [page 892] | PAL_ROCKET_TRAIN<br>PAL_ROCKET_PREDICT<br>PAL_ROCKET_FIT<br>PAL_ROCKET_TRANSFORM | |
| | Single Exponential Smooth-ing [page 912] | PAL_SINGLE_EXPSMOOTH | SINGLESMOOTH |
| | Double Exponential Smooth-ing [page 917] | PAL_DOUBLE_EXPSMOOTH | DOUBLESMOOTH |
| | Triple Exponential Smoothing [page 924] | PAL_TRIPLE_EXPSMOOTH | TRIPLESMOOTH |
| | Seasonality Test [page 934] | PAL_SEASONALITY_TEST | SEASONALITYTEST |
| | Stationarity Test [page 941] | PAL_STATIONARITY_TEST | |
| | Trend Test [page 946] | PAL_TREND_TEST | TRENDTEST |
| | White Noise Test [page 972] | PAL_WN_TEST | WHITENOISETEST |
| Pipeline | Pipeline [page 975] | PAL_PIPELINE_FIT<br>PAL_PIPELINE_PREDICT<br>PAL_PIPELINE_EVALUATE<br>PAL_PIPELINE_EXPLAIN | |
| AutoML | AutoML [page 1013] | PAL_AUTOML_CONFIG<br>PAL_AUTOML_FIT | |
| Preprocessing | Categorical Principal Compo-nent Analysis [page 1050] | PAL_CATPCA<br>PAL_CATPCA_PROJECT | |

| Category | PAL Algorithm | PAL Procedure Name | Built-in Function Name |
|---|---|---|---|
| | Discretize [page 1061] | PAL_DISCRETIZE | |
| | Discretize Apply [page 1067] | PAL_DISCRETIZE_APPLY | |
| | Feature Selection [page 1071] | PAL_FEATURE_SELECTION | |
| | Isolation Forest [page 1078] | _SYS_AFL.PAL_ISOLATION_FOREST<br>_SYS_AFL.PAL_ISOLATION_FOREST_PREDICT | |
| | Inter-Quartile Range [page 1084] | PAL_IQR | IQRTEST |
| | MCMC Sampling [page 1087] | PAL_MCMC | |
| | Multidimensional Scaling (MDS) [page 1120] | PAL_MDS | |
| | Missing Value Handling [page 1095] | PAL_MISSING_VALUE_HANDLING | |
| | Partition [page 1124] | PAL_PARTITION | PARTITION |
| | Power Transform [page 1129] | PAL_POWER_TRANSFORMER_FIT<br>PAL_POWER_TRANSFORMER_TRANSFORM | |
| | Principal Component Analysis (PCA) [page 1139] | PAL_PCA<br>PAL_PCA_PROJECT | PCA<br>PCAPROJECTION |
| | Random Distribution Sampling [page 1147] | PAL_DISTRIBUTION_RANDOM<br>PAL_DISTIBUTION_RANDOM_MULTIVARIATE | DISTRRANDOM |
| | Sampling [page 1158] | PAL_SAMPLING | SAMPLING |
| | Scale [page 1167] | PAL_SCALE | SCALINGRANGE |
| | Scale with Model [page 1172] | PAL_SCALE_WITH_MODEL | POSTERIORSCALING |
| | SMOTETomek [page 1184] | PAL_SMOTETOMEK | |
| | SMOTE [page 1176] | PAL_SMOTE | |
| | Tomek's Links [page 1189] | PAL_TOMEKLINKS | |

| Category | PAL Algorithm | PAL Procedure Name | Built-in Function Name |
|---|---|---|---|
| | Variance Test [page 1195] | PAL_VARIANCE_TEST | VARIANCETEST |
| Statistics | Anova [page 1200] | PAL_ONEWAY_ANOVA<br>PAL_ONEWAY_REPEATED_MEASURES_ANOVA | ANOVAONEWAY<br>ANOVAONEWAYRM |
| | Chi-Square Goodness-of-Fit Test [page 1211] | PAL_CHISQUARED_GOF_TEST | CHISQTESTFIT |
| | Chi-Squared Test of Independence [page 1213] | PAL_CHISQUARED_IND_TEST | CHISQTESTIND |
| | Condition Index [page 1216] | PAL_CONDITION_INDEX | CONDITIONINDEX |
| | Correlation Function [page 1220] | PAL_CORRELATION_FUNCTION<br>PAL_CORRELATION_FUNCTION_WITH_CONFIDENCE | CORRELATIONFUNCTION |
| | Cumulative Distribution Function [page 1229] | PAL_DISTRIBUTION_FUNCTION | DISTRPROB |
| | Distribution Fitting [page 1233] | PAL_DISTRIBUTION_FIT<br>PAL_DISTRIBUTION_FIT_CENSORED | DISTRFIT<br>DISTRFITCENSORED |
| | Distribution Quantile [page 1240] | PAL_DISTRIBUTION_QUANTILE | DISTRQUANTILE |
| | Entropy [page 1243] | PAL_ENTROPY | |
| | Equal Variance Test [page 1247] | PAL_EQUAL_VARIANCE_TEST | VAREQUALTEST |
| | Factor Analysis [page 1250] | PAL_FACTOR_ANALYSIS | |
| | Grubbs' Test [page 1258] | PAL_GRUBBS_TEST | GRUBBSTEST |
| | Kaplan-Meier Survival Analysis [page 1262] | PAL_KMSURV | KMSURV |
| | Kernel Density [page 1267] | PAL_KDE | |
| | Kolmogorov-Smirnov Test [page 1278] | PAL_KS_TEST | |

| Category | PAL Algorithm | PAL Procedure Name | Built-in Function Name |
|---|---|---|---|
| | Multivariate Analysis [page 1283] | PAL_MULTIVARIATE_ANALYSIS | MULTIVARSTAT |
| | One-Sample Median Test [page 1286] | PAL_SIGN_TEST | SIGNTEST |
| | T-Test [page 1289] | PAL_T_TEST | TTEST |
| | Univariate Analysis [page 1294] | PAL_UNIVARIATE_ANALYSIS | DATASUMMARY |
| | Wilcox Signed Rank Test [page 1300] | PAL_WILCOX_TEST | WILCOXTEST |
| Social Network Analysis | Link Prediction [page 1303] | PAL_LINK_PREDICT | LINKPREDICTION |
| | PageRank [page 1307] | PAL_ PAGERANK | |
| Recommender System | Alternating Least Squares [page 1310] | PAL_ALS<br>PAL_ALS_PREDICT | |
| | Factorized Polynomial Regression Models [page 1328] | PAL_FRM<br>PAL_FRM_PREDICT | FRMTRAIN<br>FRMPREDICT |
| | Field-Aware Factorization Machine [page 1354] | PAL_FFM<br>PAL_FFM_PREDICT | |
| Miscellaneous | ABC Analysis [page 1372] | PAL_ABC | ABC |
| | Model Visualization [page 1375] | PAL_VISUALIZE_MODEL | |
| | T-distributed Stochastic Neighbour Embedding [page 1378] | PAL_TSNE | |
| | Weighted Score Table [page 1386] | PAL_WEIGHTED_TABLE | WEIGHTEDTABLE |
| Unified Interface | Unified Classification [page 1390] | PAL_UNIFIED_CLASSIFICATION<br>PAL_UNIFIED_CLASSIFICATION_PREDICT<br>PAL_UNIFIED_CLASSIFICATION_SCORE | |

| Category | PAL Algorithm | PAL Procedure Name | Built-in Function Name |
|---|---|---|---|
| | Unified Classification with Pivoted Input Data [page 1575] | PAL_UNIFIED_CLASSIFICATION_PIVOT<br>PAL_UNIFIED_CLASSIFICATION_PREDICT_PIVOT<br>PAL_UNIFIED_CLASSIFICATION_SCORE_PIVOT | |
| | Unified Massive Classification [page 1753] | PAL_UNIFIED_MASSIVE_CLASSIFICATION<br>PAL_UNIFIED_MASSIVE_CLASSIFICATION_PREDICT<br>PAL_UNIFIED_MASSIVE_CLASSIFICATION_SCORE | |
| | Unified Massive Classification with Pivoted Input Data [page 1768] | PAL_UNIFIED_MASSIVE_CLASSIFICATION_PIVOT<br>PAL_UNIFIED_MASSIVE_CLASSIFICATION_PREDICT_PIVOT<br>PAL_UNIFIED_MASSIVE_CLASSIFICATION_SCORE_PIVOT | |
| | Unified Clustering [page 1785] | PAL_UNIFIED_CLUSTERING<br>PAL_UNIFIED_CLUSTERING_ASSIGNMENT | |
| | Unified Clustering with Pivoted Input Data [page 1831] | PAL_UNIFIED_CLUSTERING_PIVOT<br>PAL_UNIFIED_CLUSTERING_ASSIGNMENT_PIVOT | |
| | Unified Massive Clustering [page 1874] | PAL_UNIFIED_MASSIVE_CLUSTERING<br>PAL_UNIFIED_MASSIVE_CLUSTERING_ASSIGNMENT | |

| Category | PAL Algorithm | PAL Procedure Name | Built-in Function Name |
|---|---|---|---|
| | Unified Massive Clustering with Pivoted Input Data [page 1883] | PAL_UNIFIED_MASSIVE_CLUSTERING_PIVOT<br>PAL_UNIFIED_MASSIVE_CLUSTERING_ASSIGNMENT_PIVOT | |
| | Unified Exponential Smoothing [page 2157] | PAL_UNIFIED_EXPONENTIALSMOOTHING | |
| | Unified Exponential Smoothing with Pivoted Input Data [page 2175] | PAL_UNIFIED_EXPONENTIALSMOOTHING_PIVOT | |
| | Unified Regression [page 1895] | PAL_UNIFIED_REGRESSION<br>PAL_UNIFIED_REGRESSION_PREDICT<br>PAL_UNIFIED_REGRESSION_SCORE | |
| | Unified Regression with Pivoted Input Data [page 2015] | PAL_UNIFIED_REGRESSION_PIVOT<br>PAL_UNIFIED_REGRESSION_PREDICT_PIVOT<br>PAL_UNIFIED_REGRESSION_SCORE_PIVOT | |
| | Unified Massive Regression [page 2133] | PAL_UNIFIED_MASSIVE_REGRESSION<br>PAL_UNIFIED_MASSIVE_REGRESSION_PREDICT<br>PAL_UNIFIED_MASSIVE_REGRESSION_SCORE | |
| | Unified Massive Regression with Pivoted Input Data [page 2144] | PAL_UNIFIED_MASSIVE_REGRESSION_PIVOT<br>PAL_UNIFIED_MASSIVE_REGRESSION_PREDICT_PIVOT<br>PAL_UNIFIED_MASSIVE_REGRESSION_SCORE_PIVOT | |
| Text Processing | Conditional Random Field [page 2196] | PAL_CRF<br>PAL_CRF_INFERENCE | |

| Category | PAL Algorithm | PAL Procedure Name | Built-in Function Name |
|---|---|---|---|
| | Latent Dirichlet Allocation [page 2209] | PAL_LATENT_DIRICH-LET_ALLOCATION<br>PAL_LATENT_DIRICH-LET_ALLOCATION_INFER-ENCE | LDAESTIMATE<br>LDAINFERENCE |
| | TF-IDF Calculation [page 2221] | PAL_TEXT_COLLECT<br>PAL_TEXT_TFIDF | |

## 5.1 Clustering Algorithms

This section describes the clustering algorithms that are provided by the Predictive Analysis Library.

### 5.1.1 Accelerated K-Means

Accelerated k-means is a variant of the k-means algorithm. It uses some technology to accelerate the calculation.

Like k-means, accelerated k-means also has two steps in each iteration: assigning each point to a cluster with closest distance to its center; calculating new center of each cluster. Moreover, the center set and cluster partition set are exactly the same with the ordinary k-means after each iteration.

The technologies used to accelerate iterations usually involve caches that store information between iterations. Therefore, it tends to use more memory than the ordinary k-means.

In PAL, another big different between accelerated k-means and ordinary k-means is that accelerated k-means keeps iterating until all clusters stop changing, instead of using an EXIT_THRESHOLD parameter which makes it possible to stop earlier. Other than that, accelerated k-means looks just like ordinary k-means.

**Support for Categorical Attributes**

If an attribute is of category type, it will be converted to a binary vector and then be used as a numerical attribute. For example, in the below table, "Gender" is of category type.

| Customer ID | Age | Income | Gender |
|---|---|---|---|
| T1 | 31 | 10,000 | Female |
| T2 | 27 | 8,000 | Male |

Because "Gender" has two distinct values, it will be converted into a binary vector with two dimensions:

| Customer ID | Age | Income | Gender_1 | Gender_2 |
|---|---|---|---|---|
| T1 | 31 | 10,000 | 0 | 1 |
| T2 | 27 | 8,000 | 1 | 0 |

Thus, the Euclidean distance between T1 and T2 is:

$$d(T_1, T_2)=\sqrt{(31-27)^2+(10000-8000)^2}+\gamma\sqrt{(0-1)^2+(1-0)^2}$$

Where γ is the weight to be given to the transposed categorical attributes to lessen the impact on the clustering from the 0/1 attributes. Then you can use the traditional method to update the mean of every cluster. Assuming one cluster only has T1 and T2, the mean is:

| Customer ID | Age | Income | Gender_1 | Gender_2 |
|---|---|---|---|---|
| Center1 | 29.0 | 9000.0 | 0.5 | 0.5 |

The means of categorical attributes will not be outputted. Instead, the means will be replaced by the modes similar to the k-modes algorithm. Take the below center for example:

| | Age | Income | Gender_1 | Gender_2 |
|---|---|---|---|---|
| Center | 29.0 | 9000.0 | 0.25 | 0.75 |

Because "Gender_2" is the maximum value, the output will be:

| | Age | Income | Gender |
|---|---|---|---|
| Center | 29.0 | 9000.0 | Female |

## Prerequisites

- The input data contains an ID column.
- The input data does not contain null value. The algorithm will issue errors when encountering null values.

## _SYS_AFL.PAL_ACCELERATED_KMEANS

This is a clustering function using the accelerated k-means algorithm.

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<RESULT table>` | OUT |
| 4 | `<CENTERS table>` | OUT |
| 5 | `<MODEL table>` | OUT |
| 6 | `<STATISTICS table>` | OUT |
| 7 | `<PLACEHOLDER table>` | OUT |

### Input Table

| Table | Column | Reference for Output Table | Data Type | Description |
|---|---|---|---|---|
| DATA | 1st column | ID | INTEGER, BIGINT, VARCHAR, or NVARCHAR | Record ID |
| | Other columns | FEATURES | INTEGER, DOUBLE, DECIMAL(p, s), VARCHAR, or NVARCHAR | Attribute data |

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| GROUP_NUMBER | INTEGER | No default value | Number of groups (k). The value range is from 2 to the number of training records.<br><br>**Note**: If this parameter is not specified, you must specify the range of k using GROUP_NUMBER_MIN and GROUP_NUMBER_MAX. Then the algorithm will iterate through the range and return the k with the highest slight silhouette. | |
| GROUP_NUMBER_MIN | INTEGER | No default value | Lower boundary of the range that k falls in.<br><br>**Note**: You must specify either an exact value or a range for k. If both are specified, the exact value will be used. | |
| GROUP_NUMBER_MAX | INTEGER | No default value | Upper boundary of the range that k falls in.<br><br>**Note**: You must specify either an exact value or a range for k. If both are specified, the exact value will be used. | |
| DISTANCE_LEVEL | INTEGER | 2 | Computes the distance between the item and the cluster center.<br>• 1: Manhattan distance<br>• 2: Euclidean distance<br>• 3: Minkowski distance<br>• 4: Chebyshev distance | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| MINKOWSKI_POWER | DOUBLE | 3.0 | When you use the Minkowski distance, this parameter controls the value of power. | Only valid when DISTANCE_LEVEL is 3. |
| CATEGORY_WEIGHTS | DOUBLE | 0.707 | Represents the weight of category attributes (γ). | |
| MAX_ITERATION | INTEGER | 100 | Maximum iterations. | |
| INIT_TYPE | INTEGER | 4 | Controls how the initial centers are selected:<br>• 1: First k observations<br>• 2: Random with replacement<br>• 3: Random without replacement<br>• 4: Patent of selecting the init center (US 6,882,998 B1) | |
| NORMALIZATION | INTEGER | 0 | Normalization type:<br>• 0: No<br>• 1: Yes. For each point `X (x1,x2,...,xn)`, the normalized value will be `X'(x1/S,x2/S,...,xn/S)`, where `S = \|x1\|+\|x2\|+...\|xn\|`.<br>• 2: for each column `C`, get the min and max value of `C`, and then `C[i] = (C[i]-min)/(max-min)`. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| THREAD_RATIO | DOUBLE | 0 | Specifies the ratio of total number of threads that can be used by this function. The value range is from 0 to 1, where 0 means only using 1 thread, and 1 means using at most all the currently available threads. Values outside the range will be ignored and this function heuristically determines the number of threads to use. | |
| CATEGORICAL_VARIABLE | NVARCHAR | No default value | Indicates whether or not a column data is actually corresponding to a category variable even the data type of this column is INTEGER.<br><br>By default, 'VARCHAR' or 'NVARCHAR' is category variable, and 'INTEGER' or 'DOUBLE' is continuous variable. | |
| MEMORY_MODE | INTEGER | 0 | Indicates the memory mode that the algorithm uses:<br>• 0: Chosen by algorithms<br>• 1: Takes speed as priority<br>• 2: Takes memory saving as priority | |
| USE_FLOAT | INTEGER | 1 | Floating point type:<br>• 0: double<br>• 1: float | Only valid when USE_FAST_LIBRARY is 1. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| USE_FAST_LIBRARY | INTEGER | 0 | Uses vectorized accelerated operation when it is set to 1. | |

### Output Tables

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| RESULT | 1st column | ID data type | ID column name | Record ID |
| | 2nd column | INTEGER | CLUSTER_ID | Cluster ID assigned to it |
| | 3rd column | DOUBLE | DISTANCE | The distance between the given point and the cluster center |
| | 4th column | DOUBLE | SLIGHT_SILHOUETTE | Estimated value (slight silhouette) |
| CENTERS | 1st column | INTEGER | CLUSTER_ID | Cluster center ID |
| | Other columns | FEATURES (in input table) data type with conversion from INTEGER to DOUBLE | FEATURES (in input table) column name | Cluster center coordinates |
| MODEL | 1st column | INTEGER | ROW_INDEX | Cluster model row index |
| | 2nd column | VARCHAR or NVARCHAR | MODEL_CONTENT | Cluster model save as JSON string.<br>The table must be a column table. The minimum length of every unit (row) is 5000 |
| STATISTICS | 1st column | VARCHAR or NVARCHAR | STAT_NAME | Statistic name.<br>1. Slight Silhouette as whole<br>2. Slight Silhouette for each cluster<br>3. ANOVA F-test for each K, if a range of candidate of K is provided instead of single one. |

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| | 2nd column | VARCHAR or NVARCHAR | STAT_VALUE | Statistic value |
| PLACEHOLDER | 1st column | NVARCHAR(256) | PARAM_NAME | Placeholder for future features |
| | 2nd column | INTEGER | INT_VALUE | |
| | 3rd column | DOUBLE | DOUBLE_VALUE | |
| | 4th column | NVARCHAR(1000) | STRING_VALUE | |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_ACCKMEANS_DATA_TBL;
CREATE COLUMN TABLE PAL_ACCKMEANS_DATA_TBL(
    "ID" INTEGER,
    "V000" DOUBLE,
    "V001" VARCHAR(2),
    "V002" INTEGER
);
INSERT INTO PAL_ACCKMEANS_DATA_TBL VALUES (0, 0.5, 'A', 0);
INSERT INTO PAL_ACCKMEANS_DATA_TBL VALUES (1, 1.5, 'A', 0);
INSERT INTO PAL_ACCKMEANS_DATA_TBL VALUES (2, 1.5, 'A', 1);
INSERT INTO PAL_ACCKMEANS_DATA_TBL VALUES (3, 0.5, 'A', 1);
INSERT INTO PAL_ACCKMEANS_DATA_TBL VALUES (4, 1.1, 'B', 1);
INSERT INTO PAL_ACCKMEANS_DATA_TBL VALUES (5, 0.5, 'B', 15);
INSERT INTO PAL_ACCKMEANS_DATA_TBL VALUES (6, 1.5, 'B', 15);
INSERT INTO PAL_ACCKMEANS_DATA_TBL VALUES (7, 1.5, 'B', 16);
INSERT INTO PAL_ACCKMEANS_DATA_TBL VALUES (8, 0.5, 'B', 16);
INSERT INTO PAL_ACCKMEANS_DATA_TBL VALUES (9, 1.2, 'C', 16);
INSERT INTO PAL_ACCKMEANS_DATA_TBL VALUES (10, 15.5, 'C', 15);
INSERT INTO PAL_ACCKMEANS_DATA_TBL VALUES (11, 16.5, 'C', 15);
INSERT INTO PAL_ACCKMEANS_DATA_TBL VALUES (12, 16.5, 'C', 16);
INSERT INTO PAL_ACCKMEANS_DATA_TBL VALUES (13, 15.5, 'C', 16);
INSERT INTO PAL_ACCKMEANS_DATA_TBL VALUES (14, 15.6, 'D', 16);
INSERT INTO PAL_ACCKMEANS_DATA_TBL VALUES (15, 15.5, 'D', 0);
INSERT INTO PAL_ACCKMEANS_DATA_TBL VALUES (16, 16.5, 'D', 0);
INSERT INTO PAL_ACCKMEANS_DATA_TBL VALUES (17, 16.5, 'D', 1);
INSERT INTO PAL_ACCKMEANS_DATA_TBL VALUES (18, 15.5, 'D', 1);
INSERT INTO PAL_ACCKMEANS_DATA_TBL VALUES (19, 15.7, 'A', 1);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL(
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR(1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 0.5, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('GROUP_NUMBER', 4, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('INIT_TYPE', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('DISTANCE_LEVEL',2, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MAX_ITERATION', 100, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('CATEGORY_WEIGHTS', NULL, 0.5, NULL);
```

```
INSERT INTO #PAL_PARAMETER_TBL VALUES ('CATEGORICAL_VARIABLE', NULL, NULL,
'V002');
CALL _SYS_AFL.PAL_ACCELERATED_KMEANS(PAL_ACCKMEANS_DATA_TBL,
#PAL_PARAMETER_TBL, ?, ?, ?, ?, ?);
```

**Expected Result**

RESULT:

| | ID | CLUSTER_ID | DISTANCE | SLIGHT_SILHOUETTE |
|---|---|---|---|---|
| 1 | 0 | 0 | 1.1989378277661475 | 0.006767382594247457 |
| 2 | 1 | 0 | 1.1239378277661478 | 0.0688994169502115 |
| 3 | 2 | 3 | 0.5 | 0.5725058985270607 |
| 4 | 3 | 3 | 0.5 | 0.5982666914900429 |
| 5 | 4 | 0 | 0.6215169943749477 | 0.22994452671895513 |
| 6 | 5 | 0 | 1.0374999999999999 | 0.3083333333333334 |
| 7 | 6 | 0 | 0.9625000000000001 | 0.3583333333333332 |
| 8 | 7 | 0 | 0.8955127018922194 | 0.40299153207185373 |
| 9 | 8 | 0 | 0.9705127018922192 | 0.3529915320718539 |
| 10 | 9 | 0 | 0.8239378277661478 | 0.3133851435282102 |
| 11 | 10 | 1 | 1.03827625302982 | 0.9315554932961414 |
| 12 | 11 | 1 | 1.178276253029824 | 0.9271301851074847 |
| 13 | 12 | 1 | 1.13568542494924 | 0.9295652565098869 |
| 14 | 13 | 1 | 0.995685424949236 | 0.9341649353304501 |
| 15 | 14 | 1 | 0.8496152422706615 | 0.9443590621261144 |
| 16 | 15 | 1 | 0.9956854249492361 | 0.9345482054265087 |
| 17 | 16 | 1 | 1.1356854249492403 | 0.9299500123392912 |
| 18 | 17 | 1 | 1.0896152422706653 | 0.9327692933117768 |
| 19 | 18 | 1 | 0.9496152422706612 | 0.9375545094846393 |
| 20 | 19 | 1 | 0.9155654600401031 | 0.9377166353714216 |

CENTERS:

| | CLUSTER_ID | V000 | V001 | V002 |
|---|---|---|---|---|
| 1 | 0 | 1.0374999999999999 | B | 16 |
| 2 | 1 | 15.929999999999998 | D | 1 |
| 3 | 2 | 10.059999999999999 | A | 1 |
| 4 | 3 | 1 | A | 1 |

MODEL:

| | ROW_INDEX | MODEL_CONTENT |
|---|---|---|
| 1 | 0 | {"Algorithm":"KMEANS","Category":[{"CategoryCol":2,"Categ... |

STATISTICS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | Center 0 Slight Silhouette | 0.25520577507524983 |
| 2 | Center 1 Slight Silhouette | 0.9339313588303716 |
| 3 | Center 2 Slight Silhouette | 0 |
| 4 | Center 3 Slight Silhouette | 0.5853862950085518 |
| 5 | SlightSilhouette | 0.627586618946141 |

PLACEHOLDER: reserved for future use

> **Note**
>
> The wrapper procedure generation approach to call PAL functions is still supported. For more information, refer to the older versions (prior to SAP HANA Platform 2.0 SPS 02) of this document.

### Related Information

K-Means [page 131]

## 5.1.2 Affinity Propagation

Affinity Propagation (AP) is a relatively new clustering algorithm that has been introduced by Brendan J. Frey and Delbert Dueck. The authors described affinity propagation as follows:

*"An algorithm that identifies exemplars among data points and forms clusters of data points around these exemplars. It operates by simultaneously considering all data point as potential exemplars and exchanging messages between data points until a good set of exemplars and clusters emerges."*

One of the most significant advantages of AP-cluster is that the number of clusters is not necessarily predetermined.

## Prerequisites

- No missing or null data in the inputs.
- The data is numeric, not categorical.

## _SYS_AFL.PAL_AFFINITY_PROPAGATION

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<SEED Seed table>` | IN |
| 3 | `<PARAMETER table>` | IN |
| 4 | `<RESULT table>` | OUT |
| 5 | `<STATISTICS table>` | OUT |

### Input Tables

| Table | Column | Reference for Output Table | Data Type | Description |
|---|---|---|---|---|
| DATA | 1st column | ID | INTEGER, VARCHAR, or NVARCHAR | ID |
| | Other columns | FEATURES | INTEGER, DOUBLE, or DECIMAL(p, s) | Attribute data |
| SEED | 1st column | | INTEGER, VARCHAR, or NVARCHAR | Record ID |
| | 2nd column | | INTEGER | Selected seed ID.<br>A subset of the first column of the DATA Table. |

### Parameter Table

Mandatory Parameters

The following parameters are mandatory and must be given a value.

| Name | Data Type | Description |
| --- | --- | --- |
| DISTANCE_METHOD | INTEGER | The method to compute the distance between two points.<br>• 1: Manhattan<br>• 2: Euclidean<br>• 3: Minkowski<br>• 4: Chebyshev<br>• 5: Standardised Euclidean<br>• 6: Cosine |
| CLUSTER_NUMBER | INTEGER | Number of clusters.<br>• 0: does not adjust AP Cluster result.<br>• Non-zero INTEGER: If AP cluster number is bigger than CLUSTER_NUMBER, PAL will merge the result to make the cluster number be the value specified for CLUSTER_NUMBER. |

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description | Dependency |
| --- | --- | --- | --- | --- |
| MAX_ITERATION | INTEGER | 500 | Maximum number of iterations. | |
| CON_ITERATION | INTEGER | 100 | When the clusters keep a steady one for the specified times, the algorithm ends. | |
| DAMP | DOUBLE | 0.9 | Controls the updating velocity.<br>Value range: 0 < DAMP < 1 | |
| PREFERENCE | DOUBLE | 0.5 | Determines the preference.<br>Value range: 0 ≤ PREFERENCE ≤ 1 | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| SEED_RATIO | DOUBLE | 1 | Select a portion of (SEED_RATIO * data_number) the input data as seed, where data_number is the row_size of the input data.<br>Value range: 0 < SEED_RATIO ≤ 1 | Only valid when the seed table is empty. If SEED_RATIO is 1, all the input data will be the seed. |
| TIMES | INTEGER | 1 | The sampling times. | Only valid when the seed table is empty and SEED_RATIO is less than 1. |
| MINKOW_P | INTEGER | 3 | The power of the Minkowski method. | Only valid when DISTANCE_METHOD is 3. |
| THREAD_RATIO | DOUBLE | 0 | Specifies the ratio of total number of threads that can be used by this function. The value range is from 0 to 1, where 0 means only using 1 thread, and 1 means using at most all the currently available threads. Values outside the range will be ignored and this function heuristically determines the number of threads to use. | |

### Output Tables

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| RESULT | 1st column | ID (in input table) data type | ID (in input table) column name | Record ID |
| | 2nd column | INTEGER | CLUSTER_ID | Cluster ID. The range is from 0 to CLUSTER_NUMBER-1. |

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| STATISTICS | 1st column | NVARCHAR | STAT_NAME | Statistic name |
| | 2nd column | DOUBLE | STAT_VALUE | Statistic value |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_AP_DATA_TBL;
CREATE COLUMN TABLE PAL_AP_DATA_TBL(
    ID INTEGER,
    ATTRIB1 DOUBLE,
    ATTRIB2 DOUBLE
);
INSERT INTO PAL_AP_DATA_TBL VALUES(1, 0.10, 0.10);
INSERT INTO PAL_AP_DATA_TBL VALUES(2, 0.11, 0.10);
INSERT INTO PAL_AP_DATA_TBL VALUES(3, 0.10, 0.11);
INSERT INTO PAL_AP_DATA_TBL VALUES(4, 0.11, 0.11);
INSERT INTO PAL_AP_DATA_TBL VALUES(5, 0.12, 0.11);
INSERT INTO PAL_AP_DATA_TBL VALUES(6, 0.11, 0.12);
INSERT INTO PAL_AP_DATA_TBL VALUES(7, 0.12, 0.12);
INSERT INTO PAL_AP_DATA_TBL VALUES(8, 0.12, 0.13);
INSERT INTO PAL_AP_DATA_TBL VALUES(9, 0.13, 0.12);
INSERT INTO PAL_AP_DATA_TBL VALUES(10, 0.13, 0.13);
INSERT INTO PAL_AP_DATA_TBL VALUES(11, 0.13, 0.14);
INSERT INTO PAL_AP_DATA_TBL VALUES(12, 0.14, 0.13);
INSERT INTO PAL_AP_DATA_TBL VALUES(13, 10.10, 10.10);
INSERT INTO PAL_AP_DATA_TBL VALUES(14, 10.11, 10.10);
INSERT INTO PAL_AP_DATA_TBL VALUES(15, 10.10, 10.11);
INSERT INTO PAL_AP_DATA_TBL VALUES(16, 10.11, 10.11);
INSERT INTO PAL_AP_DATA_TBL VALUES(17, 10.11, 10.12);
INSERT INTO PAL_AP_DATA_TBL VALUES(18, 10.12, 10.11);
INSERT INTO PAL_AP_DATA_TBL VALUES(19, 10.12, 10.12);
INSERT INTO PAL_AP_DATA_TBL VALUES(20, 10.12, 10.13);
INSERT INTO PAL_AP_DATA_TBL VALUES(21, 10.13, 10.12);
INSERT INTO PAL_AP_DATA_TBL VALUES(22, 10.13, 10.13);
INSERT INTO PAL_AP_DATA_TBL VALUES(23, 10.13, 10.14);
INSERT INTO PAL_AP_DATA_TBL VALUES(24, 10.14, 10.13);
DROP TABLE PAL_AP_SEED_TBL;
CREATE COLUMN TABLE PAL_AP_SEED_TBL(
    ID INTEGER,
    SEED INTEGER
);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL(
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR(1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES('THREAD_RATIO', NULL, 0.3, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES('MAX_ITERATION', 500, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES('CON_ITERATION', 100, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES('DAMP', NULL, 0.9, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES('PREFERENCE', NULL, 0.5, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES('DISTANCE_METHOD', 2, NULL, NULL);
```

```
INSERT INTO #PAL_PARAMETER_TBL VALUES('CLUSTER_NUMBER', 0, NULL, NULL);
CALL _SYS_AFL.PAL_AFFINITY_PROPAGATION(PAL_AP_DATA_TBL, PAL_AP_SEED_TBL,
"#PAL_PARAMETER_TBL", ?, ?);
```

**Expected Result**

RESULT:

| | ID | CLUSTER_ID |
|---|---|---|
| 1 | 1 | 0 |
| 2 | 2 | 0 |
| 3 | 3 | 0 |
| 4 | 4 | 0 |
| 5 | 5 | 0 |
| 6 | 6 | 0 |
| 7 | 7 | 0 |
| 8 | 8 | 0 |
| 9 | 9 | 0 |
| 10 | 10 | 0 |
| 11 | 11 | 0 |
| 12 | 12 | 0 |
| 13 | 13 | 1 |
| 14 | 14 | 1 |
| 15 | 15 | 1 |
| 16 | 16 | 1 |
| 17 | 17 | 1 |
| 18 | 18 | 1 |
| 19 | 19 | 1 |
| 20 | 20 | 1 |
| 21 | 21 | 1 |
| 22 | 22 | 1 |
| 23 | 23 | 1 |
| 24 | 24 | 1 |

STATISTICS:

| STAT_NAME | STAT_VALUE |
| --- | --- |

> **Note**
>
> The wrapper procedure generation approach to call PAL functions is still supported. For more information, refer to the older versions (prior to SAP HANA Platform 2.0 SPS 02) of this document.

## 5.1.3 Agglomerate Hierarchical Clustering

This algorithm is a widely used clustering method which can find natural groups within a set of data. The idea is to group the data into a hierarchy or a binary tree of the subgroups.

A hierarchical clustering can be either agglomerate or divisive, depending on the method of hierarchical decomposition.

The implementation in PAL follows the agglomerate approach, which merges the clusters with a bottom-up strategy. Initially, each data point is considered as an own cluster. The algorithm iteratively merges two clusters based on the dissimilarity measure in a greedy manner and forms a larger cluster. Therefore, the input data must be numeric and a measure of dissimilarity between sets of data is required, which is achieved by using the following two parameters:

- An appropriate metric (a measure of distance between pairs of groups)
- A linkage criterion which specifies the distances between groups

An advantage of hierarchical clustering is that it does not require the number of clusters to be specified as the input. And the hierarchical structure can also be used for data summarization and visualization.

The agglomerate hierarchical clustering functions in PAL now supports eight kinds of appropriate metrics and seven kinds of linkage criteria.

**Support for Category Attributes**

If the input data has category attributes, you must set the DISTANCE_FUNC parameter to Gower Distance to support calculating the distance matrix. Gower Distance is calculated in the following way.

Suppose that the items $X_i$ and $X_j$ have K attributes, the distance between $X_i$ and $X_j$ is:

$$s(i,j) = \sum_k (S_{ijk} * W_k) / \sum_k W_k$$

For continuous attributes,

$S_{ijk}=(X_{ik}-X_{jk})/R_k$

$W_k=1$

$R_k$ is the range of values for the $k_{th}$ variable; $W_k$ is set by user and the default is 1.

For category attributes,

If $X_{ik}=X_{jk}$: $S_{ijk}=0$

Other cases: $S_{ijk}=1$

## Prerequisites

- The first column of the input data is an ID column and the other columns are of INTEGER, DOUBLE, VARCHAR, or NVARCHAR data type.
- The input data does not contain null value. The algorithm will issue errors when encountering null values.

## _SYS_AFL.PAL_HIERARCHICAL_CLUSTERING

This is a clustering function using the agglomerate hierarchical clustering algorithm.

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<COMBINE_PROCESS table>` | OUT |
| 4 | `<RESULT table>` | OUT |

### Input Table

| Table | Column | Reference for Output Table | Data Type | Description |
|---|---|---|---|---|
| DATA | 1st column | ID | VARCHAR, NVARCHAR, or INTEGER<br>**Note**<br>For precalculated distance, the data table has 3 columns. The first column is the ID of the first point. | ID |

| Table | Column | Reference for Output Table | Data Type | Description |
|---|---|---|---|---|
| | Other columns | | INTEGER, DOUBLE, DECIMAL(p, s), VARCHAR, or NVARCHAR<br>**Note**<br>For precalculated distance, the second column is the ID of the second point. The third column is of type DOUBLE. It's the distance between the first and the second point. | Attribute data |

**Parameter Table**

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| CLUSTER_NUM | INTEGER | 1 | Number of clusters after agglomerate hierarchical clustering algorithm.<br>Value range: between 1 and the initial number of input data | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| DISTANCE_FUNC | INTEGER | 8 | Measure of distance between two clusters.<br>• 1: Manhattan Distance<br>• 2: Euclidean Distance<br>• 3: Minkowski Distance<br>• 4: Chebyshev Distance<br>• 6: Cosine<br>• 7: Pearson Correlation<br>• 8: Squared Euclidean Distance<br>• 9: Jaccard Distance<br>• 10: Gower Distance<br>• 11: Precalculated Distance<br>**Note**<br>• For Jaccard Distance, non-zero input data will be treated as 1, and zero input data will be treated as 0.<br>Jaccard Distance = $(M_{01} + M_{10}) / (M_{11} + M_{01} + M_{10})$<br>• Only Gower Distance supports category attributes. | When CLUSTER_METHOD is 5, 6, or 7, this parameter must be set to 8. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| CLUSTER_METHOD | INTEGER | 5 | Linkage type between two clusters.<br>• 1: Nearest Neighbor (single linkage)<br>• 2: Furthest Neighbor (complete linkage)<br>• 3: Group Average (UPGMA)<br>• 4: Weighted Average (WPGMA)<br>• 5: Centroid Clustering<br>• 6: Median Clustering<br>• 7: Ward Method<br>**Note**<br>• For clustering method 5, 6, or 7, the corresponding distance function (DISTANCE_FUNC) must be set to 8.<br>• The initial cluster distances in Ward's method are defined to be the squared Euclidean distance between points. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| THREAD_RATIO | DOUBLE | 0 | Specifies the ratio of total number of threads that can be used by this function. The value range is from 0 to 1, where 0 means only using 1 thread, and 1 means using at most all the currently available threads. Values outside the range will be ignored and this function heuristically determines the number of threads to use. | |
| DISTANCE_DIMENSION | DOUBLE | 3 | Distance dimension can be set if DISTANCE_FUNC is set to 3 (Minkowski Distance). The value should be no less than 1. | Only valid when DISTANCE_FUNC is 3. |
| NORMALIZE_TYPE | INTEGER | 0 | Normalization type:<br>• 0: does nothing<br>• 1: Z score standardize<br>• 2: transforms to new range: -1 to 1<br>• 3: transforms to new range: 0 to 1 | |
| CATEGORICAL_VARIABLE | VARCHAR | No default value | Indicates whether or not a column data is actually corresponding to a category variable even the data type of this column is INTEGER.<br>By default, 'VARCHAR' or 'NVARCHAR' is category variable, and 'INTEGER' or 'DOUBLE' is continuous variable. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| CATEGORY_WEIGHTS | DOUBLE | 1 | Represents the weight of category columns. | |
| EXCLUDED_FEATURE | VARCHAR | No default value | Specifies the feature(s) to be excluded from training. | |

### Output Tables

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| COMBINE_PROCESS | 1st column | INTEGER | STAGE | Cluster stage. |
| | 2nd column | ID (in input table) data type | LEFT_ + ID (in input table) column name | One of the clusters that is to be combined in one combine stage, name as its row number in the input data table.<br>After the combining, the new cluster is named after the left one. |
| | 3rd column | ID (in input table) data type | RIGHT_ + ID (in input table) column name | The other cluster to be combined in the same combine stage, named as its row number in the input data table. |
| | 4th column | DOUBLE | DISTANCE | Distance between the two combined clusters. |
| RESULT | 1st column | ID (in input table) data type | ID (in input table) column name | ID of the input data. |
| | 2nd column | INTEGER | CLUSTER_ID | Cluster number after applying the hierarchical agglomerate algorithm. |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

### Example 1

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_HIERARCHICAL_CLUSTERING_DATA_TBL;
CREATE COLUMN TABLE PAL_HIERARCHICAL_CLUSTERING_DATA_TBL ("POINT" NVARCHAR(100),
"X1" DOUBLE, "X2" DOUBLE ,  "X3" INTEGER);
INSERT INTO PAL_HIERARCHICAL_CLUSTERING_DATA_TBL VALUES ('0', 0.5, 0.5, 1);
INSERT INTO PAL_HIERARCHICAL_CLUSTERING_DATA_TBL VALUES ('1', 1.5, 0.5, 2);
INSERT INTO PAL_HIERARCHICAL_CLUSTERING_DATA_TBL VALUES ('2', 1.5, 1.5, 2);
INSERT INTO PAL_HIERARCHICAL_CLUSTERING_DATA_TBL VALUES ('3', 0.5, 1.5, 2);
INSERT INTO PAL_HIERARCHICAL_CLUSTERING_DATA_TBL VALUES ('4', 1.1, 1.2, 2);
INSERT INTO PAL_HIERARCHICAL_CLUSTERING_DATA_TBL VALUES ('5', 0.5, 15.5, 2);
INSERT INTO PAL_HIERARCHICAL_CLUSTERING_DATA_TBL VALUES ('6', 1.5, 15.5, 3);
INSERT INTO PAL_HIERARCHICAL_CLUSTERING_DATA_TBL VALUES ('7', 1.5, 16.5, 3);
INSERT INTO PAL_HIERARCHICAL_CLUSTERING_DATA_TBL VALUES ('8', 0.5, 16.5, 3);
INSERT INTO PAL_HIERARCHICAL_CLUSTERING_DATA_TBL VALUES ('9', 1.2, 16.1, 3);
INSERT INTO PAL_HIERARCHICAL_CLUSTERING_DATA_TBL VALUES ('10', 15.5, 15.5, 3);
INSERT INTO PAL_HIERARCHICAL_CLUSTERING_DATA_TBL VALUES ('11', 16.5, 15.5, 4);
INSERT INTO PAL_HIERARCHICAL_CLUSTERING_DATA_TBL VALUES ('12', 16.5, 16.5, 4);
INSERT INTO PAL_HIERARCHICAL_CLUSTERING_DATA_TBL VALUES ('13', 15.5, 16.5, 4);
INSERT INTO PAL_HIERARCHICAL_CLUSTERING_DATA_TBL VALUES ('14', 15.6, 16.2, 4);
INSERT INTO PAL_HIERARCHICAL_CLUSTERING_DATA_TBL VALUES ('15', 15.5, 0.5, 4);
INSERT INTO PAL_HIERARCHICAL_CLUSTERING_DATA_TBL VALUES ('16', 16.5, 0.5, 1);
INSERT INTO PAL_HIERARCHICAL_CLUSTERING_DATA_TBL VALUES ('17', 16.5, 1.5, 1);
INSERT INTO PAL_HIERARCHICAL_CLUSTERING_DATA_TBL VALUES ('18', 15.5, 1.5, 1);
INSERT INTO PAL_HIERARCHICAL_CLUSTERING_DATA_TBL VALUES ('19', 15.7, 1.6, 1);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL ("PARAM_NAME"
VARCHAR(256), "INT_VALUE" INTEGER, "DOUBLE_VALUE" DOUBLE, "STRING_VALUE"
VARCHAR(1000));
INSERT INTO #PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 0.5, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('CLUSTER_NUM', 4, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('CLUSTER_METHOD', 4, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('DISTANCE_FUNC', 10, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('DISTANCE_DIMENSION', NULL, 3, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('NORMALIZE_TYPE', 0, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('CATEGORICAL_VARIABLE', NULL, NULL, 'X3');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('CATEGORY_WEIGHTS', NULL, 0.1, NULL);
CALL _SYS_AFL.PAL_HIERARCHICAL_CLUSTERING (PAL_HIERARCHICAL_CLUSTERING_DATA_TBL,
#PAL_PARAMETER_TBL, ?, ?);
```

### Expected Result

COMBINE_PROCESS:

| | STAGE | LEFT_POINT | RIGHT_POINT | DISTANCE |
|---|---|---|---|---|
| 1 | 1 | 18 | 19 | 0.0187 |
| 2 | 2 | 13 | 14 | 0.025 |
| 3 | 3 | 7 | 9 | 0.0437 |
| 4 | 4 | 2 | 4 | 0.0438 |
| 5 | 5 | 2 | 3 | 0.0594 |
| 6 | 6 | 17 | 18 | 0.0594 |
| 7 | 7 | 6 | 7 | 0.0594 |
| 8 | 8 | 11 | 12 | 0.0625 |
| 9 | 9 | 11 | 13 | 0.0906 |
| 10 | 10 | 16 | 17 | 0.0922 |
| 11 | 11 | 6 | 8 | 0.0953 |
| 12 | 12 | 1 | 2 | 0.0953 |
| 13 | 13 | 0 | 1 | 0.1727 |
| 14 | 14 | 5 | 6 | 0.1727 |
| 15 | 15 | 10 | 11 | 0.175 |
| 16 | 16 | 15 | 16 | 0.1805 |
| 17 | 17 | 0 | 15 | 1.0381 |
| 18 | 18 | 5 | 10 | 1.0425 |
| 19 | 19 | 0 | 5 | 1.5146 |

RESULT:

| | POINT | CLUSTER_ID |
|---|---|---|
| 1 | 0 | 1 |
| 2 | 1 | 1 |
| 3 | 2 | 1 |
| 4 | 3 | 1 |
| 5 | 4 | 1 |
| 6 | 5 | 2 |
| 7 | 6 | 2 |
| 8 | 7 | 2 |
| 9 | 8 | 2 |
| 10 | 9 | 2 |
| 11 | 10 | 3 |
| 12 | 11 | 3 |
| 13 | 12 | 3 |
| 14 | 13 | 3 |
| 15 | 14 | 3 |
| 16 | 15 | 4 |
| 17 | 16 | 4 |
| 18 | 17 | 4 |
| 19 | 18 | 4 |
| 20 | 19 | 4 |

### Example 2

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_PRECALCULATED_DISTANCE;
CREATE COLUMN TABLE PAL_PRECALCULATED_DISTANCE (
    "LEFT_POINT"  NVARCHAR(5),
    "RIGHT_POINT" NVARCHAR(5),
    "DISTANCE"    DOUBLE
);
INSERT INTO PAL_PRECALCULATED_DISTANCE VALUES('0', '1',  0.707106781);
INSERT INTO PAL_PRECALCULATED_DISTANCE VALUES('0', '2', 12.74754878 );
INSERT INTO PAL_PRECALCULATED_DISTANCE VALUES('0', '3', 12.72792206 );
INSERT INTO PAL_PRECALCULATED_DISTANCE VALUES('1', '2', 12.72792206 );
INSERT INTO PAL_PRECALCULATED_DISTANCE VALUES('1', '3', 12.74754878 );
```

```
INSERT INTO PAL_PRECALCULATED_DISTANCE VALUES('2', '3',  0.707106781);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL ("PARAM_NAME"
VARCHAR(256), "INT_VALUE" INTEGER, "DOUBLE_VALUE" DOUBLE, "STRING_VALUE"
VARCHAR(1000));
INSERT INTO #PAL_PARAMETER_TBL VALUES ('CLUSTER_NUM',     2, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('DISTANCE_FUNC',  11, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('CLUSTER_METHOD',  1, NULL, NULL);
CALL _SYS_AFL.PAL_HIERARCHICAL_CLUSTERING (PAL_PRECALCULATED_DISTANCE,
#PAL_PARAMETER_TBL, ?, ?);
```

**Expected Result**

COMBINE_PROCESS

| | STAGE | LEFT_LEFT_POINT | RIGHT_LEFT_POINT | DISTANCE |
|---|---|---|---|---|
| 1 | 1 | 0 | 1 | 0.7071 |
| 2 | 2 | 2 | 3 | 0.7071 |
| 3 | 3 | 0 | 2 | 12.7279 |

RESULT

| | LEFT_POINT | CLUSTER_ID |
|---|---|---|
| 1 | 0 | 1 |
| 2 | 1 | 1 |
| 3 | 2 | 2 |
| 4 | 3 | 2 |

> **Note**
>
> The wrapper procedure generation approach to call PAL functions is still supported. For more information, refer to the older versions (prior to SAP HANA Platform 2.0 SPS 02) of this document.

## 5.1.4 Anomaly Detection

Anomaly detection is used to find existing data objects that do not comply with the general behavior or model of the data. Such data objects, which are grossly different from or inconsistent with the remaining set of data, are called anomalies or outliers. Sometimes anomalies are also referred to as discordant observations, exceptions, aberrations, surprises, peculiarities or contaminants in different application domains.

Anomalies in data can translate to significant (and often critical) actionable information in a wide variety of application domains. For example, an anomalous traffic pattern in a computer network could mean that a hacked computer is sending out sensitive data to an unauthorized destination. An anomalous MRI image may indicate presence of malignant tumors. Anomalies in credit card transaction data could indicate credit card theft or identity theft. Anomalous readings from a space craft sensor could signify a fault in some component of the space craft.

PAL uses k-means to realize anomaly detection in two steps:

1. Use k-means to group the origin data into k clusters.

2. Identify some points that are far from all cluster centers as anomalies.

## prerequisites

- The input data contains an ID column and the other columns are of integer or double data type.
- The input data does not contain null value. The algorithm will issue errors when encountering null values.

## _SYS_AFL.PAL_ANOMALY_DETECTION

This function finds the outliers of the data according to the clusters clustered by k-means.

**Overview of All Tables**

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<RESULT table>` | OUT |
| 4 | `<Statistics table>` | OUT |
| 5 | `<Centers table>` | OUT |

**Input Table**

| Table | Column | Data Type | Description |
|---|---|---|---|
| DATA | 1st column | INTEGER, BIGINT, VARCHAR, or NVARCHAR | ID |
| | Other columns | INTEGER, DOUBLE, or DECIMAL(p, s) | Attribute data |

**Parameter Table**

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description |
| --- | --- | --- | --- |
| GROUP_NUMBER | INTEGER | No default value | Number of groups (k).<br>If k is not specified, the G-means method will be used to determine the number of clusters. |
| DISTANCE_LEVEL | INTEGER | 2 | Computes the distance between the item and the cluster center.<br>• 1: Manhattan distance<br>• 2: Euclidean distance<br>• 3: Minkowski distance |
| OUTLIER_PERCENTAGE | DOUBLE | 0.1 | Indicates the proportion of anomalies in the source data. |
| OUTLIER_DEFINE | INTEGER | 2 | Specifies which point should be defined as outlier:<br>• 1: Max distance between the point and the center it belongs to<br>• 2: Max sum distance from the point to all centers |
| MAX_ITERATION | INTEGER | 100 | Maximum number of iterations. |
| INIT_TYPE | INTEGER | 4 | Controls how the initial centers are selected:<br>• 1: First k observations<br>• 2: Random with replacement<br>• 3: Random without replacement<br>• 4: Patent of selecting the init center (US 6,882,998 B1) |

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| NORMALIZATION | INTEGER | 0 | Normalization type:<br>• 0: No<br>• 1: Yes. For each point X (x1,x2,...,xn), the normalized value will be X'(x1/S,x2/S,...,xn/S), where S = \|x1\|+\|x2\|+...\|xn\|.<br>• 2: For each column C, get the min and max value of C, and then C[i] = (C[i]-min)/(max-min). |
| THREAD_NUMBER | INTEGER | 1 | Number of threads. |
| EXIT_THRESHOLD | DOUBLE | 0.000000001 | Threshold (actual value) for exiting the iterations. |

### Output Tables

| Table | Column | Column Data Type | Description |
|---|---|---|---|
| Outliers | 1st column | INTEGER, BIGINT, VARCHAR, or NVARCHAR | ID |
| | Other columns | INTEGER or DOUBLE | Coordinates of outliers |
| Statistics | 1st column | INTEGER, BIGINT, VARCHAR, or NVARCHAR | ID |
| | 2nd column | INTEGER | ID of cluster center |
| | 3rd column | DOUBLE | distance value |
| Centers | 1st column | INTEGER | ID of cluster center |
| | Other columns | INTEGER or DOUBLE | Coordinates of centers |

### Examples

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFLPM_CREATOR_ERASER_EXECUTE` role;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_AD_DATA_TBL;
```

```
CREATE COLUMN TABLE PAL_AD_DATA_TBL(
    "ID" INTEGER,
    "V000" DOUBLE,
    "V001" DOUBLE
);
INSERT INTO PAL_AD_DATA_TBL VALUES (0 , 0.5, 0.5);
INSERT INTO PAL_AD_DATA_TBL VALUES (1 , 1.5, 0.5);
INSERT INTO PAL_AD_DATA_TBL VALUES (2 , 1.5, 1.5);
INSERT INTO PAL_AD_DATA_TBL VALUES (3 , 0.5, 1.5);
INSERT INTO PAL_AD_DATA_TBL VALUES (4 , 1.1, 1.2);
INSERT INTO PAL_AD_DATA_TBL VALUES (5 , 0.5, 15.5);
INSERT INTO PAL_AD_DATA_TBL VALUES (6 , 1.5, 15.5);
INSERT INTO PAL_AD_DATA_TBL VALUES (7 , 1.5, 16.5);
INSERT INTO PAL_AD_DATA_TBL VALUES (8 , 0.5, 16.5);
INSERT INTO PAL_AD_DATA_TBL VALUES (9 , 1.2, 16.1);
INSERT INTO PAL_AD_DATA_TBL VALUES (10, 15.5, 15.5);
INSERT INTO PAL_AD_DATA_TBL VALUES (11, 16.5, 15.5);
INSERT INTO PAL_AD_DATA_TBL VALUES (12, 16.5, 16.5);
INSERT INTO PAL_AD_DATA_TBL VALUES (13, 15.5, 16.5);
INSERT INTO PAL_AD_DATA_TBL VALUES (14, 15.6, 16.2);
INSERT INTO PAL_AD_DATA_TBL VALUES (15, 15.5, 0.5);
INSERT INTO PAL_AD_DATA_TBL VALUES (16, 16.5, 0.5);
INSERT INTO PAL_AD_DATA_TBL VALUES (17, 16.5, 1.5);
INSERT INTO PAL_AD_DATA_TBL VALUES (18, 15.5, 1.5);
INSERT INTO PAL_AD_DATA_TBL VALUES (19, 15.7, 1.6);
INSERT INTO PAL_AD_DATA_TBL VALUES (20, -1.0, -1.0);
DROP TABLE #PAL_CONTROL_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_CONTROL_TBL(
    "NAME" VARCHAR (100),
    "INTARGS" INTEGER,
    "DOUBLEARGS" DOUBLE,
    "STRINGARGS" VARCHAR(100)
);
INSERT INTO #PAL_CONTROL_TBL VALUES ('THREAD_NUMBER',2,null,null);
INSERT INTO #PAL_CONTROL_TBL VALUES ('GROUP_NUMBER',4,null,null);
INSERT INTO #PAL_CONTROL_TBL VALUES ('INIT_TYPE',4,null,null);
INSERT INTO #PAL_CONTROL_TBL VALUES ('DISTANCE_LEVEL',2,null,null);
INSERT INTO #PAL_CONTROL_TBL VALUES ('MAX_ITERATION',100,null,null);
CALL _SYS_AFL.PAL_ANOMALY_DETECTION(PAL_AD_DATA_TBL, #PAL_CONTROL_TBL, ?, ?, ?);
```

### Expected Results

Outliers:

| V<br>O<br>D | V<br>O<br>1 |
|---|---|
| 2<br>0 | -<br>1 |
| 1<br>6<br>.<br>5 | 0<br>.<br>5 |

Statistics:

| CLUSTER_ID | SCORE |
|---|---|
| 20 | 60.619864431293074 |
| 36 | 054.110424127543936 |

Centers:

| CLUSTER_ID | V001 |
|---|---|
| 0 | 0.683333333333332 | 0.7000000001 |
| 1 | 15.919999999998 | 16.04 |
| 2 | 2.04 | 16.02 |

| CLUSTER_ID | V001 |
|---|---|
| 35.940000000000001 | 1.1199999999999999 |

## 5.1.5 Cluster Assignment

Cluster assignment is used to assign data to the clusters that were previously generated by some clustering methods such as K-means, DBSCAN (Density-Based Spatial Clustering of Applications with Noise), SOM (Self-Organizing Maps), and GMM (Gaussian Mixture Model).

This algorithm requires that the corresponding clustering procedures save cluster information, or cluster model, which also includes the control parameters for consistency. It assumes that new data is from similar distribution as previous data, and will not update the cluster information.

For clusters generated by K-means, distances between new data and cluster centers are calculated, and then the new data is assigned to the cluster with the smallest distance.

For clusters generated by DBSCAN, all core objects are stored. For each piece of new data, the algorithm tries to find a core object in some formed cluster whose distance is less than the value of the RADIUS parameter. If such a core object is found, the new data is then assigned to the corresponding cluster, otherwise it is assigned to cluster -1, indicating that it is noise. It is possible that a piece of data can belong to more than one cluster, which can be further divided into the following two cases:

- If the number of core objects whose distances to the new data is less than the MINPTS parameter value, meaning that the new data is a border object, the new data is assigned to the cluster where there is a core object having the smallest distance to the new data.

- If the number of core objects whose distances to the new data is not less than MINPTS, which means the new data is also a core object, it is then assigned to cluster -2, indicating that it belongs to more than one cluster. In this case, re-running the DBSCAN function is highly suggested.

For GMM, the probabilities of each new data belong to each cluster are calculated and outputted in the assignment table.

For clusters generated by SOM, similar to K-means, the distances between new data and weight vector are calculated, and the new data is then assigned to the cluster with the smallest distance.

## Prerequisites

- No missing or null data in the inputs.
- Data types must be identical to those in the corresponding clustering procedure.

## _SYS_AFL.PAL_CLUSTER_ASSIGNMENT

This function directly assigns data to clusters based on the previous cluster model, without running clustering procedure thoroughly. It currently supports the K-means, DBSCAN, and SOM clustering methods.

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<MODEL table>` | IN |
| 3 | `<PARAMETER table>` | IN |
| 4 | `<ASSIGNMENT table>` | OUT |

### Input Tables

| Table | Column | Reference for Output Table | Data Type | Description | Constraint |
|---|---|---|---|---|---|
| DATA | 1st column | ID | INTEGER, VARCHAR, or NVARCHAR | Data ID | This must be the first column. |

| Table | Column | Reference for Output Table | Data Type | Description | Constraint |
|---|---|---|---|---|---|
| | Other columns | | INTEGER, DOUBLE, DECIMAL(p, s), VARCHAR, or NVARCHAR<br>**Note**: For SOM clustering method, attribute data only supports numeric values,hence its column data type can only be INTEGER or DOUBLE. | Attribute data | Should be identical to the model.<br>As for SOM, attribute data supports only numeric, hence its column data types exclude VARCHAR and NVARCHAR. |

K-means, DBSCAN, and SOM:

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| MODEL | 1st column | INTEGER | Row index | This must be the first column. |
| | 2nd column | VARCHAR or NVARCHAR | Model content in JSON format | |

GMM:

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| MODEL | 1st column | INTEGER | Row index | This must be the first column. |
| | 2nd column | INTEGER | Cluster ID | |
| | 3rd column | VARCHAR or NVARCHAR | Content of the corresponding cluster in JSON format | |

### Parameter Table

None.

### Output Table

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| ASSIGNMENT | 1st column | ID (in input table) data type | ID (in input table) column name | Data ID |
| | 2nd column | INTEGER | CLUSTER_ID | The assigned cluster ID |
| | 3rd column | DOUBLE | DISTANCE | Distance between a given point and the cluster center (K-means), nearest core object (DBSCAN), or weight vector (SOM).<br>Or probability of a given point belongs to the corresponding cluster (GMM). |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

**For DBSCAN:**

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_DBSCAN_DATA_TBL;
CREATE COLUMN TABLE PAL_DBSCAN_DATA_TBL (
    "ID" INTEGER,
    "V1" DOUBLE,
    "V2" DOUBLE,
    "V3" VARCHAR(10)
);
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(1, 0.10, 0.10,  'B');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(2, 0.11, 0.10,  'A');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(3, 0.10, 0.11,  'C');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(4, 0.11, 0.11,  'B');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(5, 0.12, 0.11,  'A');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(6, 0.11, 0.12,  'E');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(7, 0.12, 0.12,  'A');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(8, 0.12, 0.13,  'C');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(9, 0.13, 0.12,  'D');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(10, 0.13, 0.13,  'D');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(11, 0.13, 0.14,  'A');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(12, 0.14, 0.13,  'C');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(13, 10.10, 10.10,  'A');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(14, 10.11, 10.10,  'F');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(15, 10.10, 10.11,  'E');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(16, 10.11, 10.11,  'E');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(17, 10.11, 10.12,  'A');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(18, 10.12, 10.11,  'B');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(19, 10.12, 10.12,  'B');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(20, 10.12, 10.13,  'D');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(21, 10.13, 10.12,  'F');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(22, 10.13, 10.13,  'A');
```

```
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(23, 10.13, 10.14,  'A');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(24, 10.14, 10.13,  'D');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(25, 4.10, 4.10,  'A');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(26, 7.11, 7.10,  'C');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(27, -3.10, -3.11,  'C');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(28, 16.11, 16.11,  'A');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(29, 20.11, 20.12,  'C');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(30, 15.12, 15.11,  'A');
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR(100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR(100)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 0.2, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('AUTO_PARAM', NULL, NULL, 'true');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('DISTANCE_METHOD', 1, NULL, NULL);
DROP TABLE PAL_DBSCAN_MODEL_TBL;
CREATE COLUMN TABLE PAL_DBSCAN_MODEL_TBL (
    "ROW_INDEX" INTEGER,
    "MODEL_CONTENT" NVARCHAR(5000)
);
CALL _SYS_AFL.PAL_DBSCAN(PAL_DBSCAN_DATA_TBL, "#PAL_PARAMETER_TBL", ?,
PAL_DBSCAN_MODEL_TBL, ?, ?) WITH OVERVIEW;
DROP TABLE PAL_DBSCAN_DATA_TBL;
CREATE COLUMN TABLE PAL_DBSCAN_DATA_TBL (
    "ID" INTEGER,
    "V1" DOUBLE,
    "V2" DOUBLE,
    "V3" VARCHAR(10)
);
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES (1, 0.10, 0.10, 'B');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES (2, 0.10, -2.10, 'A');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES (3, 3.10, 0.10, 'B');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES (4, 10.10, 10.10, 'D');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES (5, 3.10, -0.50, 'C');
CALL _SYS_AFL.PAL_CLUSTER_ASSIGNMENT(PAL_DBSCAN_DATA_TBL, PAL_DBSCAN_MODEL_TBL,
"#PAL_PARAMETER_TBL",  ?);
```

**Expected Result**

ASSIGNMENT:

| | ID | CLUSTER_ID | DISTANCE |
|---|---|---|---|
| 1 | 1 | 0 | 0 |
| 2 | 2 | 0 | 2.21 |
| 3 | 3 | -1 | 3 |
| 4 | 4 | 1 | 0.050000... |
| 5 | 5 | -1 | 3.59 |

> **Note**
>
> The wrapper procedure generation approach to call PAL functions is still supported. For more information, refer to the older versions (prior to SAP HANA Platform 2.0 SPS 02) of this document.

## Related Information

K-Means [page 131]
DBSCAN [page 111]
Self-Organizing Maps [page 157]

# 5.1.6 DBSCAN

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is a density-based data clustering algorithm. It finds a number of clusters starting from the estimated density distribution of corresponding nodes.

DBSCAN requires two parameters: scan radius (eps) and the minimum number of points required to form a cluster (minPts). The algorithm starts with an arbitrary starting point that has not been visited. This point's eps-neighborhood is retrieved, and if the number of points it contains is equal to or greater than minPts, a cluster is started. Otherwise, the point is labeled as noise. These two parameters are very important and are usually determined by user.

PAL provides a method to automatically determine these two parameters. You can choose to specify the parameters by yourself or let the system determine them for you.

## Prerequisites

- No missing or null data in the inputs.

## _SYS_AFL.PAL_DBSCAN

This function clusters data based on density, and is able to identify noises.

**Overview of All Tables**

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<RESULT table>` | OUT |
| 4 | `<MODEL table>` | OUT |
| 5 | `<STATISTICS table>` | OUT |

| Position | Table Name | Parameter Type |
|---|---|---|
| 6 | `<PLACEHOLDER table>` | OUT |

### Input Table

| Table | Column | Reference | Data Type | Description |
|---|---|---|---|---|
| DATA | 1st column | ID | INTEGER, BIGINT, VARCHAR, or NVARCHAR | Data ID<br>This must be the first column. |
| | Other columns | FEATURES | INTEGER, DOUBLE, DECIMAL(p, s), VARCHAR, or NVARCHAR | Attribute data |

### Parameter Table

Mandatory Parameters

The following parameters are mandatory and must be given a value.

| Name | Data Type | Description | Dependency |
|---|---|---|---|
| AUTO_PARAM | VARCHAR or NVARCHAR | Specifies whether the MINPTS and RADIUS parameters are determined automatically or by user. This parameter is case sensitive.<br>• true: automatically determines the parameters<br>• false: uses parameter values provided by user | |
| MINPTS | INTEGER | Specifies the minimum number of points required to form a cluster. | Only valid when AUTO_PARAM is false. |
| RADIUS | DOUBLE | Specifies the scan radius (eps). | Only valid when AUTO_PARAM is false. |

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| THREAD_RATIO | DOUBLE | -1 | The ratio of available threads.<br>• 0: single thread<br>• 0~1: percentage<br>• Others: heuristically determined | |
| DISTANCE_METHOD | INTEGER | 2 | Specifies the method to compute the distance between two points.<br>• 1: Manhattan<br>• 2: Euclidean<br>• 3: Minkowski<br>• 4: Chebyshev<br>• 5: Standardized Euclidean<br>• 6: Cosine | |
| MINKOW_P | INTEGER | 3 | Specifies the power of the Minkowski method. | Only valid when DISTANCE_METHOD is 3. |
| CATEGORICAL_VARIABLE | VARCHAR or NVARCHAR | No default value | Indicates whether or not a column data is actually corresponding to a category variable even the data type of this column is INTEGER.<br>By default, VARCHAR or NVARCHAR is category variable, and INTEGER or DOUBLE is continuous variable. | |
| STRING_VARIABLE | NVARCHAR | No default value | Indicates a string column storing not categorical data. Levenshtein distance is used to calculate similarity between two strings. Ignored if it is not a string column. | If DISTANCE_LEVEL is set to 6 (COSINE), this parameter is not supported. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| VARIABLE_WEIGHT | NVARCHAR for variable name and DOUBLE for weight value | No default value | Specifies the weight of a variable participating in distance calculation. The value must be equal to or greater than 0. Defaults to 1 for variables not specified. | |
| CATEGORY_WEIGHTS | DOUBLE | 0.707 | Represents the weight of categorical variable (γ). | |
| METHOD | INTEGER | 1 | Specifies the searching method.<br>• 0: Brute force searching<br>• 1: KD-tree searching | |
| SAVE_MODEL | INTEGER | 1 | Indicates whether model should be saved.<br>• 0: Not save<br>• 1: Save | |

### Output Tables

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| RESULT | 1st column | ID (in input table) column type | ID (in input table) column name | Data ID. |
| | 2nd column | INTEGER | CLUSTER_ID | Cluster ID (from 0 to cluster_number -1).<br>**Note**: -1 means the point is labeled as noise. |
| MODEL | 1st column | INTEGER | ROW_INDEX | Specifies row. |
| | 2nd column | NVARCHAR(5000) | MODEL_CONTENT | Model contents. |
| STATISTICS | 1st column | NVARCHAR(1000) | STAT_NAME | Statistics name. |
| | 2nd column | NVARCHAR(1000) | STAT_VALUE | Statistics value. |

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| PLACEHOLDER | 1st column | NVARCHAR (256) | PARAM_NAME | Placeholder for future features. |
| | 2nd column | INTEGER | INT_VALUE | |
| | 3rd column | DOUBLE | DOUBLE_VALUE | |
| | 4th column | NVARCHAR (1000) | STRING_VALUE | |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_DBSCAN_DATA_TBL;
CREATE COLUMN TABLE PAL_DBSCAN_DATA_TBL (
    "ID" INTEGER,
    "V1" DOUBLE,
    "V2" DOUBLE,
    "V3" VARCHAR(10)
);
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(1, 0.10, 0.10, 'B');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(2, 0.11, 0.10, 'A');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(3, 0.10, 0.11, 'C');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(4, 0.11, 0.11, 'B');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(5, 0.12, 0.11, 'A');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(6, 0.11, 0.12, 'E');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(7, 0.12, 0.12, 'A');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(8, 0.12, 0.13, 'C');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(9, 0.13, 0.12, 'D');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(10, 0.13, 0.13, 'D');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(11, 0.13, 0.14, 'A');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(12, 0.14, 0.13, 'C');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(13, 10.10, 10.10, 'A');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(14, 10.11, 10.10, 'F');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(15, 10.10, 10.11, 'E');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(16, 10.11, 10.11, 'E');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(17, 10.11, 10.12, 'A');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(18, 10.12, 10.11, 'B');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(19, 10.12, 10.12, 'B');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(20, 10.12, 10.13, 'D');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(21, 10.13, 10.12, 'F');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(22, 10.13, 10.13, 'A');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(23, 10.13, 10.14, 'A');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(24, 10.14, 10.13, 'D');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(25, 4.10, 4.10, 'A');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(26, 7.11, 7.10, 'C');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(27, -3.10, -3.11, 'C');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(28, 16.11, 16.11, 'A');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(29, 20.11, 20.12, 'C');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(30, 15.12, 15.11, 'A');
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR(100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRIN_VALUE" VARCHAR(100)
```

```
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 0.2, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('AUTO_PARAM', NULL, NULL, 'true');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('DISTANCE_METHOD', 1, NULL, NULL);
CALL _SYS_AFL.PAL_DBSCAN (PAL_DBSCAN_DATA_TBL, "#PAL_PARAMETER_TBL", ?, ?, ?, ?);
```

**Expected Result**

RESULT:

| | ID | CLUSTER_ID |
|---|---|---|
| 1 | 1 | 0 |
| 2 | 2 | 0 |
| 3 | 3 | 0 |
| 4 | 4 | 0 |
| 5 | 5 | 0 |
| 6 | 6 | 0 |
| 7 | 7 | 0 |
| 8 | 8 | 0 |
| 9 | 9 | 0 |
| 10 | 10 | 0 |
| 11 | 11 | 0 |
| 12 | 12 | 0 |
| 13 | 13 | 1 |
| 14 | 14 | 1 |
| 15 | 15 | 1 |
| 16 | 16 | 1 |
| 17 | 17 | 1 |
| 18 | 18 | 1 |
| 19 | 19 | 1 |
| 20 | 20 | 1 |
| 21 | 21 | 1 |
| 22 | 22 | 1 |
| 23 | 23 | 1 |
| 24 | 24 | 1 |
| 25 | 25 | -1 |
| 26 | 26 | -1 |
| 27 | 27 | -1 |
| 28 | 28 | -1 |
| 29 | 29 | -1 |
| 30 | 30 | -1 |

MODEL:

| | ROW_INDEX | MODEL_CONTENT |
|---|---|---|
| 1 | 0 | {"Algorithm":"DBSCAN","Category":[{"CategoryCol":3,"CategoryName":["B","A","C","E","... |

## 5.1.7 Gaussian Mixture Model (GMM)

GMM is a Gaussian mixture model in which each component has its own weight, mean, and covariance matrix.

Weight means the importance of a Gaussian distribution in the GMM, and mean and covariance matrix are the basic parameters of a Gaussian distribution, as shown in the following formula:

$$P(x \mid \theta_1 \ldots \theta_K p_1 \ldots p_K = \sum_{k=1}^{K} p_k p(x \mid \theta_k)$$

$$p(x \mid \theta_k) = p(x \mid \mu_k \Sigma_k)$$

$$= \frac{1}{(2\pi)^{d/2} \mid \Sigma \mid^{1/2}} e^{-\frac{1}{2}(\vec{x}-\vec{\mu})^t \Sigma^{-1} (\vec{x}-\vec{\mu})}$$

Expectation maximization (EM) algorithm is used to inference all of the unknown parameters of GMM. The algorithm performs two steps: the expectation step and the maximization step.

The expectation step calculates the contribution of training sample i to the Gaussian k:

$$\gamma_{i,k}^{j} = \frac{p_k^j P(x_i \mid \theta_k^j)}{\sum_{k'} p_{k'}^j \cdot P(x_i \mid \theta_{k'}^j)}$$

The maximization step calculates the parameters weight, mean, and covariance matrix:

$$p_k^{j+1} = \frac{1}{N} \sum_{i=1}^{N} \frac{p_k^j P(x_i \mid \theta_k^j)}{\sum_{k'} p_{k'}^j \cdot P(x_i \mid \theta_{k'}^j)}$$

$$\mu_k^{j+1} = \frac{1}{\sum_{i=1}^{N} \gamma_{i,k}^j} \sum_{i=1}^{N} \gamma_{i,k}^j x_i$$

$$\Sigma_k^{j+1} = \frac{1}{\sum_{i=1}^{N} \gamma_{i,k}^j} \sum_{i=1}^{N} \gamma_{i,k}^j (x_i - \mu_k)(x_i - \mu_k)^t$$

GMM can be used in image segmentation, clustering, and so on. It gives the probability of a sample belonging to each Gaussian component.

### Prerequisite

- No missing or null data in the inputs

## _SYS_AFL.PAL_GMM

#### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<INITIALIZE_PARAMETER table>` | IN |
| 3 | `<PARAMETER table>` | IN |
| 4 | `<RESULT table>` | OUT |
| 5 | `<MODEL table>` | OUT |
| 6 | `<STATISTICS table>` | OUT |
| 7 | `<PLACEHOLDER table>` | OUT |

### Input Tables

| Table | Column | Reference for Output Table | Data Type | Description |
|---|---|---|---|---|
| DATA | 1st column | ID | INTEGER, VARCHAR, or NVARCHAR | ID |
| | Feature columns | | INTEGER, DOUBLE, DECIMAL(p, s), VARCHAR, or NVARCHAR | Data |
| INITIALIZE_PARAMETER | 1st column | | INTEGER, VARCHAR, or NVARCHAR | ID |

| Table | Column | Reference for Output Table | Data Type | Description |
|---|---|---|---|---|
| | 2nd columns | | INTEGER, VARCHAR, or NVARCHAR | When `INIT_MODE` is set to 0, 2, or 3 in the parameter table, then this value is the number of clusters in GMM. In this case, only the first row of this table is used.<br><br>When `INIT_MODE` is set to 1 in the parameter table, then values in this column are the "IDs" of the specified initial centers in the data table. Each row of this column represents an initial center in the data table. The number of clusters is equal to the number of the given initial centers. For example, setting initial centers 1, 2, 10 means selecting the rows in the input data table whose value of the ID column is 1, 2, or 10, and the number of clusters will be 3.<br><br>When `INIT_MODE` is set to 1, the type of this column should be the same as the type of the ID column in the DATA table. |

### Parameter Table

Mandatory Parameter

The following parameter is mandatory and must be given a value.

| Name | Data Type | Description |
| --- | --- | --- |
| INIT_MODE | INTEGER | Specifies the initialization mode.<br>• 0: Sets the number of the Gaussian distributions of GMM in the initialize parameter table. The initial centers will be given by the farthest-first traversal algorithm.<br>• 1: Specifies the ID of the initial centers in the initialize parameter table. The number of the components in GMM will be equal to the number of different initial centers that are given.<br>• 2: Sets the number of the Gaussian distributions of GMM in the initialize parameter table. The responsibilities matrix (r_ik) is randomly initialized, which means the initial centers are the means of all the data that are randomly weighted.<br>• 3: Sets the number of the Gaussian distributions of GMM in the initialize parameter table. The initial centers are given using the k-means++ approach. |

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description |
| --- | --- | --- | --- |
| COVARIANCE_TYPE | INTEGER | 0 | Indicates the type of covariance matrices in the model.<br>• 0: uses full covariance matrices.<br>• 1: uses diagonal covariance matrices.<br>• 2: uses diagonal covariance matrices with all equal diagonal entries. |

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| SHARED_COVARIANCE | INTEGER | 0 | Indicates whether all the clusters share the same covariance matrix.<br>• 0: clusters have different covariance matrices.<br>• 1: clusters share a same covariance matrix. |
| THREAD_RATIO | DOUBLE | 0 | Specifies the ratio of total number of threads that can be used by this function. The value range is from 0 to 1, where 0 means only using 1 thread, and 1 means using at most all the currently available threads. Values outside the range will be ignored and this function heuristically determines the number of threads to use. |
| MAX_ITERATION | INTEGER | 100 | Specifies the maximum iteration number of the EM algorithm.<br>Value range: [1, +∞) |
| CATEGORICAL_VARIABLE | VARCHAR | No default value | Indicates whether or not a column data is actually corresponding to a category variable even the data type of this column is INTEGER.<br>By default, 'VARCHAR' or 'NVARCHAR' is category variable, and 'INTEGER' or 'DOUBLE' is continuous variable. |
| CATEGORY_WEIGHT | DOUBLE | 0.707 | Represents the weight of category attributes (γ).<br>Value range: [0, +∞) |
| ERROR_TOL | DOUBLE | 1e-5 | Specifies the error tolerance, which is the stop condition.<br>Value range: (0, +∞) |

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| REGULARIZATION | DOUBLE | 1e-6 | Regularization to be added to the diagonal of covariance matrices to ensure positive-definite. |
| SEED | INTEGER | 0 | Specifies the seed for random number generator.<br>• 0: Uses the current time as the seed.<br>• Others: Uses the specified value as the seed. |
| EXCLUDED_FEATURE | VARCHAR | No default value | Specifies the feature(s) to be excluded from training. |

### Output Tables

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| RESULT | 1st column | ID (in input table) data type | ID (in input table) column name | ID |
| | 2nd column | INTEGER | CLUSTER_ID | Cluster ID |
| | 3rd column | DOUBLE | PROBABILITY | The probabilities the data belongs to each component in GMM. |
| MODEL | 1st column | INTEGER | ROW_INDEX | Row index |
| | 2nd column | INTEGER | CLUSTER_ID | Cluster ID |
| | 3rd column | NVARCHAR(5000) | MODEL_CONTENT | Content of the corresponding cluster.<br>For cluster ID -1, it contains the metadata of the model instead of any specific clusters. |
| STATISTICS | 1st column | NVARCHAR(256) | STAT_NAME | Statistics |
| | 2nd column | NVARCHAR(1000) | STAT_VALUE | |
| PLACEHOLDER | 1st column | NVARCHAR(256) | PARAM_NAME | Placeholder for future features |

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| | 2nd column | INTEGER | INT_VALUE | |
| | 3rd column | DOUBLE | DOUBLE_VALUE | |
| | 4th column | NVARCHAR(1000) | STRING_VALUE | |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_GMM_DATA_TBL;
CREATE COLUMN TABLE PAL_GMM_DATA_TBL (ID INTEGER, X1 DOUBLE, X2 DOUBLE, X3
INTEGER);
INSERT INTO PAL_GMM_DATA_TBL VALUES (0, 0.10, 0.10, 1);
INSERT INTO PAL_GMM_DATA_TBL VALUES (1, 0.11, 0.10, 1);
INSERT INTO PAL_GMM_DATA_TBL VALUES (2, 0.10, 0.11, 1);
INSERT INTO PAL_GMM_DATA_TBL VALUES (3, 0.11, 0.11, 1);
INSERT INTO PAL_GMM_DATA_TBL VALUES (4, 0.12, 0.11, 1);
INSERT INTO PAL_GMM_DATA_TBL VALUES (5, 0.11, 0.12, 1);
INSERT INTO PAL_GMM_DATA_TBL VALUES (6, 0.12, 0.12, 1);
INSERT INTO PAL_GMM_DATA_TBL VALUES (7, 0.12, 0.13, 1);
INSERT INTO PAL_GMM_DATA_TBL VALUES (8, 0.13, 0.12, 2);
INSERT INTO PAL_GMM_DATA_TBL VALUES (9, 0.13, 0.13, 2);
INSERT INTO PAL_GMM_DATA_TBL VALUES (10, 0.13, 0.14, 2);
INSERT INTO PAL_GMM_DATA_TBL VALUES (11, 0.14, 0.13, 2);
INSERT INTO PAL_GMM_DATA_TBL VALUES (12, 10.10, 10.10, 1);
INSERT INTO PAL_GMM_DATA_TBL VALUES (13, 10.11, 10.10, 1);
INSERT INTO PAL_GMM_DATA_TBL VALUES (14, 10.10, 10.11, 1);
INSERT INTO PAL_GMM_DATA_TBL VALUES (15, 10.11, 10.11, 1);
INSERT INTO PAL_GMM_DATA_TBL VALUES (16, 10.11, 10.12, 2);
INSERT INTO PAL_GMM_DATA_TBL VALUES (17, 10.12, 10.11, 2);
INSERT INTO PAL_GMM_DATA_TBL VALUES (18, 10.12, 10.12, 2);
INSERT INTO PAL_GMM_DATA_TBL VALUES (19, 10.12, 10.13, 2);
INSERT INTO PAL_GMM_DATA_TBL VALUES (20, 10.13, 10.12, 2);
INSERT INTO PAL_GMM_DATA_TBL VALUES (21, 10.13, 10.13, 2);
INSERT INTO PAL_GMM_DATA_TBL VALUES (22, 10.13, 10.14, 2);
INSERT INTO PAL_GMM_DATA_TBL VALUES (23, 10.14, 10.13, 2);
DROP TABLE PAL_GMM_INITIALIZE_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_GMM_INITIALIZE_PARAMETER_TBL (ID INTEGER, CLUSTER_NUMBER
INTEGER);
INSERT INTO PAL_GMM_INITIALIZE_PARAMETER_TBL VALUES (0, 2);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL ("PARAM_NAME"
VARCHAR(256), "INT_VALUE" INTEGER, "DOUBLE_VALUE" DOUBLE, "STRING_VALUE"
VARCHAR(1000));
INSERT INTO #PAL_PARAMETER_TBL VALUES ('COVARIANCE_TYPE', 0, NULL, NULL); -- use
full covariance matrices
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SHARED_COVARIANCE', 0, NULL, NULL); --
use different covariance matrices for different clusters
INSERT INTO #PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 0.5, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MAX_ITERATION', 500, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('INIT_MODE', 0, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('ERROR_TOL', NULL, 0.001, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('CATEGORICAL_VARIABLE', NULL, NULL, 'X3');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SEED', 1, NULL, NULL);
```

```
CALL _SYS_AFL.PAL_GMM (PAL_GMM_DATA_TBL, PAL_GMM_INITIALIZE_PARAMETER_TBL,
#PAL_PARAMETER_TBL, ?, ?, ?, ?);
```

### Expected Results

RESULT:

| | ID | CLUSTER_ID | PROBABILITY |
|---|---|---|---|
| 1 | 0 | 0 | 0 |
| 2 | 1 | 0 | 0 |
| 3 | 2 | 0 | 0 |
| 4 | 3 | 0 | 0 |
| 5 | 4 | 0 | 0 |
| 6 | 5 | 0 | 0 |
| 7 | 6 | 0 | 0 |
| 8 | 7 | 0 | 0 |
| 9 | 8 | 0 | 1 |
| 10 | 9 | 0 | 1 |
| 11 | 10 | 0 | 1 |
| 12 | 11 | 0 | 1 |
| 13 | 12 | 0 | 0 |
| 14 | 13 | 0 | 0 |
| 15 | 14 | 0 | 0 |
| 16 | 15 | 0 | 0 |
| 17 | 16 | 0 | 1 |
| 18 | 17 | 0 | 1 |
| 19 | 18 | 0 | 1 |
| 20 | 19 | 0 | 1 |
| 21 | 20 | 0 | 1 |
| 22 | 21 | 0 | 1 |
| 23 | 22 | 0 | 1 |
| 24 | 23 | 0 | 1 |
| 25 | 0 | 1 | 1 |
| 26 | 1 | 1 | 1 |
| 27 | 2 | 1 | 1 |

MODEL:

| | ROW_INDEX | CLUSTER_ID | MODEL_CONTENT |
|---|---|---|---|
| 1 | 0 | -1 | {"Algorithm":"GMM","Metadata":{"DataProcessor":{"categoricalMaps":[[... |
| 2 | 1 | 0 | {"GaussModel":{"covariance":[22.188958638888884,22.19447777777777... |
| 3 | 2 | 1 | {"GaussModel":{"covariance":[22.194503083333331,22.19170833333332... |

STATISTICS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | log-likelihood | 11.719866294110679 |
| 2 | aic | -504.55358211731254 |
| 3 | bic | -470.39002103722214 |

PLACEHOLDER: reserved for future use

> **Note**
>
> The wrapper procedure generation approach to call PAL functions is still supported. For more information, refer to the older versions (prior to SAP HANA Platform 2.0 SPS 02) of this document.

## 5.1.8 Geometry DBSCAN

This function is a geometry version of DBSCAN, which only accepts geometry points as input data.

Currently it only accepts 2-D points.

For more information, see the DBSCAN function.

### Prerequisites

- No missing or null data in the inputs.
- Input data only contains ID and 2-D geometry points.

### _SYS_AFL.PAL_GEO_DBSCAN

This function performs DBSCAN algorithm on 2-D geometry points.

**Overview of All Tables**

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<RESULT table>` | OUT |
| 4 | `<MODEL table>` | OUT |
| 5 | `<STATISTICS table>` | OUT |
| 6 | `<PLACEHOLDER table>` | OUT |

### Input Table

| Table | Column | Reference | Data Type | Description |
|---|---|---|---|---|
| DATA | 1st column | ID | INTEGER, BIGINT, VARCHAR, or NVARCHAR | Data ID<br>This must be the first column. |
| | 2nd columns | | ST_GEOMETRY | 2-D geometry point |

### Parameter Table

Mandatory Parameters

The following parameters are mandatory and must be given a value.

| Name | Data Type | Description | Dependency |
|---|---|---|---|
| AUTO_PARAM | VARCHAR or NVARCHAR | Specifies whether the MINPTS and RADIUS parameters are determined automatically or by user.<br>• true: automatically determines the parameters<br>• false: uses parameter values provided by user | |
| MINPTS | INTEGER | Specifies the minimum number of points required to form a cluster. | Only valid when AUTO_PARAM is false. |
| RADIUS | DOUBLE | Specifies the scan radius (eps). | Only valid when AUTO_PARAM is false. |

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| THREAD_RATIO | DOUBLE | -1 | The ratio of available threads.<br>• 0: single thread<br>• 0~1: percentage<br>• Others: heuristically determined | |
| DISTANCE_METHOD | INTEGER | 2 | Specifies the method to compute the distance between two points.<br>• 1: Manhattan<br>• 2: Euclidean<br>• 3: Minkowski<br>• 4: Chebyshev<br>• 5: Standardized Euclidean<br>• 6: Cosine | |
| MINKOW_P | INTEGER | 3 | Specifies the power of the Minkowski method. | Only valid when DISTANCE_METHOD is 3. |
| METHOD | INTEGER | 1 | Specifies the searching method.<br>• 0: Brute force searching<br>• 1: KD-tree searching | |
| SAVE_MODEL | INTEGER | 1 | Indicates whether model should be saved.<br>• 0: Not save<br>• 1: Save | |

### Output Tables

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| RESULT | 1st column | ID (in input table) column type | ID (in input table) column name | Data ID. |

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| | 2nd column | INTEGER | CLUSTER_ID | Cluster ID (from 0 to cluster_number -1).<br>**Note**<br>-1 means the point is labeled as noise. |
| MODEL | 1st column | INTEGER | ROW_INDEX | Specifies row. |
| | 2nd column | NVARCHAR(5000) | MODEL_CONTENT | Model contents. |
| STATISTICS | 1st column | NVARCHAR(1000) | STAT_NAME | Statistics name. |
| | 2nd column | NVARCHAR(1000) | STAT_VALUE | Statistics value. |
| PLACEHOLDER | 1st column | NVARCHAR (256) | PARAM_NAME | Placeholder for future features. |
| | 2nd column | INTEGER | INT_VALUE | |
| | 3rd column | DOUBLE | DOUBLE_VALUE | |
| | 4th column | NVARCHAR (1000) | STRING_VALUE | |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_GEO_DBSCAN_DATA_TBL;
CREATE COLUMN TABLE PAL_GEO_DBSCAN_DATA_TBL (
    "ID" INTEGER,
    "POINT" ST_GEOMETRY
);
INSERT INTO PAL_GEO_DBSCAN_DATA_TBL VALUES(1, ST_GEOMFROMTEXT('Point(0.10
0.10)'));
INSERT INTO PAL_GEO_DBSCAN_DATA_TBL VALUES(2, ST_GEOMFROMTEXT('Point(0.11
0.10)'));
INSERT INTO PAL_GEO_DBSCAN_DATA_TBL VALUES(3, ST_GEOMFROMTEXT('Point(0.10
0.11)'));
INSERT INTO PAL_GEO_DBSCAN_DATA_TBL VALUES(4, ST_GEOMFROMTEXT('Point(0.11
0.11)'));
INSERT INTO PAL_GEO_DBSCAN_DATA_TBL VALUES(5, ST_GEOMFROMTEXT('Point(0.12
0.11)'));
INSERT INTO PAL_GEO_DBSCAN_DATA_TBL VALUES(6, ST_GEOMFROMTEXT('Point(0.11
0.12)'));
INSERT INTO PAL_GEO_DBSCAN_DATA_TBL VALUES(7, ST_GEOMFROMTEXT('Point(0.12
0.12)'));
INSERT INTO PAL_GEO_DBSCAN_DATA_TBL VALUES(8, ST_GEOMFROMTEXT('Point(0.12
0.13)'));
INSERT INTO PAL_GEO_DBSCAN_DATA_TBL VALUES(9, ST_GEOMFROMTEXT('Point(0.13
0.12)'));
```

```
INSERT INTO PAL_GEO_DBSCAN_DATA_TBL VALUES(10, ST_GEOMFROMTEXT('Point(0.13
0.13)'));
INSERT INTO PAL_GEO_DBSCAN_DATA_TBL VALUES(11, ST_GEOMFROMTEXT('Point(0.13
0.14)'));
INSERT INTO PAL_GEO_DBSCAN_DATA_TBL VALUES(12, ST_GEOMFROMTEXT('Point(0.14
0.13)'));
INSERT INTO PAL_GEO_DBSCAN_DATA_TBL VALUES(13, ST_GEOMFROMTEXT('Point(10.10
10.10)'));
INSERT INTO PAL_GEO_DBSCAN_DATA_TBL VALUES(14, ST_GEOMFROMTEXT('Point(10.11
10.10)'));
INSERT INTO PAL_GEO_DBSCAN_DATA_TBL VALUES(15, ST_GEOMFROMTEXT('Point(10.10
10.11)'));
INSERT INTO PAL_GEO_DBSCAN_DATA_TBL VALUES(16, ST_GEOMFROMTEXT('Point(10.11
10.11)'));
INSERT INTO PAL_GEO_DBSCAN_DATA_TBL VALUES(17, ST_GEOMFROMTEXT('Point(10.11
10.12)'));
INSERT INTO PAL_GEO_DBSCAN_DATA_TBL VALUES(18, ST_GEOMFROMTEXT('Point(10.12
10.11)'));
INSERT INTO PAL_GEO_DBSCAN_DATA_TBL VALUES(19, ST_GEOMFROMTEXT('Point(10.12
10.12)'));
INSERT INTO PAL_GEO_DBSCAN_DATA_TBL VALUES(20, ST_GEOMFROMTEXT('Point(10.12
10.13)'));
INSERT INTO PAL_GEO_DBSCAN_DATA_TBL VALUES(21, ST_GEOMFROMTEXT('Point(10.13
10.12)'));
INSERT INTO PAL_GEO_DBSCAN_DATA_TBL VALUES(22, ST_GEOMFROMTEXT('Point(10.13
10.13)'));
INSERT INTO PAL_GEO_DBSCAN_DATA_TBL VALUES(23, ST_GEOMFROMTEXT('Point(10.13
10.14)'));
INSERT INTO PAL_GEO_DBSCAN_DATA_TBL VALUES(24, ST_GEOMFROMTEXT('Point(10.14
10.13)'));
INSERT INTO PAL_GEO_DBSCAN_DATA_TBL VALUES(25, ST_GEOMFROMTEXT('Point(4.10
4.10)'));
INSERT INTO PAL_GEO_DBSCAN_DATA_TBL VALUES(26, ST_GEOMFROMTEXT('Point(7.11
7.10)'));
INSERT INTO PAL_GEO_DBSCAN_DATA_TBL VALUES(27, ST_GEOMFROMTEXT('Point(-3.10
-3.11)'));
INSERT INTO PAL_GEO_DBSCAN_DATA_TBL VALUES(28, ST_GEOMFROMTEXT('Point(16.11
16.11)'));
INSERT INTO PAL_GEO_DBSCAN_DATA_TBL VALUES(29, ST_GEOMFROMTEXT('Point(20.11
20.12)'));
INSERT INTO PAL_GEO_DBSCAN_DATA_TBL VALUES(30, ST_GEOMFROMTEXT('Point(15.12
15.11)'));
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR(100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRIN_VALUE" VARCHAR(100)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 0.2, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('AUTO_PARAM', NULL, NULL, 'true');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('DISTANCE_METHOD', 1, NULL, NULL);
CALL _SYS_AFL.PAL_GEO_DBSCAN(PAL_GEO_DBSCAN_DATA_TBL,
"#PAL_PARAMETER_TBL", ?, ?, ?, ?);
```

### Expected Results

RESULT:

| | ID | CLUSTER_ID |
|---|---|---|
| 1 | 1 | 0 |
| 2 | 2 | 0 |
| 3 | 3 | 0 |
| 4 | 4 | 0 |
| 5 | 5 | 0 |
| 6 | 6 | 0 |
| 7 | 7 | 0 |
| 8 | 8 | 0 |
| 9 | 9 | 0 |
| 10 | 10 | 0 |
| 11 | 11 | 0 |
| 12 | 12 | 0 |
| 13 | 13 | 1 |
| 14 | 14 | 1 |
| 15 | 15 | 1 |
| 16 | 16 | 1 |
| 17 | 17 | 1 |
| 18 | 18 | 1 |
| 19 | 19 | 1 |
| 20 | 20 | 1 |
| 21 | 21 | 1 |
| 22 | 22 | 1 |
| 23 | 23 | 1 |
| 24 | 24 | 1 |
| 25 | 25 | -1 |
| 26 | 26 | -1 |
| 27 | 27 | -1 |
| 28 | 28 | -1 |
| 29 | 29 | -1 |
| 30 | 30 | -1 |

MODEL:

| | ROW_INDEX | MODEL_CONTENT |
|---|---|---|
| 1 | 0 | {"Algorithm":"DBSCAN","Cluster":[{"ClusterID":0,"CoreObject... |

## 5.1.9 K-Means

In predictive analysis, k-means clustering is a method of cluster analysis. The k-means algorithm partitions n observations or records into k clusters in which each observation belongs to the cluster with the nearest center.

In marketing and customer relationship management areas, this algorithm uses customer data to track customer behavior and create strategic business initiatives. Organizations can thus divide their customers into segments based on variants such as demography, customer behavior, customer profitability, measure of risk, and lifetime value of a customer or retention probability.

Clustering works to group records together according to an algorithm or mathematical formula that attempts to find centroids, or centers, around which similar records gravitate. The most common algorithm uses an iterative refinement technique. It is also referred to as Lloyd's algorithm:

Given an initial set of k means $m_1, ..., m_k$, the algorithm proceeds by alternating between two steps:

- Assignment step: assigns each observation to the cluster with the closest mean.
- Update step: calculates the new means to be the center of the observations in the cluster.

The algorithm repeats until the assignments no longer change.

The k-means implementation in PAL supports multi-thread, data normalization, different distance level measurement, and cluster quality measurement (Silhouette). The implementation does not support categorical data, but this can be managed through data transformation. The first K and random K starting methods are supported.

**Support for Categorical Attributes**

If an attribute is of category type, it will be converted to a binary vector and then be used as a numerical attribute. For example, in the below table, "Gender" is of category type.

| Customer ID | Age | Income | Gender |
|---|---|---|---|
| T1 | 31 | 10,000 | Female |
| T2 | 27 | 8,000 | Male |

Because "Gender" has two distinct values, it will be converted into a binary vector with two dimensions:

| Customer ID | Age | Income | Gender_1 | Gender_2 |
|---|---|---|---|---|
| T1 | 31 | 10,000 | 0 | 1 |

| Customer ID | Age | Income | Gender_1 | Gender_2 |
|---|---|---|---|---|
| T2 | 27 | 8,000 | 1 | 0 |

Thus, the Euclidean distance between T1 and T2 is:

$$d(T_1, T_2)=\sqrt{(31-27)^2+(10000-8000)^2}+\gamma\sqrt{(0-1)^2+(1-0)^2}$$

Where γ is the weight to be given to the transposed categorical attributes to lessen the impact on the clustering from the 0/1 attributes. Then you can use the traditional method to update the mean of every cluster. Assuming one cluster only has T1 and T2, the mean is:

| Customer ID | Age | Income | Gender_1 | Gender_2 |
|---|---|---|---|---|
| Center1 | 29.0 | 9000.0 | 0.5 | 0.5 |

The means of categorical attributes will not be outputted. Instead, the means will be replaced by the modes similar to the K-Modes algorithm. Take the below center for example:

| | Age | Income | Gender_1 | Gender_2 |
|---|---|---|---|---|
| Center | 29.0 | 9000.0 | 0.25 | 0.75 |

Because "Gender_2" is the maximum value, the output will be:

| | Age | Income | Gender |
|---|---|---|---|
| Center | 29.0 | 9000.0 | Female |

## Prerequisites

- The input data contains an ID column.
- The input data does not contain null value. The algorithm will issue errors when encountering null values.

## _SYS_AFL.PAL_KMEANS

This is a clustering function using the k-means algorithm.

**Overview of All Tables**

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<RESULT table>` | OUT |
| 4 | `<CENTERS table>` | OUT |
| 5 | `<MODEL table>` | OUT |
| 6 | `<STATISTICS table>` | OUT |
| 7 | `<PLACEHOLDER table>` | OUT |

### Input Table

| Table | Column | Reference for Output Table | Data Type | Description |
|---|---|---|---|---|
| DATA | 1st column | ID | INTEGER, BIGINT, VARCHAR, or NVARCHAR | Record ID |
| | Other columns | FEATURES | INTEGER, DOUBLE, DECIMAL(p, s), VARCHAR, or NVARCHAR | Attribute data |

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| GROUP_NUMBER | INTEGER | No default value | Number of groups (k). The value range is from 2 to the number of training records.<br><br>**Note**: If this parameter is not specified, you must specify the range of k using GROUP_NUMBER_MIN and GROUP_NUMBER_MAX. Then the algorithm will iterate through the range and return the k with the highest slight silhouette. | |
| GROUP_NUMBER_MIN | INTEGER | No default value | Lower boundary of the range that k falls in.<br><br>**Note**: You must specify either an exact value or a range for k. If both are specified, the exact value will be used. | |
| GROUP_NUMBER_MAX | INTEGER | No default value | Upper boundary of the range that k falls in.<br><br>**Note**: You must specify either an exact value or a range for k. If both are specified, the exact value will be used. | |
| DISTANCE_LEVEL | INTEGER | 2 | Computes the distance between the item and the cluster center.<br>• 1: Manhattan distance<br>• 2: Euclidean distance<br>• 3: Minkowski distance<br>• 4: Chebyshev distance<br>• 6: Cosine distance | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| MINKOWSKI_POWER | DOUBLE | 3.0 | When you use the Minkowski distance, this parameter controls the value of power. | Only valid when DISTANCE_LEVEL is 3. |
| CATEGORY_WEIGHTS | DOUBLE | 0.707 | Represents the weight of category attributes (γ). | |
| MAX_ITERATION | INTEGER | 100 | Maximum iterations. | |
| INIT_TYPE | INTEGER | 4 | Controls how the initial centers are selected:<br>• 1: First k observations<br>• 2: Random with replacement<br>• 3: Random without replacement<br>• 4: Patent of selecting the init center (US 6,882,998 B1) | |
| NORMALIZATION | INTEGER | 0 | Normalization type:<br>• 0: No<br>• 1: Yes. For each point X (x1,x2,...,xn), the normalized value will be X'(x1/S,x2/S,...,xn/S), where S = \|x1\|+\|x2\|+...\|xn\|.<br>• 2: for each column C, get the min and max value of C, and then C[i] = (C[i]-min)/(max-min). | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| THREAD_RATIO | DOUBLE | 0 | Specifies the ratio of total number of threads that can be used by this function. The value range is from 0 to 1, where 0 means only using 1 thread, and 1 means using at most all the currently available threads. Values outside the range will be ignored and this function heuristically determines the number of threads to use. | |
| CATEGORICAL_VARIABLE | NVARCHAR | No default value | Indicates whether or not a column data is actually corresponding to a category variable even the data type of this column is INTEGER.<br><br>By default, 'VARCHAR' or 'NVARCHAR' is category variable, and 'INTEGER' or 'DOUBLE' is continuous variable. | |
| EXIT_THRESHOLD | DOUBLE | 1e-6 | Threshold (actual value) for exiting the iterations. | |

## Output Tables

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| RESULT | 1st column | ID (in input table) data type | ID (in input table) column name | Record ID |
| | 2nd column | INTEGER | CLUSTER_ID | Cluster ID assigned to it |

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| | 3rd column | DOUBLE | DISTANCE | The distance between the given point and the cluster center |
| | 4th column | DOUBLE | SLIGHT_SILHOUETTE | Estimated value (slight silhouette) |
| CENTERS | 1st column | INTEGER | CLUSTER_ID | Cluster center ID |
| | Other columns | FEATURES (in input table) data type with conversion from integer to double | FEATURES (in input table) column name | Cluster center coordinates |
| MODEL | 1st column | INTEGER | ROW_INDEX | Cluster model row index |
| | 2nd column | VARCHAR or NVARCHAR | MODEL_CONTENT | Cluster model saved as JSON string. The table must be a column table. The minimum length of every unit (row) is 5000. |
| STATISTICS | 1st column | VARCHAR or NVARCHAR | STAT_NAME | Statistic name.<br>1. Slight Silhouette as whole<br>2. Slight Silhouette for each cluster<br>3. ANOVA F-test for each K, if a range of candidate of K is provided instead of single one. |
| | 2nd column | VARCHAR or NVARCHAR | STAT_VALUE | Statistic value |
| PLACEHOLDER | 1st column | NVARCHAR(256) | PARAM_NAME | Placeholder for future features |
| | 2nd column | INTEGER | INT_VALUE | |
| | 3rd column | DOUBLE | DOUBLE_VALUE | |
| | 4th column | NVARCHAR(1000) | STRING_VALUE | |

### Example

Assume that:

- DM_PAL is a schema belonging to USER1;
- USER1 has been assigned the AFL__SYS_AFL_AFLPAL_EXECUTE or AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION role.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_KMEANS_DATA_TBL;
CREATE COLUMN TABLE PAL_KMEANS_DATA_TBL(
    "ID" INTEGER,
    "V000" DOUBLE,
    "V001" VARCHAR(2),
    "V002" DOUBLE
);
INSERT INTO PAL_KMEANS_DATA_TBL VALUES (0, 0.5, 'A', 0.5);
INSERT INTO PAL_KMEANS_DATA_TBL VALUES (1, 1.5, 'A', 0.5);
INSERT INTO PAL_KMEANS_DATA_TBL VALUES (2, 1.5, 'A', 1.5);
INSERT INTO PAL_KMEANS_DATA_TBL VALUES (3, 0.5, 'A', 1.5);
INSERT INTO PAL_KMEANS_DATA_TBL VALUES (4, 1.1, 'B', 1.2);
INSERT INTO PAL_KMEANS_DATA_TBL VALUES (5, 0.5, 'B', 15.5);
INSERT INTO PAL_KMEANS_DATA_TBL VALUES (6, 1.5, 'B', 15.5);
INSERT INTO PAL_KMEANS_DATA_TBL VALUES (7, 1.5, 'B', 16.5);
INSERT INTO PAL_KMEANS_DATA_TBL VALUES (8, 0.5, 'B', 16.5);
INSERT INTO PAL_KMEANS_DATA_TBL VALUES (9, 1.2, 'C', 16.1);
INSERT INTO PAL_KMEANS_DATA_TBL VALUES (10, 15.5, 'C', 15.5);
INSERT INTO PAL_KMEANS_DATA_TBL VALUES (11, 16.5, 'C', 15.5);
INSERT INTO PAL_KMEANS_DATA_TBL VALUES (12, 16.5, 'C', 16.5);
INSERT INTO PAL_KMEANS_DATA_TBL VALUES (13, 15.5, 'C', 16.5);
INSERT INTO PAL_KMEANS_DATA_TBL VALUES (14, 15.6, 'D', 16.2);
INSERT INTO PAL_KMEANS_DATA_TBL VALUES (15, 15.5, 'D', 0.5);
INSERT INTO PAL_KMEANS_DATA_TBL VALUES (16, 16.5, 'D', 0.5);
INSERT INTO PAL_KMEANS_DATA_TBL VALUES (17, 16.5, 'D', 1.5);
INSERT INTO PAL_KMEANS_DATA_TBL VALUES (18, 15.5, 'D', 1.5);
INSERT INTO PAL_KMEANS_DATA_TBL VALUES (19, 15.7, 'A', 1.6);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL(
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR(1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 0.2, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('GROUP_NUMBER', 4, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('INIT_TYPE', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('DISTANCE_LEVEL',2, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MAX_ITERATION', 100, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('EXIT_THRESHOLD', NULL, 1.0E-6, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('CATEGORY_WEIGHTS', NULL, 0.5, NULL);
CALL _SYS_AFL.PAL_KMEANS(PAL_KMEANS_DATA_TBL,
"#PAL_PARAMETER_TBL", ?, ?, ?, ?, ?);
```

### Expected Result

RESULT:

| | ID | CLUSTER_ID | DISTANCE | SLIGHT_SILHOUETTE |
|---|---|---|---|---|
| 1 | 0 | 0 | 0.8910879487969621 | 0.9443700340289641 |
| 2 | 1 | 0 | 0.8639170309648472 | 0.9424783519324079 |
| 3 | 2 | 0 | 0.8062521617810958 | 0.9462881150703033 |
| 4 | 3 | 0 | 0.8356835545457009 | 0.944941534308603 |
| 5 | 4 | 0 | 0.7445708631492213 | 0.9502343526704448 |
| 6 | 5 | 3 | 0.8910879487969616 | 0.9407325552117667 |
| 7 | 6 | 3 | 0.8356835545457004 | 0.9444124645366334 |
| 8 | 7 | 3 | 0.8062521617810963 | 0.9465187213724425 |
| 9 | 8 | 3 | 0.8639170309648476 | 0.9461211197877126 |
| 10 | 9 | 3 | 0.744570863149222 | 0.9498994935338706 |
| 11 | 10 | 2 | 0.8255266112967905 | 0.945092293345542 |
| 12 | 11 | 2 | 0.9338858664836682 | 0.9379022885331548 |
| 13 | 12 | 2 | 0.8816915771701814 | 0.9450081614922042 |
| 14 | 13 | 2 | 0.7643178163332063 | 0.9491595311531508 |
| 15 | 14 | 2 | 0.9234563013492032 | 0.939282992463176 |
| 16 | 15 | 1 | 0.9016844685872387 | 0.940436243125126 |
| 17 | 16 | 1 | 0.9768852992055255 | 0.9393860344624405 |
| 18 | 17 | 1 | 0.8181783288648536 | 0.9458778291291957 |
| 19 | 18 | 1 | 0.7227990303872558 | 0.9521697340450603 |
| 20 | 19 | 1 | 1.1023417395491886 | 0.9256786403653229 |

CENTERS:

| | CLUSTER_ID | V000 | V001 | V002 |
|---|---|---|---|---|
| 1 | 0 | 1.02 | A | 1.04 |
| 2 | 1 | 15.940000000000001 | D | 1.1199999999999999 |
| 3 | 2 | 15.919999999999998 | C | 16.04 |
| 4 | 3 | 1.04 | B | 16.02 |

MODEL:

| | ROW_INDEX | MODEL_CONTENT |
|---|---|---|
| 1 | 0 | {"Algorithm":"KMEANS","Category":[{"CategoryCol":2,"Categ... |

STATISTICS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | Center 0 Slight Silhouette | 0.9456624776021446 |
| 2 | Center 1 Slight Silhouette | 0.9407096962254291 |
| 3 | Center 2 Slight Silhouette | 0.9432890533974454 |
| 4 | Center 3 Slight Silhouette | 0.9455368708884853 |
| 5 | SlightSilhouette | 0.943799524528376 |

PLACEHOLDER: reserved for future use

## _SYS_AFL.PAL_VALIDATE_KMEANS

This is a quality measurement function for k-means clustering.

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<CLASS DATA table>` | IN |
| 3 | `<PARAMETER table>` | IN |
| 4 | `<MEASURE table>` | OUT |

### Input Tables

| Table | Column | Data Type | Description |
|---|---|---|---|
| DATA | 1st column | INTEGER | Record ID |
| | Other columns | INTEGER or DOUBLE | Attribute data |
| CLASS_DATA | 1st column | INTEGER | Record ID |
| | 2nd column | INTEGER | Class type |

### Parameter Table

Mandatory Parameter

The following parameter is mandatory and must be given a value.

| Name | Data Type | Description |
|---|---|---|
| VARIABLE_NUM | INTEGER | Number of variables |

Optional Parameter

The following parameter is optional. If it is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| THREAD_RATIO | DOUBLE | 0 | Specifies the ratio of total number of threads that can be used by this function. The value range is from 0 to 1, where 0 means only using 1 thread, and 1 means using at most all the currently available threads. Values outside the range will be ignored and this function heuristically determines the number of threads to use. |

### Output Table

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| MEASURE | 1st column | VARCHAR or NVARCHAR | STAT_NAME | Statistic name |
| | 2nd column | DOUBLE | STAT_VALUE | Statistic value |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_SILHOUETTE_DATA_TBL;
CREATE COLUMN TABLE PAL_SILHOUETTE_DATA_TBL(
    "ID" INTEGER,
    "V000" DOUBLE,
    "A0" INTEGER,
    "A1" INTEGER,
    "A2" INTEGER,
    "A3" INTEGER,
    "V002" DOUBLE
);
INSERT INTO PAL_SILHOUETTE_DATA_TBL VALUES (0, 0.5, 1, 0, 0, 0, 0.5);
INSERT INTO PAL_SILHOUETTE_DATA_TBL VALUES (1, 1.5, 1, 0, 0, 0, 0.5);
INSERT INTO PAL_SILHOUETTE_DATA_TBL VALUES (2, 1.5, 1, 0, 0, 0, 1.5);
```

```
INSERT INTO PAL_SILHOUETTE_DATA_TBL VALUES (3, 0.5, 1, 0, 0, 0, 1.5);
INSERT INTO PAL_SILHOUETTE_DATA_TBL VALUES (4, 1.1, 0, 1, 0, 0, 1.2);
INSERT INTO PAL_SILHOUETTE_DATA_TBL VALUES (5, 0.5, 0, 1, 0, 0, 15.5);
INSERT INTO PAL_SILHOUETTE_DATA_TBL VALUES (6, 1.5, 0, 1, 0, 0, 15.5);
INSERT INTO PAL_SILHOUETTE_DATA_TBL VALUES (7, 1.5, 0, 1, 0, 0, 16.5);
INSERT INTO PAL_SILHOUETTE_DATA_TBL VALUES (8, 0.5, 0, 1, 0, 0, 16.5);
INSERT INTO PAL_SILHOUETTE_DATA_TBL VALUES (9, 1.2, 0, 0, 1, 0, 16.1);
INSERT INTO PAL_SILHOUETTE_DATA_TBL VALUES (10, 15.5, 0, 0, 1, 0, 15.5);
INSERT INTO PAL_SILHOUETTE_DATA_TBL VALUES (11, 16.5, 0, 0, 1, 0, 15.5);
INSERT INTO PAL_SILHOUETTE_DATA_TBL VALUES (12, 16.5, 0, 0, 1, 0, 16.5);
INSERT INTO PAL_SILHOUETTE_DATA_TBL VALUES (13, 15.5, 0, 0, 1, 0, 16.5);
INSERT INTO PAL_SILHOUETTE_DATA_TBL VALUES (14, 15.6, 0, 0, 0, 1, 16.2);
INSERT INTO PAL_SILHOUETTE_DATA_TBL VALUES (15, 15.5, 0, 0, 0, 1, 0.5);
INSERT INTO PAL_SILHOUETTE_DATA_TBL VALUES (16, 16.5, 0, 0, 0, 1, 0.5);
INSERT INTO PAL_SILHOUETTE_DATA_TBL VALUES (17, 16.5, 0, 0, 0, 1, 1.5);
INSERT INTO PAL_SILHOUETTE_DATA_TBL VALUES (18, 15.5, 0, 0, 0, 1, 1.5);
INSERT INTO PAL_SILHOUETTE_DATA_TBL VALUES (19, 15.7, 1, 0, 0, 0, 1.6);
DROP TABLE PAL_SILHOUETTE_ASSIGN_TBL;
CREATE COLUMN TABLE PAL_SILHOUETTE_ASSIGN_TBL(
    "ID" INTEGER,
    "CLASS_LABEL" INTEGER
);
INSERT INTO PAL_SILHOUETTE_ASSIGN_TBL VALUES (0, 0);
INSERT INTO PAL_SILHOUETTE_ASSIGN_TBL VALUES (1, 0);
INSERT INTO PAL_SILHOUETTE_ASSIGN_TBL VALUES (2, 0);
INSERT INTO PAL_SILHOUETTE_ASSIGN_TBL VALUES (3, 0);
INSERT INTO PAL_SILHOUETTE_ASSIGN_TBL VALUES (4, 0);
INSERT INTO PAL_SILHOUETTE_ASSIGN_TBL VALUES (5, 1);
INSERT INTO PAL_SILHOUETTE_ASSIGN_TBL VALUES (6, 1);
INSERT INTO PAL_SILHOUETTE_ASSIGN_TBL VALUES (7, 1);
INSERT INTO PAL_SILHOUETTE_ASSIGN_TBL VALUES (8, 1);
INSERT INTO PAL_SILHOUETTE_ASSIGN_TBL VALUES (9, 1);
INSERT INTO PAL_SILHOUETTE_ASSIGN_TBL VALUES (10, 2);
INSERT INTO PAL_SILHOUETTE_ASSIGN_TBL VALUES (11, 2);
INSERT INTO PAL_SILHOUETTE_ASSIGN_TBL VALUES (12, 2);
INSERT INTO PAL_SILHOUETTE_ASSIGN_TBL VALUES (13, 2);
INSERT INTO PAL_SILHOUETTE_ASSIGN_TBL VALUES (14, 2);
INSERT INTO PAL_SILHOUETTE_ASSIGN_TBL VALUES (15, 3);
INSERT INTO PAL_SILHOUETTE_ASSIGN_TBL VALUES (16, 3);
INSERT INTO PAL_SILHOUETTE_ASSIGN_TBL VALUES (17, 3);
INSERT INTO PAL_SILHOUETTE_ASSIGN_TBL VALUES (18, 3);
INSERT INTO PAL_SILHOUETTE_ASSIGN_TBL VALUES (19, 3);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL(
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR(1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('VARIABLE_NUM', 6, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 0.2, NULL);
CALL _SYS_AFL.PAL_VALIDATE_KMEANS(PAL_SILHOUETTE_DATA_TBL,
PAL_SILHOUETTE_ASSIGN_TBL, "#PAL_PARAMETER_TBL", ?);
```

### Expected Result

MEASURE:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | Silhouette | 0.9275111543031423 |

> **Note**
>
> The wrapper procedure generation approach to call PAL functions is still supported. For more information, refer to the older versions (prior to SAP HANA Platform 2.0 SPS 02) of this document.

## 5.1.10 K-Medians

K-medians is a clustering algorithm similar to K-means. K-medians and K-means both partition n observations into K clusters according to their nearest cluster center. In contrast to K-means, while calculating cluster centers, K-medians uses medians of each feature instead of means of it.

A median value is the middle value of a set of values arranged in order.

Given an initial set of K cluster centers: $m_1, \ldots, m_k$, the algorithm proceeds by alternating between the following two steps and repeats until the assignments no longer change.

- Assignment step: assigns each observation to the cluster with the closest center.
- Update step: calculates the new median of each feature of each cluster to be the new center of that cluster.

The K-medians implementation in PAL supports multi-threads, data normalization, and different distance level measurements. The implementation does not support categorical data, but this can be managed through data transformation. Because median method cannot apply to categorical data, the K-medians implementation uses the most frequent one instead. The first K and random K starting methods are supported.

**Support for Categorical Attributes**

If an attribute is of category type, it will be converted to a binary vector and then be used as a numerical attribute. For example, in the below table, "Gender" is of category type.

| Customer ID | Age | Income | Gender |
|---|---|---|---|
| T1 | 31 | 10,000 | Female |
| T2 | 27 | 8,000 | Male |

Because "Gender" has two distinct values, it will be converted into a binary vector with two dimensions:

| Customer ID | Age | Income | Gender_1 | Gender_2 |
|---|---|---|---|---|
| T1 | 31 | 10,000 | 0 | 1 |
| T2 | 27 | 8,000 | 1 | 0 |

Thus, the Euclidean distance between T1 and T2 is:

$$d(T_1, T_2) = \sqrt{(31-27)^2 + (10000-8000)^2} + \gamma\sqrt{(0-1)^2 + (1-0)^2}$$

Where γ is the weight to be given to the transposed categorical attributes to lessen the impact on the clustering from the 0/1 attributes.

## Prerequisites

- The input data contains an ID column.
- The input data does not contain null value. The algorithm will issue errors when encountering null values.

## _SYS_AFL.PAL_KMEDIANS

This is a clustering function using the K-medians algorithm.

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<RESULT table>` | OUT |
| 4 | `<CENTERS table>` | OUT |

### Input Table

| Table | Column | Reference for Output Table | Data Type | Description |
|---|---|---|---|---|
| DATA | 1st column | ID | INTEGER, BIGINT, VARCHAR, or NVARCHAR | Record ID |
| | Other columns | FEATURES | INTEGER, DOUBLE, DECIMAL(p, s), VARCHAR, or NVARCHAR | Attribute data |

### Parameter Table

Mandatory Parameters

The following parameter is mandatory and must be given a value.

| Name | Data Type | Description |
|---|---|---|
| GROUP_NUMBER | INTEGER | Number of groups (k). The value range is from 2 to the number of training records. |

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| DISTANCE_LEVEL | INTEGER | 2 | Computes the distance between the item and the cluster center.<br>• 1: Manhattan distance<br>• 2: Euclidean distance<br>• 3: Minkowski distance<br>• 4: Chebyshev distance<br>• 6: Cosine distance | |
| MINKOWSKI_POWER | DOUBLE | 3.0 | When you use the Minkowski distance, this parameter controls the value of power. | Only valid when DISTANCE_LEVEL is 3. |
| CATEGORY_WEIGHTS | DOUBLE | 0.707 | Represents the weight of category attributes (γ). | |
| MAX_ITERATION | INTEGER | 100 | Maximum iterations. | |
| INIT_TYPE | INTEGER | 4 | Controls how the initial centers are selected:<br>• 1: first k observations<br>• 2: random with replacement<br>• 3: random without replacement<br>• 4: patent of selecting the initial center (US 6,882,998 B1) | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| NORMALIZATION | INTEGER | 0 | Normalization type:<br>• 0: No<br>• 1: Yes. For each point X (x1,x2,...,xn), the normalized value will be X'(x1/S,x2/S,...,xn/S), where S = \|x1\|+\|x2\|+...\|xn\|.<br>• 2: For each column C, get the min and max value of C, and then C[i] = (C[i]-min)/(max-min). | |
| THREAD_RATIO | DOUBLE | 0 | Specifies the ratio of total number of threads that can be used by this function. The value range is from 0 to 1, where 0 means only using 1 thread, and 1 means using at most all the currently available threads. Values outside the range will be ignored and this function heuristically determines the number of threads to use. | |
| CATEGORICAL_VARIABLE | NVARCHAR | No default value | Indicates whether or not a column data is actually corresponding to a category variable even the data type of this column is INTEGER.<br>By default, 'VARCHAR' or 'NVARCHAR' is category variable, and 'INTEGER' or 'DOUBLE' is continuous variable. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| EXIT_THRESHOLD | DOUBLE | 1e-6 | Threshold (actual value) for exiting the iterations. | |

### Output Tables

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| RESULT | 1st column | ID data type | ID column name | Record ID |
| | 2nd column | INTEGER | CLUSTER_ID | Cluster ID assigned to it |
| | 3rd column | DOUBLE | DISTANCE | The distance between the given point and the cluster center |
| CENTERS | 1st column | INTEGER | CLUSTER_ID | Cluster center ID |
| | Other columns | FEATURES (in input table) data type with conversion from integer to double | FEATURES (in input table) column name | Cluster center coordinates |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_KMEDIANS_DATA_TBL;
CREATE COLUMN TABLE PAL_KMEDIANS_DATA_TBL(
    "ID" INTEGER,
    "V000" DOUBLE,
    "V001" VARCHAR(5),
    "V002" DOUBLE
);
INSERT INTO PAL_KMEDIANS_DATA_TBL VALUES (0 , 0.5, 'A', 0.5);
INSERT INTO PAL_KMEDIANS_DATA_TBL VALUES (1 , 1.5, 'A', 0.5);
INSERT INTO PAL_KMEDIANS_DATA_TBL VALUES (2 , 1.5, 'A', 1.5);
INSERT INTO PAL_KMEDIANS_DATA_TBL VALUES (3 , 0.5, 'A', 1.5);
INSERT INTO PAL_KMEDIANS_DATA_TBL VALUES (4 , 1.1, 'B', 1.2);
INSERT INTO PAL_KMEDIANS_DATA_TBL VALUES (5 , 0.5, 'B', 15.5);
INSERT INTO PAL_KMEDIANS_DATA_TBL VALUES (6 , 1.5, 'B', 15.5);
INSERT INTO PAL_KMEDIANS_DATA_TBL VALUES (7 , 1.5, 'B', 16.5);
INSERT INTO PAL_KMEDIANS_DATA_TBL VALUES (8 , 0.5, 'B', 16.5);
INSERT INTO PAL_KMEDIANS_DATA_TBL VALUES (9 , 1.2, 'C', 16.1);
INSERT INTO PAL_KMEDIANS_DATA_TBL VALUES (10, 15.5, 'C', 15.5);
INSERT INTO PAL_KMEDIANS_DATA_TBL VALUES (11, 16.5, 'C', 15.5);
INSERT INTO PAL_KMEDIANS_DATA_TBL VALUES (12, 16.5, 'C', 16.5);
INSERT INTO PAL_KMEDIANS_DATA_TBL VALUES (13, 15.5, 'C', 16.5);
```

```
INSERT INTO PAL_KMEDIANS_DATA_TBL VALUES (14, 15.6, 'D', 16.2);
INSERT INTO PAL_KMEDIANS_DATA_TBL VALUES (15, 15.5, 'D', 0.5);
INSERT INTO PAL_KMEDIANS_DATA_TBL VALUES (16, 16.5, 'D', 0.5);
INSERT INTO PAL_KMEDIANS_DATA_TBL VALUES (17, 16.5, 'D', 1.5);
INSERT INTO PAL_KMEDIANS_DATA_TBL VALUES (18, 15.5, 'D', 1.5);
INSERT INTO PAL_KMEDIANS_DATA_TBL VALUES (19, 15.7, 'A', 1.6);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL(
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR(1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 0.3, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('GROUP_NUMBER', 4, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('INIT_TYPE', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('DISTANCE_LEVEL', 2, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MAX_ITERATION', 100, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('EXIT_THRESHOLD', NULL, 1.0E-6, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('NORMALIZATION', 0, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('CATEGORY_WEIGHTS', NULL, 0.5, NULL);
CALL _SYS_AFL.PAL_KMEDIANS(PAL_KMEDIANS_DATA_TBL, "#PAL_PARAMETER_TBL", ?, ?);
```

### Expected Results

RESULT:

| | ID | CLUSTER_ID | DISTANCE |
|---|---|---|---|
| 1 | 0 | 0 | 0.9219544457292888 |
| 2 | 1 | 0 | 0.8062257748298549 |
| 3 | 2 | 0 | 0.49999999999999... |
| 4 | 3 | 0 | 0.670820393249937 |
| 5 | 4 | 0 | 0.7071067811865476 |
| 6 | 5 | 3 | 0.9219544457292896 |
| 7 | 6 | 3 | 0.6708203932499383 |
| 8 | 7 | 3 | 0.4999999999999989 |
| 9 | 8 | 3 | 0.8062257748298542 |
| 10 | 9 | 3 | 0.7071067811865476 |
| 11 | 10 | 2 | 0.7071067811865468 |
| 12 | 11 | 2 | 1.1401754250991378 |
| 13 | 12 | 2 | 0.9486832980505143 |
| 14 | 13 | 2 | 0.3162277660168385 |
| 15 | 14 | 2 | 0.7071067811865476 |
| 16 | 15 | 1 | 1.0198039027185568 |
| 17 | 16 | 1 | 1.2806248474865702 |
| 18 | 17 | 1 | 0.8000000000000007 |
| 19 | 18 | 1 | 0.1999999999999993 |
| 20 | 19 | 1 | 0.8071067811865477 |

CENTERS:

| | CLUSTER_ID | V000 | V001 | V002 |
|---|---|---|---|---|
| 1 | 0 | 1.1 | A | 1.2 |
| 2 | 1 | 15.7 | D | 1.5 |
| 3 | 2 | 15.6 | C | 16.2 |
| 4 | 3 | 1.2 | B | 16.1 |

> **Note**
>
> The wrapper procedure generation approach to call PAL functions is still supported. For more information, refer to the older versions (prior to SAP HANA Platform 2.0 SPS 02) of this document.

## 5.1.11 K-Medoids

This is a clustering algorithm related to the K-means algorithm.

Both k-medoids and k-means algorithms partition n observations into k clusters in which each observation is assigned to the cluster with the closest center. In contrast to k-means algorithm, k-medoids clustering does not calculate means, but medoids to be the new cluster centers.

A medoid is defined as the center of a cluster, whose average dissimilarity to all the objects in the cluster is minimal. Compared with the K-means algorithm, K-medoids is more robust to noise and outliers.

Given an initial set of K medoids: $m_1, \ldots, m_k$, the algorithm proceeds by alternating between two steps as below:

- Assignment step: assigns each observation to the cluster with the closest center.
- Update step: calculates the new medoid to be the center of the observations for each cluster.

The algorithm repeats until the assignments no longer change.

In PAL, the K-medoids algorithm supports multi-threads, data normalization, and different distance level measurements. It does not support categorical data, but this can be managed through data transformation. The first K and random K starting methods are supported.

**Support for Categorical Attributes**

If an attribute is of category type, it will be converted to a binary vector and then be used as a numerical attribute. For example, in the below table, "Gender" is of category type.

| Customer ID | Age | Income | Gender |
|---|---|---|---|
| T1 | 31 | 10,000 | Female |
| T2 | 27 | 8,000 | Male |

Because "Gender" has two distinct values, it will be converted into a binary vector with two dimensions:

| Customer ID | Age | Income | Gender_1 | Gender_2 |
|---|---|---|---|---|
| T1 | 31 | 10,000 | 0 | 1 |
| T2 | 27 | 8,000 | 1 | 0 |

Thus, the Euclidean distance between T1 and T2 is:

$$d(T_1, T_2)=\sqrt{(31-27)^2+(10000-8000)^2}+\gamma\sqrt{(0-1)^2+(1-0)^2}$$

Where γ is the weight to be given to the transposed categorical attributes to lessen the impact on the clustering from the 0/1 attributes. Then you can use the traditional method to update the medoid of every cluster.

## Prerequisites

- The input data contains an ID column.
- The input data does not contain null value. The algorithm will issue errors when encountering null values.

## _SYS_AFL.PAL_KMEDOIDS

This is a clustering function using the K-medoids algorithm.

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<RESULT table>` | OUT |
| 4 | `<CENTERS table>` | OUT |

### Input Table

| Table | Column | Reference for Output Table | Data Type | Description |
|---|---|---|---|---|
| DATA | 1st column | ID | INTEGER, BIGINT, VARCHAR, or NVARCHAR | Record ID |
| | Other columns | FEATURES | INTEGER, DOUBLE, DECIMAL(p, s), VARCHAR, or NVARCHAR | Attribute data |

### Parameter Table

Mandatory Parameters

The following parameter is mandatory and must be given a value.

| Name | Data Type | Description |
|---|---|---|
| GROUP_NUMBER | INTEGER | Number of groups (k). The value range is from 2 to the number of training records. |

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| DISTANCE_LEVEL | INTEGER | 2 | Computes the distance between the item and the cluster center.<br>• 1: Manhattan distance<br>• 2: Euclidean distance<br>• 3: Minkowski distance<br>• 4: Chebyshev distance<br>• 5: Jaccard distance<br>• 6: Cosine distance | When Jaccard distance is used, all columns must be categorical. |
| MINKOWSKI_POWER | DOUBLE | 3.0 | When you use the Minkowski distance, this parameter controls the value of power. | Only valid when DISTANCE_LEVEL is 3. |
| CATEGORY_WEIGHTS | DOUBLE | 0.707 | Represents the weight of category attributes (γ). | |
| MAX_ITERATION | INTEGER | 100 | Maximum iterations. | |
| INIT_TYPE | INTEGER | 4 | Controls how the initial centers are selected:<br>• 1: First k observations<br>• 2: Random with replacement<br>• 3: Random without replacement<br>• 4: Patent of selecting the init center (US 6,882,998 B1) | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| NORMALIZATION | INTEGER | 0 | Normalization type:<br>• 0: No<br>• 1: Yes. For each point X (x1,x2,...,xn), the normalized value will be X'(x1/S,x2/S,...,xn/S), where S = \|x1\|+\|x2\|+...\|xn\|.<br>• 2: For each column C, get the min and max value of C, and then C[i] = (C[i]-min)/(max-min). | |
| THREAD_RATIO | DOUBLE | 0 | Specifies the ratio of total number of threads that can be used by this function. The value range is from 0 to 1, where 0 means only using 1 thread, and 1 means using at most all the currently available threads. Values outside the range will be ignored and this function heuristically determines the number of threads to use. | |
| CATEGORICAL_VARIABLE | NVARCHAR | No default value | Indicates whether or not a column data is actually corresponding to a category variable even the data type of this column is INTEGER.<br>By default, 'VARCHAR' or 'NVARCHAR' is category variable, and 'INTEGER' or 'DOUBLE' is continuous variable. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| EXIT_THRESHOLD | DOUBLE | 1e-6 | Threshold (actual value) for exiting the iterations. | |

### Output Tables

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| RESULT | 1st column | ID (in input table) data type | ID (in input table) column name | Record ID |
| | 2nd column | INTEGER | CLUSTER_ID | Clustered ID assigned to it |
| | 3rd column | DOUBLE | DISTANCE | The distance between the given point and the cluster center |
| CENTERS | 1st column | INTEGER | CLUSTER_ID | Cluster center ID |
| | Other columns | FEATURES (in input table) data type with conversion from integer to double | FEATURES (in input table) column name | Cluster center coordinates |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_KMEDOIDS_DATA_TBL;
CREATE COLUMN TABLE PAL_KMEDOIDS_DATA_TBL(
    "ID" INTEGER,
    "V000" DOUBLE,
    "V001" VARCHAR(5),
    "V002" DOUBLE
);
INSERT INTO PAL_KMEDOIDS_DATA_TBL VALUES (0 , 0.5, 'A', 0.5);
INSERT INTO PAL_KMEDOIDS_DATA_TBL VALUES (1 , 1.5, 'A', 0.5);
INSERT INTO PAL_KMEDOIDS_DATA_TBL VALUES (2 , 1.5, 'A', 1.5);
INSERT INTO PAL_KMEDOIDS_DATA_TBL VALUES (3 , 0.5, 'A', 1.5);
INSERT INTO PAL_KMEDOIDS_DATA_TBL VALUES (4 , 1.1, 'B', 1.2);
INSERT INTO PAL_KMEDOIDS_DATA_TBL VALUES (5 , 0.5, 'B', 15.5);
INSERT INTO PAL_KMEDOIDS_DATA_TBL VALUES (6 , 1.5, 'B', 15.5);
INSERT INTO PAL_KMEDOIDS_DATA_TBL VALUES (7 , 1.5, 'B', 16.5);
INSERT INTO PAL_KMEDOIDS_DATA_TBL VALUES (8 , 0.5, 'B', 16.5);
INSERT INTO PAL_KMEDOIDS_DATA_TBL VALUES (9 , 1.2, 'C', 16.1);
INSERT INTO PAL_KMEDOIDS_DATA_TBL VALUES (10, 15.5, 'C', 15.5);
```

```
INSERT INTO PAL_KMEDOIDS_DATA_TBL VALUES (11, 16.5, 'C', 15.5);
INSERT INTO PAL_KMEDOIDS_DATA_TBL VALUES (12, 16.5, 'C', 16.5);
INSERT INTO PAL_KMEDOIDS_DATA_TBL VALUES (13, 15.5, 'C', 16.5);
INSERT INTO PAL_KMEDOIDS_DATA_TBL VALUES (14, 15.6, 'D', 16.2);
INSERT INTO PAL_KMEDOIDS_DATA_TBL VALUES (15, 15.5, 'D', 0.5);
INSERT INTO PAL_KMEDOIDS_DATA_TBL VALUES (16, 16.5, 'D', 0.5);
INSERT INTO PAL_KMEDOIDS_DATA_TBL VALUES (17, 16.5, 'D', 1.5);
INSERT INTO PAL_KMEDOIDS_DATA_TBL VALUES (18, 15.5, 'D', 1.5);
INSERT INTO PAL_KMEDOIDS_DATA_TBL VALUES (19, 15.7, 'A', 1.6);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL(
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR(1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 0.2, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('GROUP_NUMBER', 4, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('INIT_TYPE', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('DISTANCE_LEVEL', 2, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MAX_ITERATION', 100, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('EXIT_THRESHOLD', NULL, 1.0E-6, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('NORMALIZATION', 0, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('CATEGORY_WEIGHTS', NULL, 0.5, NULL);
CALL _SYS_AFL.PAL_KMEDOIDS(PAL_KMEDOIDS_DATA_TBL, "#PAL_PARAMETER_TBL", ?, ?);
```

### Expected Result

PAL_KMEDOIDS_ASSIGN_TBL:

| | ID | CLUSTER_ID | DISTANCE |
|---|---|---|---|
| 1 | 0 | 0 | 1.4142135623730951 |
| 2 | 1 | 0 | 1 |
| 3 | 2 | 0 | 0 |
| 4 | 3 | 0 | 1 |
| 5 | 4 | 0 | 1.2071067811865475 |
| 6 | 5 | 3 | 1.4142135623730951 |
| 7 | 6 | 3 | 1 |
| 8 | 7 | 3 | 0 |
| 9 | 8 | 3 | 1 |
| 10 | 9 | 3 | 1.2071067811865466 |
| 11 | 10 | 2 | 1 |
| 12 | 11 | 2 | 1.4142135623730951 |
| 13 | 12 | 2 | 1 |
| 14 | 13 | 2 | 0 |
| 15 | 14 | 2 | 1.0233345472033861 |
| 16 | 15 | 1 | 1 |
| 17 | 16 | 1 | 1.4142135623730951 |
| 18 | 17 | 1 | 1 |
| 19 | 18 | 1 | 0 |
| 20 | 19 | 1 | 0.930713578936526 |

PAL_KMEDOIDS_CENTERS_TBL:

| | CLUSTER_ID | V000 | V001 | V002 |
|---|---|---|---|---|
| 1 | 0 | 1.5 | A | 1.5 |
| 2 | 1 | 15.5 | D | 1.5 |
| 3 | 2 | 15.5 | C | 16.5 |
| 4 | 3 | 1.5 | B | 16.5 |

> **Note**
>
> The wrapper procedure generation approach to call PAL functions is still supported. For more information, refer to the older versions (prior to SAP HANA Platform 2.0 SPS 02) of this document.

## 5.1.12 Self-Organizing Maps

Self-organizing feature maps (SOMs) are one of the most popular neural network methods for cluster analysis. They are sometimes referred to as Kohonen self-organizing feature maps, after their creator, Teuvo Kohonen, or as topologically ordered maps.

SOMs aim to represent all points in a high-dimensional source space by points in a low-dimensional (usually 2-D or 3-D) target space, such that the distance and proximity relationships are preserved as much as possible. This makes SOMs useful for visualizing low-dimensional views of high-dimensional data, akin to multidimensional scaling.

SOMs can also be viewed as a constrained version of k-means clustering, in which the cluster centers tend to lie in low-dimensional manifold in the feature or attribute space. The learning process mainly includes three steps:

1. Initialize the weighted vectors in each unit.
2. Select the Best Matching Unit (BMU) for every point and update the weighted vectors of BMU and its neighbors.
3. Repeat Step 2 until convergence or the maximum iterations are reached.

An important variant is batch SOM, which updates the weighted vectors only at the end of every learning epoch. It requires that the whole set of training data is present, and is independent on the order of input vectors.

The SOM approach has many applications such as visualization, web document clustering, and speech recognition.

### Prerequisites

- The first column of the input data is an ID column.
- The input data does not contain null value. The algorithm will issue errors when encountering null values.

### _SYS_AFL.PAL_SOM

#### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<MAP table>` | OUT |
| 4 | `<ASSIGNMENT table>` | OUT |

| Position | Table Name | Parameter Type |
|---|---|---|
| 5 | `<MODEL table>` | OUT |
| 6 | `<STATISTICS table>` | OUT |
| 7 | `<PLACEHOLDER table>` | OUT |

### Input Table

| Table | Column | Reference for Output Table | Column Data Type | Description |
|---|---|---|---|---|
| DATA | 1st column | ID | INTEGER, VARCHAR, or NVARCHAR | Data ID<br>This must be the first column. |
| | Other columns | FEATURES | INTEGER, DOUBLE, or DECIMAL(p, s) | Attribute data |

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| CONVERGENCE_CRITERION | DOUBLE | 1.0e-6 | If the largest difference of the successive maps is less than this value, the calculation is regarded as convergence, and SOM is completed consequently. |
| NORMALIZATION | INTEGER | 0 | Normalization type:<br>• 0: No<br>• 1: Transform to new range (0.0, 1.0)<br>• 2: Z-score normalization |

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| RANDOM_SEED | INTEGER | -1 | • -1: Random<br>• 0: Sets every weight to zero<br>• Other value: Uses this value as seed |
| HEIGHT_OF_MAP | INTEGER | 10 | Indicates the height of the map. |
| WIDTH_OF_MAP | INTEGER | 10 | Indicates the width of the map. |
| KERNEL_FUNCTION | INTEGER | 1 | Represents the neighborhood kernel function.<br>• 1: Gaussian<br>• 2: Bubble/Flat |
| ALPHA | DOUBLE | 0.5 | Specifies the learning rate. |
| LEARNING_RATE | INTEGER | 1 | Indicates the decay function for learning rate.<br>• 1: Exponential<br>• 2: Linear |
| SHAPE_OF_GRID | INTEGER | 2 | Indicates the shape of the grid.<br>• 1: Rectangle<br>• 2: Hexagon |
| RADIUS | DOUBLE | The bigger value of HEIGHT_OF_MAP and WIDTH_OF_MAP | Specifies the scan radius. |
| BATCH_SOM | INTEGER | 0 | Indicates whether batch SOM is carried out.<br>• 0: Classical SOM<br>• 1: Batch SOM<br>For batch SOM, KERNEL_FUNCTION is always Gaussian, and the LEARNING_RATE factors take no effect. |

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| MAX_ITERATION | INTEGER | 1000 plus 500 times the number of neurons in the lattice | Maximum number of iterations.<br>Note that the training might not converge if this value is too small, for example, less than 1000. |

### Output Tables

| Table | Column | Data Type | Column | Description |
|---|---|---|---|---|
| MAP | 1st column | INTEGER | CLUSTER_ID | Unit cell ID.<br>This must be the first column. |
| | Other columns except the last one | DOUBLE | FEATURE (in input data) column with prefix "WEIGHT_" | Weight vectors used to simulate the original tuples. |
| | Last column | INTEGER | COUNT | Number of original tuples that every unit cell contains. |
| ASSIGNMENT | 1st column | ID (in input table) data type | ID (in input table) column name | ID of original tuples<br>This must be the first column. |
| | 2nd column | INTEGER | BMU | Best match unit (BMU) |
| | 3rd column | DOUBLE | DISTANCE | The distance between the tuple and its BMU |
| | 4th column | INTEGER | SECOND_BMU | Second BMU |
| | 5th column | INTEGER | IS_ADJACENT | Indicates whether the BMU and the second BMU are adjacent.<br>• 0: Not adjacent<br>• 1: Adjacent |
| MODEL | 1st column | INTEGER | ROW_INDEX | Specifies row.<br>This must be the first column. |

| Table | Column | Data Type | Column | Description |
|---|---|---|---|---|
| | 2nd column | NVARCHAR(5000) | MODEL_CONTENT | Model contents in JSON format. |
| STATISTICS | 1st column | NVARCHAR(1000) | STAT_NAME | Statistics name |
| | 2nd column | NVARCHAR(1000) | STAT_VALUE | Statistics value |
| PLACEHOLDER | 1st column | NVARCHAR(256) | PARAM_NAME | Placeholder for future features |
| | 2nd column | INTEGER | INT_VALUE | |
| | 3rd column | DOUBLE | DOUBLE_VALUE | |
| | 4th column | NVARCHAR(1000) | STRING_VALUE | |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_SOM_DATA_TBL;
CREATE COLUMN TABLE PAL_SOM_DATA_TBL(
    "TRANS_ID" INTEGER,
    "V000" DOUBLE,
    "V001" DOUBLE
);
INSERT INTO PAL_SOM_DATA_TBL VALUES (0 , 0.1, 0.2);
INSERT INTO PAL_SOM_DATA_TBL VALUES (1 , 0.22, 0.25);
INSERT INTO PAL_SOM_DATA_TBL VALUES (2 , 0.3, 0.4);
INSERT INTO PAL_SOM_DATA_TBL VALUES (3 , 0.4, 0.5);
INSERT INTO PAL_SOM_DATA_TBL VALUES (4 , 0.5, 1.0);
INSERT INTO PAL_SOM_DATA_TBL VALUES (5 , 1.1, 15.1);
INSERT INTO PAL_SOM_DATA_TBL VALUES (6 , 2.2, 11.2);
INSERT INTO PAL_SOM_DATA_TBL VALUES (7 , 1.3, 15.3);
INSERT INTO PAL_SOM_DATA_TBL VALUES (8 , 1.4, 15.4);
INSERT INTO PAL_SOM_DATA_TBL VALUES (9 , 3.5, 15.9);
INSERT INTO PAL_SOM_DATA_TBL VALUES (10, 13.1, 1.1);
INSERT INTO PAL_SOM_DATA_TBL VALUES (11, 16.2, 1.5);
INSERT INTO PAL_SOM_DATA_TBL VALUES (12, 16.3, 1.3);
INSERT INTO PAL_SOM_DATA_TBL VALUES (13, 12.4, 2.4);
INSERT INTO PAL_SOM_DATA_TBL VALUES (14, 16.9, 1.9);
INSERT INTO PAL_SOM_DATA_TBL VALUES (15, 49.0, 40.1);
INSERT INTO PAL_SOM_DATA_TBL VALUES (16, 50.1, 50.2);
INSERT INTO PAL_SOM_DATA_TBL VALUES (17, 50.2, 48.3);
INSERT INTO PAL_SOM_DATA_TBL VALUES (18, 55.3, 50.4);
INSERT INTO PAL_SOM_DATA_TBL VALUES (19, 50.4, 56.5);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL(
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR(1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MAX_ITERATION', 4000, null, null);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('HEIGHT_OF_MAP', 4, null, null);
```

```
INSERT INTO #PAL_PARAMETER_TBL VALUES ('WIDTH_OF_MAP', 4, null, null);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('NORMALIZATION', 0, null, null);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('RANDOM_SEED', 1, null, null);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SHAPE_OF_GRID', 2, null, null);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('KERNEL_FUNCTION', 1, null, null);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('LEARNING_RATE', 1, null, null);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('CONVERGENCE_CRITERION', null, 1.0e-6,
null);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('BATCH_SOM', 0, null, null);
CALL _SYS_AFL.PAL_SOM(PAL_SOM_DATA_TBL, "#PAL_PARAMETER_TBL", ?, ?, ?, ?, ?);
```

### Expected Result

MAP:

| | CLUSTER_ID | WEIGHT_V000 | WEIGHT_V001 | COUNT |
|---|---|---|---|---|
| 1 | 0 | 52.837688436447 | 53.4653266227774 | 2 |
| 2 | 1 | 50.15025126124673 | 49.24522613704447 | 2 |
| 3 | 2 | 18.597606740665... | 27.174589674066... | 0 |
| 4 | 3 | 1.2676711221847... | 15.267671122184... | 3 |
| 5 | 4 | 49.00002113918622 | 40.1000986237828 | 1 |
| 6 | 5 | 33.43099411575232 | 34.4504914886213 | 0 |
| 7 | 6 | 3.499980745400457 | 15.89997204453496 | 1 |
| 8 | 7 | 2.2000010370442... | 11.200050825324... | 1 |
| 9 | 8 | 24.814991938100... | 11.738392237008... | 0 |
| 10 | 9 | 20.69625298338181 | 8.31512780833647 | 0 |
| 11 | 10 | 9.817004542651661 | 6.453149527206734 | 0 |
| 12 | 11 | 1.1444060344046... | 4.24620510253602 | 0 |
| 13 | 12 | 16.4690145112879 | 1.5680111780752... | 3 |
| 14 | 13 | 12.748241216635... | 1.7532663311415... | 2 |
| 15 | 14 | 3.8868516028923... | 0.8463564826804... | 0 |
| 16 | 15 | 0.3059695792263... | 0.4737270589467... | 5 |

ASSIGNMENT:

| | TRANS_ID | BMU | DISTANCE | SECOND_BMU | IS_ADJACENT |
|---|---|---|---|---|---|
| 1 | 0 | 15 | 0.3425638194064086 | 14 | 1 |
| 2 | 1 | 15 | 0.239675959281097 | 14 | 1 |
| 3 | 2 | 15 | 0.07396833847719... | 14 | 1 |
| 4 | 3 | 15 | 0.09763189777148... | 14 | 1 |
| 5 | 4 | 15 | 0.5609019635109566 | 11 | 1 |
| 6 | 5 | 3 | 0.2371227750115362 | 6 | 1 |
| 7 | 6 | 7 | 0.00005083590337... | 3 | 1 |
| 8 | 7 | 3 | 0.04571993746308... | 6 | 1 |
| 9 | 8 | 3 | 0.1871412937003927 | 6 | 1 |
| 10 | 9 | 6 | 0.00003394477322... | 3 | 1 |
| 11 | 10 | 13 | 0.741950902066637 | 12 | 1 |
| 12 | 11 | 12 | 0.27747851741467... | 13 | 1 |
| 13 | 12 | 12 | 0.3168531151798187 | 13 | 1 |
| 14 | 13 | 13 | 0.7345314039567026 | 12 | 1 |
| 15 | 14 | 12 | 0.5440267175088088 | 13 | 1 |
| 16 | 15 | 4 | 0.00010086384747... | 1 | 1 |
| 17 | 16 | 1 | 0.9560953501821456 | 0 | 1 |
| 18 | 17 | 1 | 0.9465344088619049 | 0 | 1 |
| 19 | 18 | 0 | 3.93181962204711 | 1 | 1 |
| 20 | 19 | 0 | 3.892501434760311 | 1 | 1 |

MODEL:

| | ROW_INDEX | MODEL_CONTENT |
|---|---|---|
| 1 | 0 | {"Algorithm":"SOM","Cluster":[{"CellID":0,"CellWeight":[52.8376884364470,53.46532662277... |

STATISTICS: reserved for future use

PLACEHOLDER: reserved for future use

> **Note**
>
> The wrapper procedure generation approach to call PAL functions is still supported. For more information, refer to the older versions (prior to SAP HANA Platform 2.0 SPS 02) of this document.

## 5.1.13 Slight Silhouette

Silhouette refers to a method used to validate the cluster of data.

The complex of Silhouette is $O(N^2)$, where N represents the number of records. When N is very large, the silhouette costs a long time.

Considering the efficiency, PAL provides a light version of silhouette called slight silhouette. Suppose you have `N` records, for every record `i`, the following is defined:

$$S(i) = \frac{B(i) - A(i)}{\max(B(i), A(i))}$$

Where `A(i)` represents the distance between `i` and the center of the cluster it belongs to, and `B(i)` is the minimum distance between `i` and other cluster centers. Finally, below formula is derived:

$$S = \frac{1}{N}\sum_{i=1}^{N} S(i)$$

It is clear that $-1 \le S \le 1$. $-1$ stands for the poor clustering result, and 1 stands for the good result.

For attributes of category type, you can process the input data beforehand using the method described in K-means.

## Prerequisites

Input data does not contain null value.

## _SYS_AFL. PAL_SLIGHT_SILHOUETTE

This function calculates the validation value of the cluster of data.

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<RESULT table>` | OUT |

### Input Table

| Table | Column | Data Type | Description |
|---|---|---|---|
| DATA | Last column | INTEGER, BIGINT, VARCHAR, or NVARCHAR | Class label |
| | Other columns | INTEGER, DOUBLE, VARCHAR, NVARCHAR, or DECIMAL(p,s) | Attribute data |

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| DISTANCE_LEVEL | INTEGER | 2 | Computes the distance between the item and the cluster center.<br>1. Manhattan distance<br>2. Euclidean distance<br>3. Minkowski distance<br>4. Chebyshev distance | |
| MINKOWSKI_POWER | DOUBLE | 3 | When you use the Minkowski distance, this parameter controls the value of power. | Only valid when DISTANCE_LEVEL is 3. |
| NORMALIZATION | INTEGER | 0 | Normalization type:<br>• • 0: No<br>• • 1: Yes. For each point X (x1,x2,...,xn), the normalized value will be X'(x1/S,x2/S,...,xn/S), where S = \|x1\|+\|x2\|+...\|xn\|.<br>• • 2: For each column C, get the min and max value of C, and then C[i] = (C[i]-min)/(max-min). | |
| THREAD_NUMBER | INTEGER | 1 | Number of threads. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| CATEGORY_COL | INTEGER | No default value | Indicates whether the column is category variable. By default, `string` is category variable, and `integer` or `double` is continuous variable. | |
| CATEGORY_WEIGHTS | DOUBLE | 0.707 | Represents the weight of category attributes (γ). | |

### Output Tables

| Table | Column | Column Data Type | Description |
|---|---|---|---|
| RESULT | 1st column | DOUBLE | Validation value. |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFLPM_CREATOR_ERASER_EXECUTE` role;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA DM_PAL;
DROP TABLE  PAL_SLIGHT_SIL_TBL;
CREATE COLUMN TABLE PAL_SLIGHT_SIL_TBL(
    "V000" DOUBLE,
    "V001" VARCHAR(5),
    "V002" DOUBLE,
    "CLUSTER" INTEGER
);
INSERT INTO PAL_SLIGHT_SIL_TBL VALUES (0.5, 'A', 0.5, 0);
INSERT INTO PAL_SLIGHT_SIL_TBL VALUES (1.5, 'A', 0.5, 0);
INSERT INTO PAL_SLIGHT_SIL_TBL VALUES (1.5, 'A', 1.5, 0);
INSERT INTO PAL_SLIGHT_SIL_TBL VALUES (0.5, 'A', 1.5, 0);
INSERT INTO PAL_SLIGHT_SIL_TBL VALUES (1.1, 'B', 1.2, 0);
INSERT INTO PAL_SLIGHT_SIL_TBL VALUES (0.5, 'B', 15.5, 1);
INSERT INTO PAL_SLIGHT_SIL_TBL VALUES (1.5, 'B', 15.5, 1);
INSERT INTO PAL_SLIGHT_SIL_TBL VALUES (1.5, 'B', 16.5, 1);
INSERT INTO PAL_SLIGHT_SIL_TBL VALUES (0.5, 'B', 16.5, 1);
INSERT INTO PAL_SLIGHT_SIL_TBL VALUES (1.2, 'C', 16.1, 1);
INSERT INTO PAL_SLIGHT_SIL_TBL VALUES (15.5, 'C', 15.5, 2);
INSERT INTO PAL_SLIGHT_SIL_TBL VALUES (16.5, 'C', 15.5, 2);
INSERT INTO PAL_SLIGHT_SIL_TBL VALUES (16.5, 'C', 16.5, 2);
INSERT INTO PAL_SLIGHT_SIL_TBL VALUES (15.5, 'C', 16.5, 2);
INSERT INTO PAL_SLIGHT_SIL_TBL VALUES (15.6, 'D', 16.2, 2);
INSERT INTO PAL_SLIGHT_SIL_TBL VALUES (15.5, 'D', 0.5, 3);
INSERT INTO PAL_SLIGHT_SIL_TBL VALUES (16.5, 'D', 0.5, 3);
INSERT INTO PAL_SLIGHT_SIL_TBL VALUES (16.5, 'D', 1.5, 3);
INSERT INTO PAL_SLIGHT_SIL_TBL VALUES (15.5, 'D', 1.5, 3);
INSERT INTO PAL_SLIGHT_SIL_TBL VALUES (15.7, 'A', 1.6, 3);
```

```
DROP TABLE #PAL_CONTROL_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_CONTROL_TBL(
    "NAME" VARCHAR (100),
    "INTARGS" INTEGER,
    "DOUBLEARGS" DOUBLE,
    "STRINGARGS" VARCHAR(100)
);
INSERT INTO #PAL_CONTROL_TBL VALUES ('THREAD_NUMBER', 2, null, null);
INSERT INTO #PAL_CONTROL_TBL VALUES ('NORMALIZATION', 0, null, null);
INSERT INTO #PAL_CONTROL_TBL VALUES ('DISTANCE_LEVEL', 2, null, null);
INSERT INTO #PAL_CONTROL_TBL VALUES ('CATEGORY_WEIGHTS', null, 0.7, null);
INSERT INTO #PAL_CONTROL_TBL VALUES ('CATEGORY_COL', 1, null, null);
CALL _SYS_AFL.PAL_SLIGHT_SILHOUETTE(PAL_SLIGHT_SIL_TBL, #PAL_CONTROL_TBL, ?);
```

**Expected Results**

| VALIDATE_VALUE |
|---|
| D.938594381574665 6 5 |

## 5.1.14 Spectral Clustering

Spectral clustering is an algorithm evolved from graph theory, and has been widely used in clustering. Its main idea is to treat all data as points in space, which can be connected by edges. The edge weight between two points farther away is low, while the edge weight between two points closer is high. Cutting the graph

composed of all data points to make the edge weight sum between different subgraphs after cutting as low as possible, while make the edge weight sum within the subgraph as high as possible to achieve the purpose of clustering.

It performs a low-dimension embedding of the affinity matrix between samples, followed by k-means clustering of the components of the eigenvectors in the low dimensional space.

## Prerequisites

- The input data contains an ID column.
- The input data does not contain null value. The algorithm will issue errors when encountering null values.

## _SYS_AFL.PAL_SPECTRAL_CLUSTERING

This is a clustering function using spectral clustering algorithm.

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | <DATA table> | IN |
| 2 | <PARAMETER table> | IN |
| 2 | <RESULT table> | OUT |
| 3 | <STATISTICS table> | OUT |
| 4 | <PLACEHOLDER table> | OUT |

### Input Table

| Table | Column | Reference | Data Type | Description |
|---|---|---|---|---|
| DATA | 1st Column | ID | INTEGER, BIGINT, VARCHAR, or NVARCHAR | Data ID<br>This must be the first column. |
| | other Columns | FEATURES | INTEGER, DOUBLE, DECIMAL(p, s), VARCHAR, or NVARCHAR | Attribute data |

### Parameter Table

Mandatory Parameters

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| GROUP_NUMBER | INTEGER | No default value | Number of groups. The value range is from 2 to the number of training records. | |

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| THREAD_RATIO | DOUBLE | 0 | Specifies the ratio of total number of threads that can be used by this function. The value range is from 0 to 1, where 0 means only using 1 thread, and 1 means using at most all the currently available threads. Values outside the range will be ignored and this function heuristically determines the number of threads to use. | |
| N_COMPONENTS | INTEGER | GROUP_NUMBER | Number of eigenvectors to use for the spectral embedding. | |
| GAMMA | DOUBLE | 1.0 | Kernel coefficient of RBF(radia basis function) used in constructing affinity matrix construction with distance d: exp(-gamma*d^2). | |
| AFFINITY | INTEGER | 2 | How to construct the affinity matrix:<br>• 0: k-nearest neighbor graph<br>• 1: mutual k-nearest neighbor graph<br>• 2: full connected graph | |
| N_NEIGHBOURS | INTEGER | 10 | Number of neighbors to use while constructing the affinity matrix using nearest neighbors method. | Only valid when AFFINITY is 0 or 1. |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| CUT_METHOD | INTEGER | 0 | How to cut the graph:<br>• 0: RatioCut<br>• 1: NCut | |
| EIGEN_TOL | double | 1e-10 | Stopping criterion for eigendecomposition of the Laplacian matrix. | |
| KRYLOV_DIMENSION | INTEGER | N_CONPONENTS*2 | Parameter that controls the convergence speed of the algorithm. Typically a larger KRYLOY_DIMMENSION means faster convergence, but it may also result in greater memory use and more matrix operations in each iteration.<br>**Note**<br>This parameter must satifiy N_COMPONENTS< KRYLOY_DIMENSION≤ the number of training records. | |
| DISTANCE_LEVEL | INTEGER | 2 | Computes the distance between the item and the cluster center:<br>• 1: Manhattan distance<br>• 2: Euclidean distance<br>• 3: Minkowski distance<br>• 4: Chebyshev distance<br>• 6: Cosine distance | |
| CATEGORY_WEIGHTS | DOUBLE | 0.707 | Represents the weight of category attributes. | |
| MINKOWSKI_POWER | DOUBLE | 3.0 | When you use the Minkowski distance, this parameter controls the value of power. | Only valid when DISTANCE_LEVEL is 3. |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| MAX_ITERATION | INTEGER | 100 | Maximum iterations for K-Means. | |
| INIT_TYPE | INTEGER | 4 | INIT_TYPE for K-Means. | |
| EXIT_THRESHOLD | DOUBLE | 1e-6 | Threshold(actual value) for exiting the iterations of K-Means. | |

### Output Table

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| RESULT | 1st Column | ID(in input table) data type | ID(in input table) column name | Record ID |
| | 2nd Column | INTEGER | CLUSTER_ID | Cluster ID assigned to it |
| STATISTICS | 1st Column | VARCHAR or NVARCHAR | STAT_NAME | Statistic name |
| | 2nd Column | VARCHAR or NVARCHAR | STAT_VALUE | Statistic value |
| PLACEHOLDER | 1st Column | NVARCHAR(256) | PARAM_NAME | Placeholder for future features |
| | 2nd Column | INTEGER | INT_VALUE | |
| | 3rd Column | DOUBLE | DOUBLE_VALUE | |
| | 4th Column | NVARCHAR(1000) | STRING_VALUE | |

### Examples

Assume that:

- DM_PAL is a schema belonging to USER1;
- USER1 has been assigned the AFL__SYS_AFL_AFLPAL_EXECUTE or AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION role.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_SPECTRAL_DATA;
CREATE COLUMN TABLE PAL_SPECTRAL_DATA (
    "ID" INTEGER,
    "X1" DOUBLE,
    "X2" DOUBLE
);
INSERT INTO PAL_SPECTRAL_DATA VALUES (0, 0.5, 0.5);
INSERT INTO PAL_SPECTRAL_DATA VALUES (1, 1.5, 0.5);
INSERT INTO PAL_SPECTRAL_DATA VALUES (2, 1.5, 1.5);
INSERT INTO PAL_SPECTRAL_DATA VALUES (3, 0.5, 1.5);
INSERT INTO PAL_SPECTRAL_DATA VALUES (4, 1.1, 1.2);
INSERT INTO PAL_SPECTRAL_DATA VALUES (5, 0.5, 15.5);
INSERT INTO PAL_SPECTRAL_DATA VALUES (6, 1.5, 15.5);
INSERT INTO PAL_SPECTRAL_DATA VALUES (7, 1.5, 16.5);
INSERT INTO PAL_SPECTRAL_DATA VALUES (8, 0.5, 16.5);
INSERT INTO PAL_SPECTRAL_DATA VALUES (9, 1.2, 16.1);
INSERT INTO PAL_SPECTRAL_DATA VALUES (10, 15.5, 15.5);
```

```
INSERT INTO PAL_SPECTRAL_DATA VALUES (11, 16.5, 15.5);
INSERT INTO PAL_SPECTRAL_DATA VALUES (12, 16.5, 16.5);
INSERT INTO PAL_SPECTRAL_DATA VALUES (13, 15.5, 16.5);
INSERT INTO PAL_SPECTRAL_DATA VALUES (14, 15.6, 16.2);
INSERT INTO PAL_SPECTRAL_DATA VALUES (15, 15.5, 0.5);
INSERT INTO PAL_SPECTRAL_DATA VALUES (16, 16.5, 0.5);
INSERT INTO PAL_SPECTRAL_DATA VALUES (17, 16.5, 1.5);
INSERT INTO PAL_SPECTRAL_DATA VALUES (18, 15.5, 1.5);
INSERT INTO PAL_SPECTRAL_DATA VALUES (19, 15.7, 1.6);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL(
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR(1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 0.2, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('GROUP_NUMBER', 4, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('N_NEIGHTBOURS', 4, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('INIT_TYPE', 4, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('DISTANCE_LEVEL',2, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MAX_ITERATION', 100, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('EXIT_THRESHOLD', NULL, 1.0E-6, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('CATEGORY_WEIGHTS', NULL, 0.5, NULL);
CALL _SYS_AFL.PAL_SPECTRAL_CLUSTERING(PAL_SPECTRAL_DATA,
"#PAL_PARAMETER_TBL", ?,?,?);
```

### Expected Result

RESULT:

Rows (20) SQL

| | ID | CLUSTER_ID |
|---|---|---|
| 1 | 0 | 0 |
| 2 | 1 | 0 |
| 3 | 2 | 0 |
| 4 | 3 | 0 |
| 5 | 4 | 0 |
| 6 | 5 | 1 |
| 7 | 6 | 1 |
| 8 | 7 | 1 |
| 9 | 8 | 1 |
| 10 | 9 | 1 |
| 11 | 10 | 2 |
| 12 | 11 | 2 |
| 13 | 12 | 2 |
| 14 | 13 | 2 |
| 15 | 14 | 2 |
| 16 | 15 | 3 |
| 17 | 16 | 3 |
| 18 | 17 | 3 |
| 19 | 18 | 3 |
| 20 | 19 | 3 |

## 5.1.15 Incremental Clustering on SAP HANA Streaming Analytics

For information on incremental clustering on SAP HANA Streaming Analytics, see the section on DenStream clustering example in the *SAP HANA Streaming Analytics: Developer Guide*.

### Related Information

SAP HANA Streaming Analytics: Developer Guide

## 5.2 Classification Algorithms

This section describes the classification algorithms that are provided by the Predictive Analysis Library.

### 5.2.1 Area Under Curve (AUC)

Area under curve (AUC) is a traditional method to evaluate the performance of classification algorithms. Basically, it can evaluate binary classifiers, but it can also be extended to multiple-class condition easily.

In an area under curve algorithm, the curve is the receiver operating characteristic (ROC) curve. The curve can be obtained by plotting the true positive rate (TPR) against the false positive rate (FPR) at several threshold. The calculation formulas are listed below.

$$TPR = \frac{TP}{TP+FN}$$

$$FPR = \frac{FP}{FP+TN}$$

Where:

TP: true positive

FN: false negative

FP: false positive

TN: true negative

After plotting the ROC curve, you can calculate the area under the curve by using numerical integral algorithms such as Simpson's rule. The value of AUC ranges from 0.5 to 1. If the AUC equals to 1, the classifier is expected to have perfect performance.

### Prerequisite

- No missing or null data in the inputs.
- The minimum number of distinct class labels is two.

# _SYS_AFL.PAL_AUC

## Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<AUC table>` | OUT |
| 4 | `<ROC table>` | OUT |

## Input Table

| Table | Column | Data Type | Description | Constraint |
|---|---|---|---|---|
| Data | 1st column | INTEGER, VARCHAR, or NVARCHAR | ID | This must be the first column. |
| | 2nd column | INTEGER, VARCHAR, or NVARCHAR | Original label | The original label should be 0 or 1.The probability should be the probability of a sample belonging to 1. |
| | 3rd column | DOUBLE | The probability belonging to positive | |

## Parameter Table

Mandatory Parameters

None.

Optional Parameters

The following parameter is optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| POSITIVE_LABEL | VARCHAR | No default value | If original label is not 0 or 1, specifies the label value which will be mapped to 1. |

## Output Tables

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| AUC | 1st column | NVARCHAR(100) | STAT_NAME | Value name: AUC |
| | 2nd column | DOUBLE | STAT_VALUE | AUC value |
| ROC | 1st column | INTEGER | ID | ID |
| | 2nd column | DOUBLE | FPR | False positive rate for various threshold setting |
| | 3rd column | DOUBLE | TPR | True positive rate for various threshold setting |

### Example: Binary Classifier

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_AUC_DATA_TBL;
CREATE COLUMN TABLE PAL_AUC_DATA_TBL(
    ID INTEGER,
    ORIGINAL INTEGER,
    PREDICT DOUBLE
);
INSERT INTO PAL_AUC_DATA_TBL VALUES(1,0,0.07);
INSERT INTO PAL_AUC_DATA_TBL VALUES(2,0,0.01);
INSERT INTO PAL_AUC_DATA_TBL VALUES(3,0,0.85);
INSERT INTO PAL_AUC_DATA_TBL VALUES(4,0,0.3);
INSERT INTO PAL_AUC_DATA_TBL VALUES(5,0,0.5);
INSERT INTO PAL_AUC_DATA_TBL VALUES(6,1,0.5);
INSERT INTO PAL_AUC_DATA_TBL VALUES(7,1,0.2);
INSERT INTO PAL_AUC_DATA_TBL VALUES(8,1,0.8);
INSERT INTO PAL_AUC_DATA_TBL VALUES(9,1,0.2);
INSERT INTO PAL_AUC_DATA_TBL VALUES(10,1,0.95);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL(
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES('POSITIVE_LABEL',NULL,NULL,'1');
CALL _SYS_AFL.PAL_AUC(PAL_AUC_DATA_TBL,#PAL_PARAMETER_TBL,?,?);
```

### Expected Result

AUC:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | AUC | 0.66 |

ROC:

| | ID | FPR | TPR |
|---|---|---|---|
| 1 | 0 | 0 | 0 |
| 2 | 1 | 0 | 0.2 |
| 3 | 2 | 0.2 | 0.2 |
| 4 | 3 | 0.2 | 0.4 |
| 5 | 4 | 0.4 | 0.6 |
| 6 | 5 | 0.6 | 0.6 |
| 7 | 6 | 0.6 | 1 |
| 8 | 7 | 0.8 | 1 |
| 9 | 8 | 1 | 1 |

## _SYS_AFL.PAL_ROC_AUC

This procedure can output threshold.

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 3 | `<PARAMETER table>` | IN |
| 4 | `<AUC table>` | OUT |
| 5 | `<ROC table>` | OUT |

### Input Table

| Table | Column | Data Type | Description | Constraint |
|---|---|---|---|---|
| DATA | 1st column | INTEGER, VARCHAR, or NVARCHAR | ID | This must be the first column. |
| | 2nd column | INTEGER, VARCHAR, or NVARCHAR | Original label | The original label should be 0 or 1. |
| | 3rd column | DOUBLE | The probability belonging to positive | The probability should be the probability of a sample belonging to 1. |

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

The following parameter is optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| POSITIVE_LABEL | VARCHAR | No default value | If the original label is not 0 or 1, specifies the label value which will be mapped to 1. |
| NTILES | INTEGER | No default value | Used to control the population tiles in metrics output.<br>The validation value should be at least 1 and no larger than the row size of the validation data.<br>Outputs all tiles in metrics if there is no value. |

### Output Tables

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| AUC | 1st column | NVARCHAR(100) | STAT_NAME | Value name: AUC |
| | 2nd column | DOUBLE | STAT_VALUE | AUC value |
| ROC | 1st column | INTEGER | ID | ID |
| | 2nd column | DOUBLE | FPR | False positive rate for various threshold setting |
| | 3rd column | DOUBLE | TPR | True positive rate for various threshold setting |
| | 4th column | DOUBLE | THRESHOLD | Corresponding threshold. |

### Example: Binary Classifier

Assume that:

- DM_PAL is a schema belonging to USER1;

- USER1 has been assigned the AFL__SYS_AFL_AFLPAL_EXECUTE or AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION role.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_AUC_DATA_TBL;
CREATE COLUMN TABLE PAL_AUC_DATA_TBL(
    ID INTEGER,
    ORIGINAL INTEGER,
    PREDICT DOUBLE
);
INSERT INTO PAL_AUC_DATA_TBL VALUES(1,0,0.07);
INSERT INTO PAL_AUC_DATA_TBL VALUES(2,0,0.01);
INSERT INTO PAL_AUC_DATA_TBL VALUES(3,0,0.85);
INSERT INTO PAL_AUC_DATA_TBL VALUES(4,0,0.3);
INSERT INTO PAL_AUC_DATA_TBL VALUES(5,0,0.5);
INSERT INTO PAL_AUC_DATA_TBL VALUES(6,1,0.5);
INSERT INTO PAL_AUC_DATA_TBL VALUES(7,1,0.2);
INSERT INTO PAL_AUC_DATA_TBL VALUES(8,1,0.8);
INSERT INTO PAL_AUC_DATA_TBL VALUES(9,1,0.2);
INSERT INTO PAL_AUC_DATA_TBL VALUES(10,1,0.95);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL(
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES('POSITIVE_LABEL',NULL,NULL,'1');
CALL _SYS_AFL.PAL_ROC_AUC(PAL_AUC_DATA_TBL,#PAL_PARAMETER_TBL,?,?);
```

**Expected Result**

AUC:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | AUC | 0.66 |

ROC:

| | ID | FPR | TPR | THRESHOLD |
|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 1 |
| 2 | 1 | 0 | 0.2 | 0.95 |
| 3 | 2 | 0.2 | 0.2 | 0.85 |
| 4 | 3 | 0.2 | 0.4 | 0.8 |
| 5 | 4 | 0.4 | 0.6 | 0.5 |
| 6 | 5 | 0.6 | 0.6 | 0.3 |
| 7 | 6 | 0.6 | 1 | 0.2 |
| 8 | 7 | 0.8 | 1 | 0.07 |
| 9 | 8 | 1 | 1 | 0.01 |

## _SYS_AFL.PAL_MULTICLASS_AUC

This function processes multi-classifier.

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<DATA2 table>` | IN |
| 3 | `<PARAMETER table>` | IN |
| 4 | `<AUC table>` | OUT |
| 5 | `<ROC table>` | OUT |

### Input Table

| Table | Column | Data Type | Description | Constraint |
|---|---|---|---|---|
| DATA | 1st column | INTEGER, VARCHAR, or NVARCHAR | ID | This must be the first column. |
| | 2nd column | INTEGER, VARCHAR, or NVARCHAR | Original label | The original label of the samples. |
| DATA2 | 1st column | INTEGER, VARCHAR, or NVARCHAR | ID | The type must be the same as the first column type of the previous table. |
| | 2nd column | INTEGER, VARCHAR, or NVARCHAR | All of the possible predictive labels | The type must be the same as the second column type of the previous table. |
| | 3rd column | DOUBLE | The probability belonging to each possible label | For each sample data, the sum of the probabilities belonging to each possible label should be 1. |

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

The following parameter is optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| NTILES | INTEGER | No default value | Used to control the population tiles in metrics output.<br>The validation value should be at least 1 and no larger than the row size of the validation data.<br>Outputs all tiles in metrics if there is no value. |

### Output Tables

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| AUC | 1st column | NVARCHAR(100) | STAT_NAME | Value name: AUC |
| | 2nd column | DOUBLE | STAT_VALUE | AUC value |
| ROC | 1st column | INTEGER | ID | ID |
| | 2nd column | DOUBLE | FPR | False positive rate for various threshold setting |
| | 3rd column | DOUBLE | TPR | True positive rate for various threshold setting |

### Example: Multi-classifier

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_AUC_DATA_TBL;
CREATE COLUMN TABLE PAL_AUC_DATA_TBL (
    ID INTEGER,
    ORIGINAL INTEGER
);
INSERT INTO PAL_AUC_DATA_TBL VALUES(1,1);
INSERT INTO PAL_AUC_DATA_TBL VALUES(2,1);
INSERT INTO PAL_AUC_DATA_TBL VALUES(3,1);
INSERT INTO PAL_AUC_DATA_TBL VALUES(4,2);
INSERT INTO PAL_AUC_DATA_TBL VALUES(5,2);
INSERT INTO PAL_AUC_DATA_TBL VALUES(6,2);
INSERT INTO PAL_AUC_DATA_TBL VALUES(7,3);
INSERT INTO PAL_AUC_DATA_TBL VALUES(8,3);
INSERT INTO PAL_AUC_DATA_TBL VALUES(9,3);
INSERT INTO PAL_AUC_DATA_TBL VALUES(10,3);
```

```
DROP TABLE PAL_AUC_DATA2_TBL;
CREATE COLUMN TABLE PAL_AUC_DATA2_TBL (
    ID INTEGER,
    PREDICT INTEGER,
    PROB DOUBLE
);
INSERT INTO PAL_AUC_DATA2_TBL VALUES(1,1,0.9);
INSERT INTO PAL_AUC_DATA2_TBL VALUES(1,2,0.05);
INSERT INTO PAL_AUC_DATA2_TBL VALUES(1,3,0.05);
INSERT INTO PAL_AUC_DATA2_TBL VALUES(2,1,0.8);
INSERT INTO PAL_AUC_DATA2_TBL VALUES(2,2,0.05);
INSERT INTO PAL_AUC_DATA2_TBL VALUES(2,3,0.15);
INSERT INTO PAL_AUC_DATA2_TBL VALUES(3,1,0.8);
INSERT INTO PAL_AUC_DATA2_TBL VALUES(3,2,0.1);
INSERT INTO PAL_AUC_DATA2_TBL VALUES(3,3,0.1);
INSERT INTO PAL_AUC_DATA2_TBL VALUES(4,1,0.1);
INSERT INTO PAL_AUC_DATA2_TBL VALUES(4,2,0.8);
INSERT INTO PAL_AUC_DATA2_TBL VALUES(4,3,0.1);
INSERT INTO PAL_AUC_DATA2_TBL VALUES(5,1,0.2);
INSERT INTO PAL_AUC_DATA2_TBL VALUES(5,2,0.7);
INSERT INTO PAL_AUC_DATA2_TBL VALUES(5,3,0.1);
INSERT INTO PAL_AUC_DATA2_TBL VALUES(6,1,0.05);
INSERT INTO PAL_AUC_DATA2_TBL VALUES(6,2,0.9);
INSERT INTO PAL_AUC_DATA2_TBL VALUES(6,3,0.05);
INSERT INTO PAL_AUC_DATA2_TBL VALUES(7,1,0.1);
INSERT INTO PAL_AUC_DATA2_TBL VALUES(7,2,0.1);
INSERT INTO PAL_AUC_DATA2_TBL VALUES(7,3,0.8);
INSERT INTO PAL_AUC_DATA2_TBL VALUES(8,1,0);
INSERT INTO PAL_AUC_DATA2_TBL VALUES(8,2,0);
INSERT INTO PAL_AUC_DATA2_TBL VALUES(8,3,1);
INSERT INTO PAL_AUC_DATA2_TBL VALUES(9,1,0.2);
INSERT INTO PAL_AUC_DATA2_TBL VALUES(9,2,0.1);
INSERT INTO PAL_AUC_DATA2_TBL VALUES(9,3,0.7);
INSERT INTO PAL_AUC_DATA2_TBL VALUES(10,1,0.2);
INSERT INTO PAL_AUC_DATA2_TBL VALUES(10,2,0.2);
INSERT INTO PAL_AUC_DATA2_TBL VALUES(10,3,0.6);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL(
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR (1000)
);
CALL
_SYS_AFL.PAL_MULTICLASS_AUC(PAL_AUC_DATA_TBL,PAL_AUC_DATA2_TBL,#PAL_PARAMETER_TBL
,?,?);
```

### Expected Result

AUC:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | AUC | 1 |

ROC:

| | ID | FPR | TPR |
|---|---|---|---|
| 1 | 0 | 1 | 1 |
| 2 | 1 | 0.9 | 1 |
| 3 | 2 | 0.65 | 1 |
| 4 | 3 | 0.25 | 1 |
| 5 | 4 | 0.2 | 1 |
| 6 | 5 | 0 | 1 |
| 7 | 6 | 0 | 0.9 |
| 8 | 7 | 0 | 0.7 |
| 9 | 8 | 0 | 0.3 |
| 10 | 9 | 0 | 0.1 |
| 11 | 10 | 0 | 0 |

> **Note**
>
> The wrapper procedure generation approach to call PAL functions is still supported. For more information, refer to the older versions (prior to SAP HANA Platform 2.0 SPS 02) of this document.

## 5.2.2 Confusion Matrix

Confusion matrix is a traditional method to evaluate the performance of classification algorithms, including multiple-class condition.

The following is an example confusion matrix of a 3-class classification problem. The rows show the original labels and the columns show the predicted labels. For example, the number of class 0 samples classified as class 0 is a; the number of class 0 samples classified as class 2 is c.

| | Class 0 | Class 1 | Class 2 |
|---|---|---|---|
| **Class 0** | a | b | c |
| **Class 1** | d | e | f |
| **Class 2** | g | h | i |

From the confusion matrix, you can compute the precision, recall, and F1-score for each class. In the above example, the precision and recall of class 0 are:

$$\text{precision}_{\text{class0}} = \frac{a}{a+d+g}$$

$$\text{recall}_{\text{class0}} = \frac{a}{a+b+c}$$

F1-score is a combination of precision and recall as follows:

$$F1score_{class0} = \frac{2*precision_{class0}*recall_{class0}}{precision_{class0}+recall_{class0}}$$

You can also calculate the Fβ-score:

$$F\beta score_{class0} = \frac{(1+\beta^2)*precision_{class0}*recall_{class0}}{\beta^2*precision_{class0}+recall_{class0}}$$

## Prerequisite

- No missing or null data in the inputs

## _SYS_AFL.PAL_CONFUSION_MATRIX

This function evaluates the performance of classification algorithms.

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<CONFUSION MATRIX table>` | OUT |
| 4 | `<CLASSIFICATION REPORT table>` | OUT |

### Input Table

| Table | Column | Reference for Output Table | Data Type | Description |
|---|---|---|---|---|
| DATA | 1st column | | INTEGER, VARCHAR, or NVARCHAR | ID.<br>This must be the first column. |
| | 2nd column | ORIGINAL_LABEL | INTEGER, VARCHAR, or NVARCHAR | Original label |

| Table | Column | Reference for Output Table | Data Type | Description |
|---|---|---|---|---|
| | 3rd column | PREDICTED_LABEL | INTEGER, VARCHAR, or NVARCHAR | Predicted label.<br>The data type of the 2nd and 3rd columns must be the same. |

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

The following parameter is optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| BETA | DOUBLE | 1 | This parameter is used to compute the Fβ-score. The value range is (0, +∞). |

### Output Tables

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| CONFUSION MATRIX | 1st column | ORIGINAL_LABEL (in input table) data type | ORIGINAL_LABEL (in input table) column name | Original class name |
| | 2nd columns | PREDICTED_LABEL (in input table) data type | PREDICTED_LABEL (in input table) column name | Predicted class name |
| | 3rd column | INTEGER | COUNT | Stores the count of the corresponding predicted label and the corresponding original label. |
| CLASSIFICATION REPORT | 1st column | NVARCHAR(100) | CLASS | Class name |
| | 2nd column | DOUBLE | RECALL | The recall of each class |

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| | 3rd column | DOUBLE | PRECISION | The precision of each class |
| | 4th column | DOUBLE | F_MEASURE | The F-measure of each class |
| | 5th column | INTEGER | SUPPORT | The support - sample number in each class |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_CM_DATA_TBL;
CREATE COLUMN TABLE PAL_CM_DATA_TBL(
    ID INTEGER,
    ORIGINAL INTEGER,
    PREDICT INTEGER
);
INSERT INTO PAL_CM_DATA_TBL VALUES(1,1,1);
INSERT INTO PAL_CM_DATA_TBL VALUES(2,1,1);
INSERT INTO PAL_CM_DATA_TBL VALUES(3,1,1);
INSERT INTO PAL_CM_DATA_TBL VALUES(4,1,2);
INSERT INTO PAL_CM_DATA_TBL VALUES(5,1,1);
INSERT INTO PAL_CM_DATA_TBL VALUES(6,2,2);
INSERT INTO PAL_CM_DATA_TBL VALUES(7,2,1);
INSERT INTO PAL_CM_DATA_TBL VALUES(8,2,2);
INSERT INTO PAL_CM_DATA_TBL VALUES(9,2,2);
INSERT INTO PAL_CM_DATA_TBL VALUES(10,2,2);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL(
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES('BETA',NULL,1,null);
CALL _SYS_AFL.PAL_CONFUSION_MATRIX(PAL_CM_DATA_TBL,#PAL_PARAMETER_TBL,?,?);
```

### Expected Results

CONFUSION MATRIX:

| | ORIGINAL | PREDICT | COUNT |
|---|---|---|---|
| 1 | 1 | 1 | 4 |
| 2 | 1 | 2 | 1 |
| 3 | 2 | 1 | 1 |
| 4 | 2 | 2 | 4 |

CLASSIFICATION REPORT:

| | CLASS | RECALL | PRECISION | F_MEASURE | SUPPORT |
|---|---|---|---|---|---|
| 1 | 1 | 0.8 | 0.8 | 0.8000000000000002 | 5 |
| 2 | 2 | 0.8 | 0.8 | 0.8000000000000002 | 5 |

> **Note**
>
> The wrapper procedure generation approach to call PAL functions is still supported. For more information, refer to the older versions (prior to SAP HANA Platform 2.0 SPS 02) of this document.

## 5.2.3 Decision Trees

A decision tree is used as a classifier for determining an appropriate action or decision among a predetermined set of actions for a given case.

A decision tree helps effectively identify the factors to consider and how each factor has historically been associated with different outcomes of the decision. A decision tree uses a tree-like structure of conditions and their possible consequences. Each node of a decision tree can be a leaf node or a decision node.

- Leaf node: indicates the value of the dependent (target) variable.
- Decision node: contains one condition that specifies some test on an attribute value. The outcome of the condition is further divided into branches with sub-trees or leaf nodes.

The PAL_DECISION_TREE function in PAL integrates the three most popular decision trees, including C45, CHAID, and CART. Some distinctions between the implements and usages of these 3 algorithms are listed as below.

1. C45 and CHAID can generate non-binary trees, besides binary tree, while CART is restricted to binary tree.
2. Unlike C45 and CHAID, CART is able to not only classify, but also do regression.
3. C45 and CHAID treat missing independent variable as a special value, whereas CART applies surrogate split to handle it.
4. As for ordered independent variable, C45 and CHAID firstly discretise it, whereas CART uses predicate `{is` $x_m \le$ `c?}` instead of `{is` $x_m \in$ `s?}` to handle it. For large data set with a number of ordered independent variables, consequently, CART is more efficient than the other two.
5. C45 uses information gain ratio, CHAID uses chi-square statistics, and CART uses Gini index (classification) or least square (regression), for split.

In this function, the dependent variable, known as the class label or response, can have missing values, but datum of such kind is discarded before growing a tree. Meanwhile, independent variables consisting of identical value or only missing values are removed beforehand.

### Prerequisites

- The table used to store the tree model is a column table.
- The number of distinct class labels should be at least 2 for classification task.

- For prediction, the input DATA table must contain an ID column.
- The independent variables of the predict data should be the same (name, order, data type) as those used in tree model training.

## _SYS_AFL.PAL_DECISION_TREE

This function creates a decision tree from the input training data, with algorithm C45, CHAID, or CART.

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<MODEL table>` | OUT |
| 4 | `<DECISION_RULES table>` | OUT |
| 5 | `<CONFUSION_MATRIX table>` | OUT |
| 6 | `<STATISTICS table>` | OUT |
| 7 | `<CROSS_VALIDATION table>` | OUT |

### Signature

### Input Table

| Table | Column | Data Type | Description | Constraint |
|---|---|---|---|---|
| DATA | Columns | VARCHAR, NVARCHAR, INTEGER, DOUBLE, or DECIMAL(p,s) | Independent variables.<br>Discrete value: INTEGER, VARCHAR, or NVARCHAR<br>Continuous value: INTEGER or DOUBLE | If the HAS_ID parameter is 1, the first column is ID column and can be of BIGINT type as well. |

| Table | Column | Data Type | Description | Constraint |
|---|---|---|---|---|
| | Last column | INTEGER, DOUBLE, VARCHAR, NVARCHAR, or DECIMAL(p,s) | Dependent variable.<br>The varchar type is used for classification, and double type is for regression (CART only).<br>The integer type is by default used for regression (CART only), but if intentionally treated as categorical variable, which is using parameter CATEGORICAL_VARIABLE to specify, then it is solved as classification. | If DEPENDENT_VARIABLE is set to other columns, the last column is used as independent variable. |

### Parameter Table

Mandatory Parameters

The following parameters are mandatory and must be given a value.

| Name | Data Type | Description | Dependency |
|---|---|---|---|
| ALGORITHM | INTEGER | Specifies the algorithm used to grow a decision tree.<br>• 1: C45<br>• 2: CHAID<br>• 3: CART | The data type in last column should be of consistency. |

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| THREAD_RATIO | DOUBLE | -1.0 | The ratio of available threads for multi-thread task:<br>• 0: single thread<br>• 0~1: percentage<br>• Others: heuristically determined | |
| ALLOW_MISSING_DEPENDENT | INTEGER | 1 | Specifies if missing target value is allowed.<br>• 0: Not allowed. If missing target presents, an error occurs.<br>• 1: Allowed. The datum with missing target is removed. | |
| PERCENTAGE | DOUBLE | 1.0 | Specifies the percentage of the input data that will be used to build the tree model.<br>For example, if you set this parameter to 0.7, then 70% of the training data will be used to build the tree model and 30% will be used to prune the tree model. | |
| MIN_RECORDS_OF_PARENT | INTEGER | 2 | Specifies the stop condition: if the number of records in one node is less than the specified value, the algorithm stops splitting. | |
| MIN_RECORDS_OF_LEAF | INTEGER | 1 | Promises the minimum number of records in a leaf. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| MAX_DEPTH | INTEGER | Unlimited | Specifies the stop condition: if the depth of the tree model is greater than the specified value, the algorithm stops splitting. | |
| CATEGORICAL_VARIABLE | VARCHAR or NVARCHAR | Detected from input data | Indicates which variables are treated as categorical. The default behavior is:<br>• STRING: categorical<br>• INTEGER and DOUBLE: continuous | Valid only for INTEGER variables; omitted otherwise. |
| SPLIT_THRESHOLD | DOUBLE | 1e-5 for C45 and CART;<br>0.05 for CHAID | Specifies the stop condition for a node:<br>• C45: The information gain ratio of the best split is less than this value.<br>• CHAID: The p-value of the best split is greater than or equal to this value.<br>• CART: The reduction of Gini index or relative MSE of the best split is less than this value.<br>The smaller the SPLIT_THRESHOLD value is, the larger a C45 or CART tree grows. On the contrary, CHAID will grow a larger tree with larger SPLIT_THRESHOLD value. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| `<name of target value>`_PRIOR_ | DOUBLE | Detected from input data | Specifies the priori probability of every class label. | Valid only for classification. |
| DISCRETIZATION_TYPE | INTEGER | 0 | Specifies the strategy for discretizing continuous attributes:<br>• 0: MDLPC<br>• 1: Equal Frequency | Valid only for C45 and CHAID. |
| `<column name>`_BIN_ | INTEGER | 10 | Specifies the number of bins for discretisation | Only valid when DISCRETIZATION_TYPE is 1. |
| MAX_BRANCH | INTEGER | 10 | Specifies the maximum number of branches. | Valid only for CHAID. |
| MERGE_THRESHOLD | DOUBLE | 0.05 | Specifies the merge condition for CHAID: if the metric value is greater than or equal to the specified value, the algorithm will merge two branches. | Valid only for CHAID. |
| USE_SURROGATE | INTEGER | 1 | Indicates whether to use surrogate split when NULL values are encountered.<br>• 0: Does not use surrogate split.<br>• 1: Uses surrogate split. | Valid only for CART. |
| MODEL_FORMAT | INTEGER | 1 | Specifies the tree model format for store:<br>• 1: JSON<br>• 2: PMML | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| IS_OUTPUT_RULES | INTEGER | 1 | Specifies whether to output decision rules or not.<br>• 0: Does not output decision rules.<br>• 1: Outputs decision rules. | |
| IS_OUTPUT_CONFUSION_MATRIX | INTEGER | 1 | Specifies whether to output confusion matrix or not.<br>• 0: Does not output confusion matrix.<br>• 1: Outputs confusion matrix. | Valid only for classification. |
| HAS_ID | INTEGER | 0 | Specifies whether the first column of DATA table is used as ID column.<br>• 0: No<br>• 1: Yes; the first column is discarded during training | |
| DEPENDENT_VARIABLE | VARCHAR or NVARCHAR | The last column name of the DATA table | Specifies the variable used as dependent. | Cannot be the first column name if HAS_ID is 1. |

The following parameters are for model evaluation and parameter search. There are two styles for evaluated parameters, `<param>`_VALUES and `<param>`_RANGE, where the former is a discrete set: '{v1, v2, ...}', and the latter is '[low, step, high]', random search applied if a step is omitted. The parameter priority is `<param>`_VALUES, `<param>`_RANGE, `<param>`.

**Parameters for Model Evaluation and Parameter Search**

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| RESAMPLING_METHOD | VARCHAR, or NVARCHAR | No default | The resampling method for model evaluation or parameter search:<br>• cv: cross validation<br>• stratified_cv: stratified cross validation<br>• bootstrap: bootstrap method<br>• stratified_bootstrap: stratified bootstrap method | |
| FOLD_NUM | INTEGER | No default | The fold number for cross validation. | Valid only for cross validation; mandatory for cross validation. |
| REPEAT_TIMES | INTEGER | 1 | The number of repeated times for model evaluation or parameter search. | |
| EVALUATION_METRIC | VARCHAR, or NVARCHAR | ERROR_RATE for classification, RMSE for regression | The evaluation metric, candidates are:<br>• RMSE<br>• MAE<br>• ERROR_RATE<br>• NLL<br>• AUC | Regression uses only RMSE and MAE; multi-classification does not use RMSE or MAE |
| TIME_OUT | INTEGER | 0 | The time allocated (seconds) for program running. 0 indicates unlimited. | |
| PARAM_SEARCH_STRATEGY | VARCHAR, or NVARCHAR | No default | The search strategy for parameters:<br>• random: random search<br>• grid: grid search | If not specified, only model evaluation is carried out. |
| RANDOM_SEARCH_TIMES | INTEGER | No default | Specifies the number of search times for random search. | Mandatory for random search. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| PROGRESS_INDICATOR_ID | NVARCHAR | No default | Sets an ID of progress indicator for model evaluation or parameter selection. | No progress indicator is active if no value is provided. |
| DISCRETIZATION_TYPE_VALUES | VARCHAR, or NVARCHAR | No default | The discrete set for the DISCRETIZATION_TYPE candidate. | |
| DISCRETIZATION_TYPE_RANGE | VARCHAR, or NVARCHAR | No default | The range for the DISCRETIZATION_TYPE candidate. | |
| MIN_RECORDS_OF_LEAF_VALUES | VARCHAR, or NVARCHAR | No default | The discrete set for the MIN_RECORDS_OF_LEAF candidate. | |
| MIN_RECORDS_OF_LEAF_RANGE | VARCHAR, or NVARCHAR | No default | The range for the MIN_RECORDS_OF_LEAF candidate. | |
| MIN_RECORDS_OF_PARENT_VALUES | VARCHAR, or NVARCHAR | No default | The discrete set for the MIN_RECORDS_OF_PARENT candidate. | |
| MIN_RECORDS_OF_PARENT_RANGE | VARCHAR, or NVARCHAR | No default | The range for the MIN_RECORDS_OF_PARENT candidate. | |
| MAX_DEPTH_VALUES | VARCHAR, or NVARCHAR | No default | The discrete set for the MAX_DEPTH candidate. | |
| MAX_DEPTH_RANGE | VARCHAR, or NVARCHAR | No default | The range for the MAX_DEPTH candidate. | |
| SPLIT_THRESHOLD_VALUES | VARCHAR, or NVARCHAR | No default | The discrete set for the SPLIT_THRESHOLD candidate. | |
| SPLIT_THRESHOLD_RANGE | VARCHAR, or NVARCHAR | No default | The range for the SPLIT_THRESHOLD candidate. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| MAX_BRANCH_VALUES | VARCHAR, or NVARCHAR | No default | The discrete set for the MAX_BRANCH candidate. | |
| MAX_BRANCH_RANGE | VARCHAR, or NVARCHAR | No default | The range for the MAX_BRANCH candidate. | |
| MERGE_THRESHOLD_VALUES | VARCHAR, or NVARCHAR | No default | The discrete set for the MERGE_THRESHOLD candidate. | |
| MERGE_THRESHOLD_RANGE | VARCHAR, or NVARCHAR | No default | The range for the MERGE_THRESHOLD candidate. | |

### Output Tables

| Table | Column | Data Type | Column Name | Description | Constraint |
|---|---|---|---|---|---|
| MODEL | 1st column | INTEGER | ROW_INDEX | Indicates the row | Must be the first column |
| | 2nd column | NVARCHAR(5000) | MODEL_CONTENT | Tree model content | JSON or PMML |
| RULES | 1st column | INTEGER | ROW_INDEX | Indicates the row | Must be the first column |
| | 2nd column | NVARCHAR(5000) | RULES_CONTENT | Decision rules content | |
| CONFUSION (only for classification) | 1st column | NVARCHAR(1000) | ACTUAL_CLASS | The actual class name | |
| | 2nd column | NVARCHAR(1000) | PREDICTED_CLASS | The predicted class name | |
| | 3rd column | INTEGER | COUNT | Number of records | |
| STATS | 1st column | NVARCHAR(1000) | STAT_NAME | Statistics name | |
| | 2nd column | NVARCHAR(1000) | STAT_VALUE | Statistics value | |
| CV | 1st column | NVARCHAR(256) | PARM_NAME | Parameter name | |
| | 2nd column | INTEGER | INT_VALUE | Integer value | |
| | 3rd column | DOUBLE | DOUBLE_VALUE | Double value | |

| Table | Column | Data Type | Column Name | Description | Constraint |
|---|---|---|---|---|---|
| | 4th column | NVARCHAR(1000) | STRING_VALUE | String value | |

### Examples

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

#### Example 1: C45 classification with cross validation

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_DT_DATA_TBL;
CREATE COLUMN TABLE PAL_DT_DATA_TBL (
    "OUTLOOK" VARCHAR(20),
    "TEMP" INTEGER,
    "HUMIDITY" DOUBLE,
    "WINDY" VARCHAR(10),
    "CLASS" VARCHAR(20)
);
INSERT INTO PAL_DT_DATA_TBL VALUES ('Sunny', 75, 70, 'Yes', 'Play');
INSERT INTO PAL_DT_DATA_TBL VALUES ('Sunny', 80, 90, 'Yes', 'Do not Play');
INSERT INTO PAL_DT_DATA_TBL VALUES ('Sunny', 85, 85, 'No', 'Do not Play');
INSERT INTO PAL_DT_DATA_TBL VALUES ('Sunny', 72, 95, 'No', 'Do not Play');
INSERT INTO PAL_DT_DATA_TBL VALUES ('Sunny', 69, 70, 'No', 'Play');
INSERT INTO PAL_DT_DATA_TBL VALUES ('Overcast', 72, 90, 'Yes', 'Play');
INSERT INTO PAL_DT_DATA_TBL VALUES ('Overcast', 83, 78, 'No', 'Play');
INSERT INTO PAL_DT_DATA_TBL VALUES ('Overcast', 64, 65, 'Yes', 'Play');
INSERT INTO PAL_DT_DATA_TBL VALUES ('Overcast', 81, 75, 'No', 'Play');
INSERT INTO PAL_DT_DATA_TBL VALUES ('Rain', 71, 80, 'Yes', 'Do not Play');
INSERT INTO PAL_DT_DATA_TBL VALUES ('Rain', 65, 70, 'Yes', 'Do not Play');
INSERT INTO PAL_DT_DATA_TBL VALUES ('Rain', 75, 80, 'No', 'Play');
INSERT INTO PAL_DT_DATA_TBL VALUES ('Rain', 68, 80, 'No', 'Play');
INSERT INTO PAL_DT_DATA_TBL VALUES ('Rain', 70, 96, 'No', 'Play');
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL ("NAME" VARCHAR
(50),"INT_VALUE" INTEGER,"DOUBLE_VALUE" DOUBLE,"STRING_VALUE" VARCHAR (100));
INSERT INTO #PAL_PARAMETER_TBL VALUES ('ALGORITHM', 1, NULL, NULL); -- C45
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MODEL_FORMAT', 1, NULL, NULL); -- JSON
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SPLIT_THRESHOLD', NULL, 1e-5, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MIN_RECORDS_OF_PARENT', 2, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MIN_RECORDS_OF_LEAF', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('IS_OUTPUT_RULES', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('IS_OUTPUT_CONFUSION_MATRIX', 1, NULL,
NULL);
--INSERT INTO #PAL_PARAMETER_TBL VALUES ('CATEGORICAL_VARIABLE', NULL, NULL,
'TEM');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 0.4, NULL);
--INSERT INTO #PAL_PARAMETER_TBL VALUES ('HAS_ID', 0, NULL, NULL);
--INSERT INTO #PAL_PARAMETER_TBL VALUES ('DEPENDENT_VARIABLE', NULL, NULL,
'CLASS');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('RESAMPLING_METHOD', NULL, NULL, 'cv');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('EVALUATION_METRIC', NULL, NULL, 'AUC');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PARAM_SEARCH_STRATEGY', NULL, NULL,
'grid');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FOLD_NUM', 5, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SPLIT_THRESHOLD_VALUES', NULL, NULL,
'{1e-3, 1e-4, 1e-5}');
--INSERT INTO #PAL_PARAMETER_TBL VALUES ('SPLIT_THRESHOLD_RANGE', NULL, NULL,
'[1e-5, 3e-5, 1e-4]');
```

```
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PROGRESS_INDICATOR_ID', NULL, NULL,
'CV');
DROP TABLE PAL_DT_MODEL_TBL; -- for predicted followed
CREATE COLUMN TABLE PAL_DT_MODEL_TBL (
    "ROW_INDEX" INTEGER,
    "MODEL_CONTENT" NVARCHAR(5000)
);
CALL _SYS_AFL.PAL_DECISION_TREE (PAL_DT_DATA_TBL, #PAL_PARAMETER_TBL,
PAL_DT_MODEL_TBL, ?, ?, ?, ?) WITH OVERVIEW;
```

**Expected Results**

MODEL:

| | ROW_INDEX | MODEL_CONTENT |
|---|---|---|
| 1 | 0 | {"CurrentVersion":"VERSION1.3","DataDictionary":[{"DataType":"String","ID":0,"Name":"OU... |

RULES:

| | ROW_INDEX | RULES_CONTENT |
|---|---|---|
| 1 | 0 | (TEMP>=84) => Do not Play |
| 2 | 1 | (TEMP<84) && (OUTLOOK=Overcast) => Play |
| 3 | 2 | (TEMP<84) && (OUTLOOK=Sunny) && (HUMIDITY<82.5) => Play |
| 4 | 3 | (TEMP<84) && (OUTLOOK=Sunny) && (HUMIDITY>=82.5) => Do not Play |
| 5 | 4 | (TEMP<84) && (OUTLOOK=Rain) && (WINDY=Yes) => Do not Play |
| 6 | 5 | (TEMP<84) && (OUTLOOK=Rain) && (WINDY=No) => Play |

CONFUSION:

| | ACTUAL_CLASS | PREDICTED_CLASS | COUNT |
|---|---|---|---|
| 1 | Play | Play | 9 |
| 2 | Play | Do not Play | 0 |
| 3 | Do not Play | Play | 0 |
| 4 | Do not Play | Do not Play | 5 |

STATISTICS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | timeout | FALSE |
| 2 | TEST_AUC | 0.5, 0.75, 0.5, 0, 0.5 |
| 3 | TEST_AUC.... | 0.45 |
| 4 | TEST_AUC.... | 0.075 |
| 5 | EVAL_RESU... | {"candidates":[{"... |

CV:

| | PARAM_NAME | INT_VALUE | DOUBLE_VALUE | STRING_VALUE |
|---|---|---|---|---|
| 1 | DISCRETIZATION_TYPE | 0 | ? | ? |
| 2 | SPLIT_THRESHOLD | ? | 0.00001 | ? |
| 3 | MAX_DEPTH | 14 | ? | ? |
| 4 | MIN_RECORDS_OF_L... | 1 | ? | ? |
| 5 | MIN_RECORDS_OF_P... | 2 | ? | ? |

### Example 2: CART regression

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_DT_DATA_TBL;
CREATE COLUMN TABLE PAL_DT_DATA_TBL (
    "OUTLOOK" VARCHAR(20),
    "TEMP" INTEGER,
    "HUMIDITY" DOUBLE,
    "WINDY" VARCHAR(10),
    "CLASS" INTEGER -- for regression
);
INSERT INTO PAL_DT_DATA_TBL VALUES ('Sunny', 75, 70, 'Yes', 1);
INSERT INTO PAL_DT_DATA_TBL VALUES ('Sunny', 80, 90, 'Yes', 0);
INSERT INTO PAL_DT_DATA_TBL VALUES ('Sunny', 85, 85, 'No', 0);
INSERT INTO PAL_DT_DATA_TBL VALUES ('Sunny', 72, 95, 'No', 0);
INSERT INTO PAL_DT_DATA_TBL VALUES ('Sunny', 69, 70, 'No', 1);
INSERT INTO PAL_DT_DATA_TBL VALUES ('Overcast', 72, 90, 'Yes', 1);
INSERT INTO PAL_DT_DATA_TBL VALUES ('Overcast', 83, 78, 'No', 0);
INSERT INTO PAL_DT_DATA_TBL VALUES ('Overcast', 64, 65, 'Yes', 1);
INSERT INTO PAL_DT_DATA_TBL VALUES ('Overcast', 81, 75, 'No', 1);
INSERT INTO PAL_DT_DATA_TBL VALUES ('Rain', 71, 80, 'Yes', 0);
INSERT INTO PAL_DT_DATA_TBL VALUES ('Rain', 65, 70, 'Yes', 0);
INSERT INTO PAL_DT_DATA_TBL VALUES ('Rain', 75, 80, 'No', 1);
INSERT INTO PAL_DT_DATA_TBL VALUES ('Rain', 68, 80, 'No', 1);
INSERT INTO PAL_DT_DATA_TBL VALUES ('Rain', 70, 96, 'No', 0);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL ("NAME" VARCHAR
(50),"INT_VALUE" INTEGER,"DOUBLE_VALUE" DOUBLE,"STRING_VALUE" VARCHAR (100));
INSERT INTO #PAL_PARAMETER_TBL VALUES ('ALGORITHM', 3, NULL, NULL); -- CART
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MODEL_FORMAT', 2, NULL, NULL); -- PMML
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SPLIT_THRESHOLD', NULL, 1e-5, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MIN_RECORDS_OF_PARENT', 2, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MIN_RECORDS_OF_LEAF', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('IS_OUTPUT_RULES', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('IS_OUTPUT_CONFUSION_MATRIX', 1, NULL,
NULL);
--INSERT INTO #PAL_PARAMETER_TBL VALUES ('CATEGORICAL_VARIABLE', NULL, NULL,
'TEM');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 0.4, NULL);
--INSERT INTO #PAL_PARAMETER_TBL VALUES ('HAS_ID', 0, NULL, NULL);
--INSERT INTO #PAL_PARAMETER_TBL VALUES ('DEPENDENT_VARIABLE', NULL, NULL,
'CLASS');
DROP TABLE PAL_DT_MODEL_TBL; -- for predicted followed
CREATE COLUMN TABLE PAL_DT_MODEL_TBL (
    "ROW_INDEX" INTEGER,
    "MODEL_CONTENT" NVARCHAR(5000)
);
CALL _SYS_AFL.PAL_DECISION_TREE (PAL_DT_DATA_TBL, #PAL_PARAMETER_TBL,
PAL_DT_MODEL_TBL, ?, ?, ?, ?) WITH OVERVIEW;
```

### Expected Results

MODEL:

| | ROW_INDEX | MODEL_CONTENT |
|---|---|---|
| 1 | 1 | <PMML version="4.0" xmlns="http://www.dmg.org/PMML-4_0" xmlns:xsi="http://www.... |

RULES:

| | ROW_INDEX | RULES_CONTENT |
|---|---|---|
| 1 | 0 | (HUMIDITY<76.5) && (OUTLOOK=Rain) => 0 |
| 2 | 1 | (HUMIDITY<76.5) && ((OUTLOOK=Sunny)\|\|(OUTLOOK=Overcast)) => 1 |
| 3 | 2 | (HUMIDITY>=76.5) && (OUTLOOK=Sunny) => 0 |
| 4 | 3 | (HUMIDITY>=76.5) && ((OUTLOOK=Overcast)\|\|(OUTLOOK=Rain)) && (TEMP<69) => 1 |
| 5 | 4 | (HUMIDITY>=76.5) && ((OUTLOOK=Overcast)\|\|(OUTLOOK=Rain)) && (TEMP>=69) && (TEMP<71.5) => 0 |
| 6 | 5 | (HUMIDITY>=76.5) && ((OUTLOOK=Overcast)\|\|(OUTLOOK=Rain)) && (TEMP>=69) && (TEMP>=71.5) && (TEMP<79) => 1 |
| 7 | 6 | (HUMIDITY>=76.5) && ((OUTLOOK=Overcast)\|\|(OUTLOOK=Rain)) && (TEMP>=69) && (TEMP>=71.5) && (TEMP>=79) => 0 |

CONFUSION:

Empty

STATS:

Empty

CV:

Empty

## _SYS_AFL.PAL_DECISION_TREE_PREDICT

This algorithm uses decision tree models to perform scoring, and currently supports C45, CHAID, and CART.

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<MODEL table>` | IN |
| 3 | `<PARAMETER table>` | IN |
| 4 | `<RESULT table>` | OUT |

### Input Tables

| Table | Column | Reference for Output Table | Data Type | Description | Constraint |
|---|---|---|---|---|---|
| DATA | 1st column | ID | INTEGER, BIGINT, VARCHAR, or NVARCHAR | Data ID | This must be the first column. |
| | Other columns | | INTEGER, DOUBLE, VARCHAR, NVARCHAR, or DECIMAL(p,s) | Independent variables | |
| MODEL | 1st column | | INTEGER | Row index | This must be the first column. |
| | 2nd column | | VARCHAR or NVARCHAR | Model content | JSON or PMML format. |

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Table | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| THREAD_RATIO | DOUBLE | -1 | The ratio of available threads.<br>• 0: single thread<br>• 0~1: percentage<br>• Others: heuristically determined | |
| VERBOSE | INTEGER | 0 | Specifies whether to output all classes and the corresponding confidences for each data.<br>• 0: no<br>• 1: yes | Valid only for classification |

### Output Tables

| Table | Column | Data Type | Column Name | Description | Constraint |
|---|---|---|---|---|---|
| RESULT | 1st column | ID column data type | ID column name | ID | |
| | 2nd column | NVARCHAR(100) | SCORE | Scoring result | |
| | 3rd column | DOUBLE | CONFIDENCE | The confidence of a class. | All 0s if for regression. |

### Examples

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

**Example:**

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_DT_SCORING_DATA_TBL;
CREATE COLUMN TABLE PAL_DT_SCORING_DATA_TBL (
    "ID" INTEGER,
    "OUTLOOK" VARCHAR(20),
    "HUMIDITY" DOUBLE,
    "TEMP" INTEGER,
    "WINDY" VARCHAR(10)
);
INSERT INTO PAL_DT_SCORING_DATA_TBL VALUES (0, 'Overcast', 75, 70, 'Yes');
INSERT INTO PAL_DT_SCORING_DATA_TBL VALUES (1, 'Rain', 78, 70, 'Yes');
INSERT INTO PAL_DT_SCORING_DATA_TBL VALUES (2, 'Sunny', 66, 70, 'Yes');
INSERT INTO PAL_DT_SCORING_DATA_TBL VALUES (3, 'Sunny', 69, 70, 'Yes');
INSERT INTO PAL_DT_SCORING_DATA_TBL VALUES (4, 'Rain', null, 70, 'Yes');
INSERT INTO PAL_DT_SCORING_DATA_TBL VALUES (5, null, 70, 70, 'Yes');
INSERT INTO PAL_DT_SCORING_DATA_TBL VALUES (6, '***', 70, 70, 'Yes');
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL ("NAME" VARCHAR
(50),"INT_VALUE" INTEGER,"DOUBLE_VALUE" DOUBLE,"STRING_VALUE" VARCHAR (100));
INSERT INTO #PAL_PARAMETER_TBL VALUES ('VERBOSE', 0, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 0.5, NULL);
CALL _SYS_AFL.PAL_DECISION_TREE_PREDICT (PAL_DT_SCORING_DATA_TBL,
PAL_DT_MODEL_TBL, #PAL_PARAMETER_TBL, ?);
```

### Expected Results

RESULT (C4.5 classification):

| | ID | SCORE | CONFIDENCE |
|---|---|---|---|
| 1 | 0 | Play | 1 |
| 2 | 1 | Do not Play | 1 |
| 3 | 2 | Play | 1 |
| 4 | 3 | Play | 1 |
| 5 | 4 | Do not Play | 1 |
| 6 | 5 | Play | 0.692307692... |
| 7 | 6 | Play | 0.692307692... |

RESULT (CART regression):

| | ID | SCORE | CONFIDENCE |
|---|---|---|---|
| 1 | 0 | 1 | 0 |
| 2 | 1 | 0 | 0 |
| 3 | 2 | 1 | 0 |
| 4 | 3 | 1 | 0 |
| 5 | 4 | 0.5 | 0 |
| 6 | 5 | 0.8 | 0 |
| 7 | 6 | 0.8 | 0 |

## 5.2.4 Hybrid Gradient Boosting Tree

Gradient boosting tree (GBT) is an ensemble machine learning technique for regression and classification problems.

GBT builds the model in a stage-wise fashion and allows optimization of some loss functions. During each iteration, it builds a new classification or regression tree based on negative gradient, for example, residual, as the training sample to be fitted.

Gradient boosting tree, is endowed with regression, and the final score is the weighted summation of each iteration's regression tree. To achieve classification, different loss criteria are adopted – logit loss for instance – which are able to connect the regression values and categorical labels.

In the $k_{th}$ iteration, the result of the regression tree to sample variable xi can be denoted as $f_k(x_i)$ in the following equations.

$$\hat{y}_i^{(0)}=0$$

$$\hat{y}_i^{(1)}=\beta f_1(x_i)=\hat{y}_i^{(0)}+\beta f_1(x_i)$$

$$\hat{y}_i^{(2)}=\beta f_1(x_i)+\beta f_2(x_i)=\hat{y}_i^{(1)}+\beta f_2(x_i)$$

...

$$\hat{y}_i^{(t)}=\sum_{k=1}^{t}\beta f_k(x_i)=\hat{y}_i^{(t-1)}+\beta f_t(x_i)$$

As is shown, the final score $\hat{y}_i^{(t)}$ is then the weighted summation of all the results of every iteration.

For simplicity, we just rewrote the regression function $f_t(x)$ as $w_{q(x)}$ in a specific iteration, where $w$ is the score vector on tree leaves, $q(\cdot)$ is a function assigning each data point to the corresponding leaf and $t$ is the number of leaves, and $\beta$ is the learning rate.

$$f_t(x)=w_{q(x)}$$

To be more precise, one can carry out a second-order expansion to the object function at $w_{q(xi)}$, rendering the Hessian.

$$Obj^{(t)} \approx \sum_{i=1}^{n}\left[g_i w_{q(x_i)} + \frac{1}{2} h_i w_{q(x_i)}^2\right] + \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2 + \alpha \sum_{k=1}^{T} \|w_j\|_1$$

$$Gain = \frac{1}{2}\left[\frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{(G_L+G_R)^2}{H_L+H_R+\lambda}\right] - \gamma$$

$$G_j = \sum_{i \in I_j} g_i$$

$$H_j = \sum_{i \in I_j} h_i$$

Here $g_i$ and $h_i$ denote the first (gradient) and second (Hessian) order derivative of loss function for sample $x_i$, respectively.

The tree grows via selecting the variable whose corresponding split leads to maximum gain.

Boosting tree is realized in PAL as hybrid gradient boosting tree (HGBT), which supports mixed feature types,that is, both continuous and categorical type of features, including:

- classification and regression tasks
- loss criteria such as square loss and logistic loss
- L1 and L2 regularization
- model evaluation and cross-validation

Specifically, the cross validation is carried out by selecting parameters that render some optimal information. In PAL, for example, classification uses relative mean square error (RMSE), mean absolute error (MAE), log-likelihood, error rate, and area under the curve (AUC), whereas regression, on the other side, employs RMSE and MAE.

PAL processes categorical feature directly, instead of applying the popular one-hot encoding scheme, which is proved to be much more effective and efficient in the case of a number of categorical features.

**Prerequisites**

- The table used to store the tree model is a column table.
- The scoring data (predict) must have an ID column.
- All variables in the training stage should be provided in the scoring data.

## _AFL_SYS.PAL_HGBT

This function generates gradient boosting tree model.

**Overview of all tables**

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |

| Position | Table Name | Parameter Type |
|---|---|---|
| 3 | `<MODEL table>` | OUT |
| 4 | `<VARIABLE_IMPORTANCE table>` | OUT |
| 5 | `<CONFUSION_MATRIX table>` | OUT |
| 6 | `<STATISTICS table>` | OUT |
| 7 | `<CROSS_VALIDATION table>` | OUT |

### Signature

### Input Table

| Table | Column | Data Type | Description | Constraint |
|---|---|---|---|---|
| DATA | Columns | VARCHAR, NVARCHAR, INTEGER, DOUBLE, or DECIMAL(p, s) | Independent variables.<br>Discrete value: INTEGER, VARCHAR, and NVARCHAR<br>Continuous value: INTEGER, DOUBLE, and DECIMAL(p, s), | If the HAS_ID parameter is 1, the first column is an ID column and can be of BIGINT type, but discarded in training. |
| | Last column | VARCHAR, NVARCHAR, INTEGER, DOUBLE, or DECIMAL(p, s) | Dependent variable.<br>The VARCHAR type is used for classification, and DOUBLE type is for regression.<br>The INTEGER type is by default used for regression, but if intentionally treated as categorical variable, that is, using parameter CATEGORICAL_VARIABLE to indicate, then it is solved as classification. | If DEPENDENT_VARIABLE is set to other columns, the last column is used as independent variable. |

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| THREAD_RATIO | DOUBLE | -1 | The ratio of available threads.<br>• 0: single thread<br>• 0~1: percentage<br>• Others: heuristically determined | |
| SEED | INTEGER | 0 | The seed for random number generating.<br>• 0: current time<br>• Others: the seed | |
| SPLIT_METHOD | VARCHAR or NVARCHAR | exact | The method to find split point for numeric feature.<br>• exact: the exact method, trying all possible points<br>• sketch: the sketch method, accounting for the distribution of the sum of hessian<br>• sampling: samples the split point randomly<br>• histogram:builds histogram upon data and uses it as split point | The exact method comparably has the highest test accuracy, but costs more time. The other three methods have relative higher computational efficiency but might lead to lower test accuracy, and are considered to be adopted as the training data set is huge.<br>Valid only for numeric features. |
| SKETCH_EPS | DOUBLE | 0.1 | The epsilon of the sketch method. It indicates that the sum of hessian between two split points is not larger than this value. That is, the number of bins is approximately 1/eps. | The less is this value, the more split points are tried. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| MAX_BIN_NUM | INTEGER | 256 | The maximum bin number for histogram method. Decreasing this number gains better performance in terms of running time at a cost of accuracy. | Only valid when SPLIT_METHOD is set to histogram. |
| ITER_NUM | INTEGER | 10 | Total iteration number, which is equivalent to the number of trees in the final model. | |
| MAX_DEPTH | INTEGER | 6 | The maximum depth of each tree. | |
| NODE_SIZE | INTEGER | 1 | The minimum number of data in a leaf. | |
| ETA | DOUBLE | 0.3 | Learning rate of each iteration.<br>Range: (0, 1] | |
| GAMMA | DOUBLE | 0.0 | The minimum loss change value to make a split in tree growth (gamma in the equation). | |
| MIN_CHILD_HESSIAN | DOUBLE | 1.0 | The minimum summation of sample weights (hessian) in the leaf node. | |
| NODE_WEIGHT_CONSTRAINT | DOUBLE | 0.0 | The maximum weight constraint assigned to each tree node. 0 for no constraint. | |
| ROW_SAMPLE_RATE | DOUBLE | 1.0 | The sample rate of row (data points).<br>Range: (0, 1] | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| COL_SAMPLE_RATE_BY-SPLIT | DOUBLE | 1.0 | The sample rate of feature set in each split.<br>Range: (0, 1] | |
| COL_SAMPLE_RATE_BY-TREE | DOUBLE | 1.0 | The sample rate of feature set in each tree growth.<br>Range: (0, 1] | |
| LAMBDA | DOUBLE | 1.0 | L2 regularization.<br>Range: [0, 1] | |
| ALPHA | DOUBLE | 0.0 | L1 regularization.<br>Range: [0, 1] | |
| BASE_SCORE | DOUBLE | 0.5 for binary classification; 0 otherwise. | Initial prediction score of all instances.<br>Global bias for sufficient number of iterations.<br>Range: (0, 1), only for binary classification. | |
| START_FROM_AVERAGE | INTEGER | 0 | Indicates whether to adopt the prior distribution as the initial point. To be specific, use average value if it is a regression problem, and use frequencies of labels if it is a classification problem.<br>• 0: no<br>• 1: yes | BASE_SCORE is ignored if this parameter is set to 1. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| OBJ_FUNC | INTEGER | 0 for regression problem.<br>5 for binary classification.<br>7 for multiple classification. | For regression:<br>• 0. Squared error<br>• 1. Squared log error<br>• 2. Pseudo Huber error<br>• 3. Gamma<br>• 4. Tweedie<br>For binary classification:<br>• 5. Logistic<br>• 6. Hinge<br>For multiple classification:<br>• 7. Softmax | |
| TWEEDIE_POWER | Double | 1.5 | For Tweedie object function; must in range [1.0, 2.0] | Only valid when OBJ_FUNC equals 4 |
| REPLACE_MISSING | INTEGER | 1 | Replace missing value by another value in the feature. If it is a continuous feature, the value is the mean value; if it is a categorical feature, the value is the most frequent one.<br>• 0: no<br>• 1: yes | |
| DEFAULT_MISSING_DIRECTION | INTEGER | 1 | Define the default direction where missing value will go to while splitting.<br>• 0: left<br>• 1: right | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| FEATURE_GROUPING | INTEGER | 0 | Apply feature grouping by grouping sparse features that only contains one significant value in each row.<br>• 0: no<br>• 1: yes | |
| TOLERANT_RATE | DOUBLE | 0.0001 | While applying feature grouping, still merge features when there are rows containing more than one significant value. This parameter specifies the rate of such rows allowed. | Only available when FEATURE_GROUPING is set to 1 |
| CATEGORICAL_VARIABLE | VARCHAR or NVARCHAR | Detected from input data | Indicates which variables are treated as categorical. The default behavior is:<br>• STRING: categorical<br>• INTEGER and DOUBLE: continuous | Valid only for INTEGER variables; omitted otherwise. |
| HAS_ID | INTEGER | 0 | Specifies if the first column of the DATA table is used as ID column<br>• 0: no<br>• 1: yes; the first column is discarded during training. | |
| DEPENDENT_VARIABLE | VARCHAR or NVARCHAR | The last column name of DATA table | Specifies the variable used as independent. | Cannot be the first column name if HAS_ID is 1. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| CALCULATE_IMPORTANCE | INTEGER | 1 | Determines whether to calculate variable importance:<br>• 0: no<br>• 1: yes | |
| CALCULATE_CONFUSION_MATRIX | INTEGER | 1 | Determines whether to calculate confusion matrix:<br>• 0: no<br>• 1: yes | For classification only. |
| COMPRESSION | INTEGER | 0 | Indicate whether result model should be compressed:<br>• 0: no<br>• 1: yes | |
| MAX_BITS | INTEGER | 12 | Maximal number of bits to quantize continuous features. Equivalent to use $2^{MAX_BITS}$ bins. | Valid only when COMPRESSION equals to 1. Must be less than 31. |
| VALIDATION_SET_RATE | DOUBLE | 0.0 | Specifies the rate of validation set that be sampled from data set. If 0.0 is set, then no early stop will be applied. | |
| STRATIFIED_VALIDATION_SET | INTEGER | 0 | Indicates whether to apply stratified method while sampling validation set. | Valid only when VALIDATION_SET_RATE is greater than 0. |
| TOLERANT_ITER_NUM | INTEGER | 10 | Indicates how many consecutive deteriorated iterations should be observed before applying early stop. | Valid only when VALIDATION_SET_RATE is greater than 0. |

### Model Evaluation and Parameter Selection

The following parameters are only valid when model evaluation or parameter selection is enabled. For more information of model evaluation and parameter selection refer to its introduction section.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| RESAMPLING_METHOD | NVARCHAR | No default value | Specifies resampling method for model evaluation or parameter selection.<br>• cv<br>• cv_sha<br>• cv_hyperband<br>• stratified_cv<br>• stratified_cv_sha<br>• stratified_cv_hyperband<br>• bootstrap<br>• bootstrap_sha<br>• bootstrap_hyperband<br>• stratified_bootstrap<br>• stratified_bootstrap_sha<br>• stratified_bootstrap_hyperband<br>If no value is specified for this parameter, then no model evaluation or parameter selection will be activated. | Options like stratified_* can only be applied to classification algorithms. Options like *_sha and *_hyperband can only be used in parameter selection, can not be used in model evaluation. |
| EVALUATION_METRIC | NVARCHAR | No default value | Specify evaluation metric for model evaluation or parameter selection.<br>Classification: RMSE, MAE, NLL (Negative Log Likelihood), ERROR_RATE, AUC<br>Regression: RMSE, MAE | It is mandatory if RESAMPLING_METHOD is set. |
| REF_METRIC | VARCHAR/ NVARCHAR | No default value | The metric(s) for reference. Could be of multi-lines, for example, more than one REF_METRIC. | The metric that is identical to EVALUATION_METRIC will be omitted. |
| FOLD_NUM | INTEGER | No default value | Specify fold number for cross validation method. | Mandatory and valid only when RESAMPLING_METHOD is set to cv or stratified_cv. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| REPEAT_TIMES | INTEGER | 1 | Specify repeat times for resampling. | |
| PARAM_SEARCH_STRATEGY | NVARCHAR | No default value | Specify value to this parameter to active parameter selection.<br>• grid<br>• random | When *_sha is specified in RESAMPLING_METHOD, PARAM_SEARCH_STRATEGY is mandatory. When *_hyperband is specified in RESAMPLING_METHOD, PARAM_SEARCH_STRATEGY is set to random whatever other value is provided. Otherwise if this parameter is not set, only model evaluation is activated. |
| RANDOM_SEARCH_TIMES | INTEGER | No default value | Specify times to randomly select candidate parameters for selection. | Mandatory and valid when PARAM_SEARCH_STRATEGY is set to random. |
| REDUCTION_RATE | DOUBLE | 3.0 | Specifies reduction rate in SHA or Hyperband method. | Only valid when RESAMPLING_METHOD is specified as *_sha or *_hyperband. |
| RESOURCE | NVARCHAR | No default value | Specifies resource definition for SHA or Hyperband method.<br>• not specified, size of data set is used as resource by default.<br>• 'ITER_NUM' | Only valid when RESAMPLING_METHOD is specified as *_sha or *_hyperband. |
| MIN_RESOURCE_RATE | DOUBLE | 0.0 or 1/MAX_RESOURCE | Specifies the minimum resource rate that should be used in SHA or Hyperband iteration. | Only valid when RESAMPLING_METHOD is specified as *_sha or *_hyperband. Default value 0.0 for default RESOURCE and default value 1/MAX_RESOURCE for RESOURCE=ITER_NUM |
| MAX_RESOURCE | INTEGER | No default value | Specifies the maximum number of ITER_NUM that should be used in SHA or Hyperband method. | Only valid when RESAMPLING_METHOD is specified as *_sha or *_hyperband. It is valid and mandatory if RESOURCE is set. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| AGGRESSIVE_ELIMINATION | INTEGER | 0 | Specifies whether to apply aggressive elimination while using SHA method. Aggressive elimination happens when the data size and parameters size to be searched does not match and there are still bunch of parameters to be searched while data size reaches its upper limits. If aggressive elimination is applied, lower bound of limit of data size will be used multiple times first to reduce number of parameters.<br>• 0: does not apply aggressive elimination<br>• 1: applies aggressive elimination | Only valid when RESAMPLING_METHOD is specified as *_sha. |
| TIMEOUT | INTEGER | 0 | Specify maximum running time for model evaluation or parameter selection, in seconds. No timeout when 0 is specified. | |
| PROGRESS_INDICATOR_ID | NVARACHAR | No default value | Set an ID of progress indicator for model evaluation or parameter selection. No progress indicator will be active if no value is provided. | |
| ITER_NUM_VALUES | NVARCHAR | No default value | Specifies values of parameter ITER_NUM to be selected | Only valid when PARAM_SEARCH_STRATEGY is specified |
| ITER_NUM_RANGE | NVARCHAR | No default value | Specifies range of parameter ITER_NUM to be selected | Only valid when PARAM_SEARCH_STRATEGY is specified |
| MAX_DEPTH_VALUES | NVARCHAR | No default value | Specifies values of parameter MAX_DEPTH to be selected | Only valid when PARAM_SEARCH_STRATEGY is specified |
| MAX_DEPTH_RANGE | NVARCHAR | No default value | Specifies range of parameter MAX_DEPTH to be selected | Only valid when PARAM_SEARCH_STRATEGY is specified |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| NODE_SIZE_VALUES | NVARCHAR | No default value | Specifies values of parameter NODE_SIZE to be selected | Only valid when PARAM_SEARCH_STRATEGY is specified |
| NODE_SIZE_RANGE | NVARCHAR | No default value | Specifies range of parameter NODE_SIZE to be selected | Only valid when PARAM_SEARCH_STRATEGY is specified |
| ETA_VALUES | NVARCHAR | No default value | Specifies values of parameter ETA to be selected | Only valid when PARAM_SEARCH_STRATEGY is specified |
| ETA_RANGE | NVARCHAR | No default value | Specifies range of parameter ETA to be selected | Only valid when PARAM_SEARCH_STRATEGY is specified |
| GAMMA_VALUES | NVARCHAR | No default value | Specifies values of parameter GAMMA to be selected | Only valid when PARAM_SEARCH_STRATEGY is specified |
| GAMMA_RANGE | NVARCHAR | No default value | Specifies range of parameter GAMMA to be selected | Only valid when PARAM_SEARCH_STRATEGY is specified |
| MIN_CHILD_HESSIAN_VALUES | NVARCHAR | No default value | Specifies values of parameter MIN_CHILD_HESSIAN to be selected | Only valid when PARAM_SEARCH_STRATEGY is specified |
| MIN_CHILD_HESSIAN_RANGE | NVARCHAR | No default value | Specifies range of parameter MIN_CHILD_HESSIAN to be selected | Only valid when PARAM_SEARCH_STRATEGY is specified |
| NODE_WEIGHT_CONSTRAINT _VALUES | NVARCHAR | No default value | Specifies values of parameter NODE_WEIGHT_CONSTRAINT to be selected | Only valid when PARAM_SEARCH_STRATEGY is specified |
| NODE_WEIGHT_CONSTRAINT _RANGE | NVARCHAR | No default value | Specifies range of parameter NODE_WEIGHT_CONSTRAINT to be selected | Only valid when PARAM_SEARCH_STRATEGY is specified |
| ROW_SAMPLE_RATE _VALUES | NVARCHAR | No default value | Specifies values of parameter ROW_SAMPLE_RATE to be selected | Only valid when PARAM_SEARCH_STRATEGY is specified |
| ROW_SAMPLE_RATE _RANGE | NVARCHAR | No default value | Specifies range of parameter ROW_SAMPLE_RATE to be selected | Only valid when PARAM_SEARCH_STRATEGY is specified |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| COL_SAMPLE_RATE_BYSPLIT_VALUES | NVARCHAR | No default value | Specifies values of parameter COL_SAMPLE_RATE_BYSPLIT to be selected | Only valid when PARAM_SEARCH_STRATEGY is specified |
| COL_SAMPLE_RATE_BYSPLIT_RANGE | NVARCHAR | No default value | Specifies range of parameter COL_SAMPLE_RATE_BYSPLIT to be selected | Only valid when PARAM_SEARCH_STRATEGY is specified |
| COL_SAMPLE_RATE_BYTREE_VALUES | NVARCHAR | No default value | Specifies values of parameter COL_SAMPLE_RATE_BYTREE to be selected | Only valid when PARAM_SEARCH_STRATEGY is specified |
| COL_SAMPLE_RATE_BYTREE_RANGE | NVARCHAR | No default value | Specifies range of parameter COL_SAMPLE_RATE_BYTREE to be selected | Only valid when PARAM_SEARCH_STRATEGY is specified |
| LAMBDA _VALUES | NVARCHAR | No default value | Specifies values of parameter LAMBDA to be selected | Only valid when PARAM_SEARCH_STRATEGY is specified |
| LAMBDA _RANGE | NVARCHAR | No default value | Specifies range of parameter LAMBDA to be selected | Only valid when PARAM_SEARCH_STRATEGY is specified |
| ALPHA _VALUES | NVARCHAR | No default value | Specifies values of parameter ALPHA to be selected | Only valid when PARAM_SEARCH_STRATEGY is specified |
| ALPHA _RANGE | NVARCHAR | No default value | Specifies range of parameter ALPHA to be selected | Only valid when PARAM_SEARCH_STRATEGY is specified |
| BASE_SCORE _VALUES | NVARCHAR | No default value | Specifies values of parameter BASE_SCORE to be selected | Only valid when PARAM_SEARCH_STRATEGY is specified |
| BASE_SCORE _RANGE | NVARCHAR | No default value | Specifies range of parameter BASE_SCORE to be selected | Only valid when PARAM_SEARCH_STRATEGY is specified |

ⓘ Note

- To active parameter selection or model evaluation, parameter RESAMPLING_METHOD must be set. Then, to active parameter selection, parameter PARAM_SEARCH_STRATEGY must be set, or only model evaluation is carried out. For parameters like *_VALUES, its format is {val1, val2, ..., valn}. For parameters like *_RANGE, its format is [RANGE_START, STEP, RANGE_END]. STEP can be omitted if using random search, in which case its format is [RANGE_START ,, RANGE_END]. Commas are still needed.
- To make this function back compatible, old cross validation fashion is still supported. In this case, RESAMPLING_METHOD is not set and FOLD_NUM is set to a value greater than 1. Parameter EVALUATION_METRIC and REF_METRIC has default value ERROR_RATE for classification and RMSE for regression. And parameters for searching purpose are of name RANGE_* and format like [RANGE_START, RANGE_END, NUM], of which the first and second ones are of integer or double types,

> whereas the last one is of integer type (not less than 2), indicating that the number of candidate values by equal space constrained by range of [RANGE_START, RANGE_END].

### Output Tables

| Table | Column | Data Type | Column Name | Description | Constraint |
|---|---|---|---|---|---|
| MODEL | 1st column | INTEGER | ROW_INDEX | Indicates the row | |
| | 2nd column | INTEGER | TREE_INDEX | Indicates the tree index | -1 indicates the global information. |
| | 3rd column | NCLOB | MODEL_CONTENT | Model value | |
| IMP | 1st column | NVARCHAR(256) | VARIABLE_NAME | Independent variable name | |
| | 2nd column | DOUBLE | IMPORTANCE | Variable importance | |
| CONFUSION (only for classification | 1st column | NVARCHAR(1000) | ACTUAL_CLASS | The actual class name | |
| | 2nd column | NVARCHAR(1000) | PREDICTED_CLASS | The predicted class name | |
| | 3rd column | INTEGER | COUNT | Number of records | |
| STATS | 1st column | NVARCHAR(1000) | STAT_NAME | Statistics name | |
| | 2nd column | NVARCHAR(1000) | STAT_VALUE | Statistics value | |
| PARAM_SELECTION | 1st column | NVARCHAR(256) | PARM_NAME | Parameter name | |
| | 2nd column | INTEGER | INT_VALUE | Integer value | |
| | 3rd column | DOUBLE | DOUBLE_VALUE | Double value | |
| | 4th column | NVARCHAR(1000) | STRING_VALUE | String value | |

### Examples

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

### Example 1: Classification

```
set schema DM_PAL;
drop table PAL_HGBT_DATA_TAB;
create column table PAL_HGBT_DATA_TAB (
```

```
    "ATT1" double,
    "ATT2" double,
    "ATT3" double,
    "ATT4" double,
    "LABEL" nvarchar(50)
);
insert into PAL_HGBT_DATA_TAB values (1.0, 10.0, 100, 1.0, 'A');
insert into PAL_HGBT_DATA_TAB values (1.1, 10.1, 100, 1.0, 'A');
insert into PAL_HGBT_DATA_TAB values (1.2, 10.2, 100, 1.0, 'A');
insert into PAL_HGBT_DATA_TAB values (1.3, 10.4, 100, 1.0, 'A');
insert into PAL_HGBT_DATA_TAB values (1.2, 10.3, 100, 1.0, 'A');
insert into PAL_HGBT_DATA_TAB values (4.0, 40.0, 400, 4.0, 'B');
insert into PAL_HGBT_DATA_TAB values (4.1, 40.1, 400, 4.0, 'B');
insert into PAL_HGBT_DATA_TAB values (4.2, 40.2, 400, 4.0, 'B');
insert into PAL_HGBT_DATA_TAB values (4.3, 40.4, 400, 4.0, 'B');
insert into PAL_HGBT_DATA_TAB values (4.2, 40.3, 400, 4.0, 'A');
insert into PAL_HGBT_DATA_TAB values (9.0, 90.0, 900, 2.0, 'A');
insert into PAL_HGBT_DATA_TAB values (9.1, 90.1, 900, 1.0, 'B');
insert into PAL_HGBT_DATA_TAB values (9.2, 90.2, 900, 2.0, 'B');
insert into PAL_HGBT_DATA_TAB values (9.3, 90.4, 900, 1.0, 'B');
insert into PAL_HGBT_DATA_TAB values (9.2, 90.3, 900, 1.0, 'B');
drop table PAL_PARAMETER_TAB;
create column table PAL_PARAMETER_TAB (
    "PARAM_NAME" nvarchar(100),
    "INT_VALUE" integer,
    "DOUBLE_VALUE" double,
    "STRING_VALUE" nvarchar(100)
);
insert into PAL_PARAMETER_TAB values ('RESAMPLING_METHOD', null, null, 'cv');
insert into PAL_PARAMETER_TAB values ('FOLD_NUM', 5, null, null);
insert into PAL_PARAMETER_TAB values ('PARAM_SEARCH_STRATEGY', null, null,
'grid');
insert into PAL_PARAMETER_TAB values ('EVALUATION_METRIC', null, null,
'ERROR_RATE');
insert into PAL_PARAMETER_TAB values ('REF_METRIC', null, null, 'AUC');
insert into PAL_PARAMETER_TAB values ('SEED', 1, null, null);
insert into PAL_PARAMETER_TAB values ('MAX_DEPTH_VALUES', null, null, '{3, 5,
6}');
insert into PAL_PARAMETER_TAB values ('ITER_NUM_RANGE', null, null, '[4, 2,
10]');
insert into PAL_PARAMETER_TAB values ('ETA_RANGE', null, null, '[0.1, 0.2,
1.0]');
insert into PAL_PARAMETER_TAB values ('GAMMA_RANGE', null, null, '[0.1, 0.2,
1.0]');
drop table PAL_HGBT_MODEL_TAB; -- for predict followed
create column table PAL_HGBT_MODEL_TAB (
    "ROW_INDEX" integer,
    "TREE_INDEX" integer,
    "MODEL_CONTENT" nclob
);
do begin
    LT_DATA_TAB = select * from PAL_HGBT_DATA_TAB;
    LT_PARAM_TAB = select * from PAL_PARAMETER_TAB;
    call _SYS_AFL.PAL_HGBT(:LT_DATA_TAB, :LT_PARAM_TAB, LT_MODEL_TAB,
LT_IMP_TAB, LT_CONFUSE_TAB, LT_STAT_TAB, LT_CV_TAB);
    select * from :LT_MODEL_TAB;
    select * from :LT_IMP_TAB;
    select * from :LT_CONFUSE_TAB;
    select * from :LT_STAT_TAB;
    select * from :LT_CV_TAB;
    insert into PAL_HGBT_MODEL_TAB select * from :LT_MODEL_TAB;
end;
```

### Expected Result

MODEL:

| | ROW_INDEX | TREE_INDEX | MODEL_CONTENT |
|---|---|---|---|
| 1 | 1 | -1 | {"nclass":2,"param":{"bs":0.5,"obj":"binary:logistic"},"var":{"m... |
| 2 | 2 | 0 | {"height":1,"nnode":3,"nodes":[{"ch":[1,2],"gn":0.0526315789... |
| 3 | 3 | 1 | {"height":1,"nnode":3,"nodes":[{"ch":[1,2],"gn":0.0166668904... |
| 4 | 4 | 2 | {"height":1,"nnode":3,"nodes":[{"ch":[1,2],"gn":1.8553484488... |
| 5 | 5 | 3 | {"height":0,"nnode":1,"nodes":[{"ch":[],"gn":0.002239983194... |

IMP:

| | VARIABLE_NAME | IMPORTANCE |
|---|---|---|
| 1 | ATT1 | 0.9230188140468646 |
| 2 | ATT2 | 0 |
| 3 | ATT3 | 0 |
| 4 | ATT4 | 0.0769811859531354 |

CONFUSION:

| | ACTUAL_CLASS | PREDICTED_CLASS | COUNT |
|---|---|---|---|
| 1 | A | A | 5 |
| 2 | A | B | 2 |
| 3 | B | A | 0 |
| 4 | B | B | 8 |

STATS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | ERROR_RATE_MEAN | 0.133333 |
| 2 | ERROR_RATE_VAR | 0.0266666 |
| 3 | AUC_MEAN | 0.9 |

PARAM_SELECTION:

| | PARAM_NAME | INT_VALUE | DOUBLE_VALUE | STRING_VALUE |
|---|---|---|---|---|
| 1 | ITER_NUM | 4 | ? | ? |
| 2 | ETA | ? | 1 | ? |
| 3 | GAMMA | ? | 0.1 | ? |

**Example 2: Regression**

```
set schema DM_PAL;
drop table PAL_HGBT_DATA_TAB;
create column table PAL_HGBT_DATA_TAB (
    "ATT1" double,
```

```
    "ATT2" double,
    "ATT3" double,
    "ATT4" double,
    "TARGET" double
);
insert into PAL_HGBT_DATA_TAB values (19.76, 6235, 100.0, 100.0, 25.10);
insert into PAL_HGBT_DATA_TAB values (17.85, 46230, 43.67, 84.53, 19.23);
insert into PAL_HGBT_DATA_TAB values (19.96, 7360, 65.51, 81.57, 21.42);
insert into PAL_HGBT_DATA_TAB values (16.80, 28715, 45.16, 93.33, 18.11);
insert into PAL_HGBT_DATA_TAB values (18.20, 21934, 49.20, 83.07, 19.24);
insert into PAL_HGBT_DATA_TAB values (16.71, 1337, 74.84, 94.99, 19.31);
insert into PAL_HGBT_DATA_TAB values (18.81, 17881, 70.66, 92.34, 20.07);
insert into PAL_HGBT_DATA_TAB values (20.74, 2319, 63.93, 95.08, 24.35);
insert into PAL_HGBT_DATA_TAB values (16.56, 18040, 14.45, 61.24, 17.60);
insert into PAL_HGBT_DATA_TAB values (18.55, 1147, 68.58, 97.90, 20.13);
insert into PAL_HGBT_DATA_TAB values (17.40, 2176, 53.33, 97.50, 18.40);
insert into PAL_HGBT_DATA_TAB values (17.62, 13267, 25.16, 56.86, 18.96);
insert into PAL_HGBT_DATA_TAB values (21.24, 3581, 35.76, 63.58, 25.75);
insert into PAL_HGBT_DATA_TAB values (18.23, 15104, 47.72, 95.29, 19.40);
insert into PAL_HGBT_DATA_TAB values (16.86, 47009, 17.21, 100.0, 18.64);
insert into PAL_HGBT_DATA_TAB values (17.45, 10139, 43.15, 89.40, 19.10);
insert into PAL_HGBT_DATA_TAB values (17.66, 6147, 67.73, 92.54, 20.00);
insert into PAL_HGBT_DATA_TAB values (18.30, 23089, 33.27, 67.53, 19.31);
insert into PAL_HGBT_DATA_TAB values (16.58, 20550, 26.61, 98.32, 20.49);
insert into PAL_HGBT_DATA_TAB values (17.51, 9450, 61.35, 86.72, 17.07);
insert into PAL_HGBT_DATA_TAB values (21.17, 1028, 100.0, 100.0, 20.61);
insert into PAL_HGBT_DATA_TAB values (16.92, 3848, 5.350, 65.58, 15.73);
insert into PAL_HGBT_DATA_TAB values (16.96, 15656, 20.53, 93.72, 18.70);
insert into PAL_HGBT_DATA_TAB values (18.24, 7725, 50.59, 96.63, 18.99);
drop table PAL_PARAMETER_TAB;
create column table PAL_PARAMETER_TAB (
    "PARAM_NAME" nvarchar (100),
    "INT_VALUE" integer,
    "DOUBLE_VALUE" double,
    "STRING_VALUE" nvarchar(100)
);
insert into PAL_PARAMETER_TAB values ('SPLIT_METHOD',  null, null, 'exact');
insert into PAL_PARAMETER_TAB values ('MAX_DEPTH', 6, null, null);
insert into PAL_PARAMETER_TAB values ('RESAMPLING_METHOD', null, null, 'cv');
insert into PAL_PARAMETER_TAB values ('FOLD_NUM', 5, null, null);
insert into PAL_PARAMETER_TAB values ('PARAM_SEARCH_STRATEGY', null, null,
'grid');
insert into PAL_PARAMETER_TAB values ('EVALUATION_METRIC', null, null, 'RMSE');
insert into PAL_PARAMETER_TAB values ('REF_METRIC', null, null, 'MAE');
insert into PAL_PARAMETER_TAB values ('SEED', 1, null, null);
insert into PAL_PARAMETER_TAB values ('ETA_RANGE', null, null, '[0.1, 0.2,
1.0]');
insert into PAL_PARAMETER_TAB values ('GAMMA_RANGE', null, null, '[0.0, 0.2,
1.0]');
insert into PAL_PARAMETER_TAB values ('ITER_NUM_RANGE', null, null, '[10, 10,
20]');
drop table PAL_HGBT_MODEL_TAB; -- for predict followed
create column table PAL_HGBT_MODEL_TAB (
    "ROW_INDEX" integer,
    "TREE_INDEX" integer,
    "MODEL_CONTENT" nclob
);
do begin
    LT_DATA_TAB = select * from PAL_HGBT_DATA_TAB;
    LT_PARAM_TAB = select * from PAL_PARAMETER_TAB;
    call _SYS_AFL.PAL_HGBT(:LT_DATA_TAB, :LT_PARAM_TAB, LT_MODEL_TAB,
LT_IMP_TAB, LT_CONFUSE_TAB, LT_STAT_TAB, LT_CV_TAB);
    select * from :LT_MODEL_TAB;
    select * from :LT_IMP_TAB;
    select * from :LT_CONFUSE_TAB;
    select * from :LT_STAT_TAB;
    select * from :LT_CV_TAB;
    insert into PAL_HGBT_MODEL_TAB select * from :LT_MODEL_TAB;
```

```
end;
```

**Expected Result**

MODEL:

| | ROW_INDEX | TREE_INDEX | MODEL_CONTENT |
|---|---|---|---|
| 1 | 1 | -1 | {"nclass":1,"param":{"bs":0.0,"obj":"reg:linear"},"var":{"map":n... |
| 2 | 2 | 0 | {"height":0,"nnode":1,"nodes":[{"ch":[],"gn":9052.000163999... |
| 3 | 3 | 1 | {"height":0,"nnode":1,"nodes":[{"ch":[],"gn":5218.533478303... |
| 4 | 4 | 2 | {"height":0,"nnode":1,"nodes":[{"ch":[],"gn":3008.516479317... |
| 5 | 5 | 3 | {"height":1,"nnode":3,"nodes":[{"ch":[1,2],"gn":1734.4281576... |
| 6 | 6 | 4 | {"height":1,"nnode":3,"nodes":[{"ch":[1,2],"gn":1007.6894132... |
| 7 | 7 | 5 | {"height":1,"nnode":3,"nodes":[{"ch":[1,2],"gn":597.86140479... |
| 8 | 8 | 6 | {"height":2,"nnode":5,"nodes":[{"ch":[1,2],"gn":355.89102463... |
| 9 | 9 | 7 | {"height":2,"nnode":5,"nodes":[{"ch":[1,2],"gn":213.98481794... |
| 10 | 10 | 8 | {"height":3,"nnode":9,"nodes":[{"ch":[1,2],"gn":129.26568432... |
| 11 | 11 | 9 | {"height":1,"nnode":3,"nodes":[{"ch":[1,2],"gn":80.140914758... |
| 12 | 12 | 10 | {"height":3,"nnode":9,"nodes":[{"ch":[1,2],"gn":47.868628264... |
| 13 | 13 | 11 | {"height":2,"nnode":7,"nodes":[{"ch":[1,2],"gn":30.067072332... |
| 14 | 14 | 12 | {"height":4,"nnode":13,"nodes":[{"ch":[1,2],"gn":18.16260500... |
| 15 | 15 | 13 | {"height":3,"nnode":11,"nodes":[{"ch":[1,2],"gn":11.65244747... |
| 16 | 16 | 14 | {"height":1,"nnode":3,"nodes":[{"ch":[1,2],"gn":7.4584527810... |
| 17 | 17 | 15 | {"height":0,"nnode":1,"nodes":[{"ch":[],"gn":4.471811826378... |
| 18 | 18 | 16 | {"height":1,"nnode":3,"nodes":[{"ch":[1,2],"gn":2.5780268782... |
| 19 | 19 | 17 | {"height":0,"nnode":1,"nodes":[{"ch":[],"gn":1.607072959788... |

IMP:

| | VARIABLE_NAME | IMPORTANCE |
|---|---|---|
| 1 | ATT1 | 0.7062101272203666 |
| 2 | ATT2 | 0.13202836324579875 |
| 3 | ATT3 | 0.13750332479092758 |
| 4 | ATT4 | 0.024258184742907... |

CONFUSION:

Empty

STATS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | RMSE_MEAN | 1.83732 |
| 2 | RMSE_VAR | 0.525659 |
| 3 | MAE_MEAN | 1.44394 |

PARAM_SELECTION:

| | PARAM_NAME | INT_VALUE | DOUBLE_VALUE | STRING_VALUE |
|---|---|---|---|---|
| 1 | ITER_NUM | 18 | ? | ? |
| 2 | ETA | ? | 0.250749 | ? |
| 3 | GAMMA | ? | 1 | ? |

## _AFL_SYS.PAL_HGBT_CONTINUE

This function continues training gradient boosting tree model based on a previously trained HGBT model.

### Overview of all tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PRE-MODEL table>` | IN |
| 3 | `<PARAMETER table>` | IN |
| 4 | `<MODEL table>` | OUT |
| 5 | `<VARIABLE_IMPORTANCE table>` | OUT |
| 6 | `<CONFUSION_MATRIX table>` | OUT |
| 7 | `<STATISTICS table>` | OUT |
| 8 | `<CROSS_VALIDATION table>` | OUT |

### Signature

### Input Table

| Table | Column | Column Data Type | Description | Constraint |
|---|---|---|---|---|
| DATA | Columns | VARCHAR, NVARCHAR, NTEGER,DOUBLE, or DECIMAL (p, s) | Independent variables.<br>Discrete value: INTEGER, VARCHAR, and NVARCHAR<br>Continuous value: INTEGER, DOUBLE, and DECIMAL (p, s), | If the HAS_ID parameter is 1, the first column is an ID column and can be of BIGINT type but discarded in training. |
| | Last column | VARCHAR, NVARCHAR, INTEGER, DOUBLE, or DECIMAL (p, s) | Dependent variable.<br>The VARCHAR type is used for classification, and DOUBLE type is for regression.<br>The INTEGER type is by default used for regression, but if intentionally treated as categorical variable, that is, using parameter CATEGORICAL_VARIABLE to indicate, then it is solved as classification. | If DEPENDENT_VARIABLE is set to other columns, the last column is used as independent variable. |
| PRE-MODEL | 1st column | Integer | Row index | |
| | 2nd column | Integer | Tree index | |
| | 3rd column | NCLOB | Model content | |

### Parameter Table

Parameters of this function are exactly like the ones of _AFL_SYS.PAL_HGBT except some ones which can be decided from model of previous training.

Following listed parameters will be IGNORED.

| Name | Data Type |
|---|---|
| BASE_SCORE | DOUBLE |
| START_FROM_AVERAGE | INTEGER |
| OBJ_FUNC | INTEGER |
| CATEGORICAL_VARIABLE | VARCHAR, or NVARCHAR |
| DEPENDENT_VARIABLE | VARCHAR, or NVARCHAR |

### Output Table

| Table | Column | Column Data Type | Column Name | Description | Constraint |
|---|---|---|---|---|---|
| MODEL | 1st column | INTEGER | ROW_INDEX | Indicates the row | |
| | 2nd column | INTEGER | TREE_INDEX | Indicates the tree index | -1 indicates the global information. |
| | 3rd column | NCLOB | MODEL_CONTENT | Model value | |
| IMP | 1st column | NVARCHAR (256) | VARIABLE_NAME | Independent variable name | |
| | 2nd column | DOUBLE | IMPORTANCE | Variable importance | |
| CONFUSION (only for classification | 1st column | NVARCHAR (1000) | ACTUAL_CLASS | The actual class name | |
| | 2nd column | NVARCHAR (1000) | PREDICTED_CLASS | The predicted class name | |
| | 3rd column | INTEGER | COUNT | Number of records | |
| STATS | 1st column | NVARCHAR (1000) | STAT_NAME | Statistics name | |
| | 2nd column | NVARCHAR (1000) | STAT_VALUE | Statistics value | |
| PARAM_SELECTION | 1st column | NVARCHAR (256) | PARM_NAME | Parameter name | |
| | 2nd column | INTEGER | INT_VALUE | Integer value | |
| | 3rd column | DOUBLE | DOUBLE_VALUE | Double value | |
| | 4th column | NVARCHAR (1000) | STRING_VALUE | String value | |

### Examples

Assume that:

- • DM_PAL is a schema belonging to USER1; and
- • USER1 has been assigned the AFL__SYS_AFL_AFLPAL_EXECUTE or AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION role.

```
set schema DM_PAL;
drop table PAL_HGBT_DATA_TAB;
create column table PAL_HGBT_DATA_TAB (
    "ATT1" double,
    "ATT2" double,
    "ATT3" double,
    "ATT4" double,
```

```
    "LABEL" nvarchar(50)
);
insert into PAL_HGBT_DATA_TAB values (1.0, 10.0, 100, 1.0, 'A');
insert into PAL_HGBT_DATA_TAB values (1.1, 10.1, 100, 1.0, 'A');
insert into PAL_HGBT_DATA_TAB values (1.2, 10.2, 100, 1.0, 'A');
insert into PAL_HGBT_DATA_TAB values (1.3, 10.4, 100, 1.0, 'A');
insert into PAL_HGBT_DATA_TAB values (1.2, 10.3, 100, 1.0, 'A');
insert into PAL_HGBT_DATA_TAB values (4.0, 40.0, 400, 4.0, 'B');
insert into PAL_HGBT_DATA_TAB values (4.1, 40.1, 400, 4.0, 'B');
insert into PAL_HGBT_DATA_TAB values (4.2, 40.2, 400, 4.0, 'B');
insert into PAL_HGBT_DATA_TAB values (4.3, 40.4, 400, 4.0, 'B');
insert into PAL_HGBT_DATA_TAB values (4.2, 40.3, 400, 4.0, 'A');
drop table PAL_PARAMETER_TAB;
create column table PAL_PARAMETER_TAB (
    "PARAM_NAME" nvarchar(100),
    "INT_VALUE" integer,
    "DOUBLE_VALUE" double,
    "STRING_VALUE" nvarchar(100)
);
insert into PAL_PARAMETER_TAB values ('ITER_NUM', 10, null, null);
drop table PAL_HGBT_MODEL_TAB1;
create column table PAL_HGBT_MODEL_TAB1 (
    "ROW_INDEX" integer,
    "TREE_INDEX" integer,
    "MODEL_CONTENT" nclob
);
-- first training
do begin
    LT_DATA_TAB = select * from PAL_HGBT_DATA_TAB;
    LT_PARAM_TAB = select * from PAL_PARAMETER_TAB;
    call _SYS_AFL.PAL_HGBT(:LT_DATA_TAB, :LT_PARAM_TAB, LT_MODEL_TAB,
LT_IMP_TAB, LT_CONFUSE_TAB, LT_STAT_TAB, LT_CV_TAB);
    select * from :LT_MODEL_TAB;
    select * from :LT_IMP_TAB;
    select * from :LT_CONFUSE_TAB;
    select * from :LT_STAT_TAB;
    select * from :LT_CV_TAB;
    insert into PAL_HGBT_MODEL_TAB1 select * from :LT_MODEL_TAB;
end;
-- add extra data
insert into PAL_HGBT_DATA_TAB values (9.0, 90.0, 900, 2.0, 'A');
insert into PAL_HGBT_DATA_TAB values (9.1, 90.1, 900, 1.0, 'B');
insert into PAL_HGBT_DATA_TAB values (9.2, 90.2, 900, 2.0, 'B');
insert into PAL_HGBT_DATA_TAB values (9.3, 90.4, 900, 1.0, 'B');
insert into PAL_HGBT_DATA_TAB values (9.2, 90.3, 900, 1.0, 'B');
truncate table PAL_PARAMETER_TAB;
insert into PAL_PARAMETER_TAB values ('ITER_NUM', 10, null, null);
drop table PAL_HGBT_MODEL_TAB2;
create column table PAL_HGBT_MODEL_TAB2 (
    "ROW_INDEX" integer,
    "TREE_INDEX" integer,
    "MODEL_CONTENT" nclob
);
-- second training
do begin
    LT_DATA_TAB = select * from PAL_HGBT_DATA_TAB;
    LT_PARAM_TAB = select * from PAL_PARAMETER_TAB;
    LT_PRE_MODEL_TAB = select * from PAL_HGBT_MODEL_TAB1;
    call
_SYS_AFL.PAL_HGBT_CONTINUE(:LT_DATA_TAB, :LT_PRE_MODEL_TAB, :LT_PARAM_TAB,
LT_MODEL_TAB, LT_IMP_TAB, LT_CONFUSE_TAB, LT_STAT_TAB, LT_CV_TAB);
    select * from :LT_MODEL_TAB;
    select * from :LT_IMP_TAB;
    select * from :LT_CONFUSE_TAB;
    select * from :LT_STAT_TAB;
    select * from :LT_CV_TAB;
    insert into PAL_HGBT_MODEL_TAB2 select * from :LT_MODEL_TAB;
end;
```

**Expected Result**

MODEL:

| | ROW_INDEX | TREE_INDEX | MODEL_CONTENT |
|---|---|---|---|
| 1 | 1 | -1 | {"model_version":"2.0","nclass":2,"nrow":15,"param":{"bs":[0.... |
| 2 | 2 | 0 | {"height":1,"nnode":3,"nodes":[{"ch":[1,2],"gn":-0.0,"id":0,"m... |
| 3 | 3 | 1 | {"height":1,"nnode":3,"nodes":[{"ch":[1,2],"gn":-0.0,"id":0,"m... |
| 4 | 4 | 2 | {"height":1,"nnode":3,"nodes":[{"ch":[1,2],"gn":-0.0,"id":0,"m... |
| 5 | 5 | 3 | {"height":1,"nnode":3,"nodes":[{"ch":[1,2],"gn":-0.0,"id":0,"m... |
| 6 | 6 | 4 | {"height":1,"nnode":3,"nodes":[{"ch":[1,2],"gn":-0.0,"id":0,"m... |
| 7 | 7 | 5 | {"height":0,"nnode":1,"nodes":[{"ch":[],"gn":-0.0,"id":0,"n":10,... |
| 8 | 8 | 6 | {"height":0,"nnode":1,"nodes":[{"ch":[],"gn":-0.0,"id":0,"n":10,... |
| 9 | 9 | 7 | {"height":0,"nnode":1,"nodes":[{"ch":[],"gn":-0.0,"id":0,"n":10,... |
| 10 | 10 | 8 | {"height":0,"nnode":1,"nodes":[{"ch":[],"gn":-0.0,"id":0,"n":10,... |
| 11 | 11 | 9 | {"height":0,"nnode":1,"nodes":[{"ch":[],"gn":-0.0,"id":0,"n":10,... |
| 12 | 12 | 10 | {"height":1,"nnode":3,"nodes":[{"ch":[1,2],"gn":0.0421354464... |
| 13 | 13 | 11 | {"height":1,"nnode":3,"nodes":[{"ch":[1,2],"gn":0.0319613643... |
| 14 | 14 | 12 | {"height":1,"nnode":3,"nodes":[{"ch":[1,2],"gn":0.0171246147... |
| 15 | 15 | 13 | {"height":1,"nnode":3,"nodes":[{"ch":[1,2],"gn":0.0135604999... |
| 16 | 16 | 14 | {"height":1,"nnode":3,"nodes":[{"ch":[1,2],"gn":0.0114785732... |
| 17 | 17 | 15 | {"height":1,"nnode":3,"nodes":[{"ch":[1,2],"gn":0.0092278107... |
| 18 | 18 | 16 | {"height":1,"nnode":3,"nodes":[{"ch":[1,2],"gn":0.0079698215... |

IMP:

| | VARIABLE_NAME | IMPORTANCE |
|---|---|---|
| 1 | ATT1 | 0.6779682741272295 |
| 2 | ATT2 | 0 |
| 3 | ATT3 | 0 |
| 4 | ATT4 | 0.32203172587277... |

CONFUSION:

| | ACTUAL_CLASS | PREDICTED_CLASS | COUNT |
|---|---|---|---|
| 1 | A | A | 5 |
| 2 | A | B | 2 |
| 3 | B | A | 0 |
| 4 | B | B | 8 |

STATS:

Empty

PARAM_SELECTION:

Empty

## _SYS_AFL.PAL_HGBT_PREDICT

This function is used for classification or regression scoring, based on trained gradient boosting tree model.

**Overview of all tables**

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<MODEL table>` | IN |
| 3 | `<PARAMETER table>` | IN |
| 4 | `<RESULT table>` | OUT |

**Signature**

**Input Tables**

| Table | Column | Reference for Output Table | Data Type | Description | Constraint |
|---|---|---|---|---|---|
| DATA | 1st column | ID | INTEGER, BIGINT, VARCHAR, or NVARCHAR | Data ID | This must be the first column. |

| Table | Column | Reference for Output Table | Data Type | Description | Constraint |
|---|---|---|---|---|---|
| | Other columns | | INTEGER, DOUBLE, DECIMAL(p, s), VARCHAR, or NVARCHAR | Independent variables | |
| MODEL | 1st column | | INTEGER | Row index | This must be the first column. |
| | 2nd column | | INTEGER | Tree index | |
| | 3rd column | | NCLOB | Model content | |

**Parameter Table**

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If they are not specified, PAL will use their default values.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| THREAD_RATIO | DOUBLE | -1 | The ratio of available threads.<br>• 0: single thread<br>• 0~1: percentage<br>• Others: heuristically determined | |
| VERBOSE | INTEGER | 0 | Specifies whether to output all classes and the corresponding confidences for each data.<br>• 0: no<br>• 1: yes | Valid only for classification |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| MISSING_REPLACEMENT | INTEGER | 1 | Specifies the missing replacement strategy:<br>• 1: marginalises each missing feature out independently<br>• 2: marginalises all missing features in an instance as a whole corresponding to each category | Valid only for classification. Strategy 2 is a much coarser approximation. Only apply strategy 2 when there are a great number of missing features and a lot of classed, which might cause memory insufficiency. |

### Output Table

| Table | Column | Data Type | Column Name | Description | Constraint |
|---|---|---|---|---|---|
| RESULT | 1st column | ID column type | ID column name | ID | |
| | 2nd column | NVARCHAR(100) | SCORE | Scoring result | |
| | 3rd column | DOUBLE | CONFIDENCE | The confidence of a class. | All NULLs for regression. |

### Examples

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

#### Example 1: Classification prediction

```
set schema DM_PAL;
drop table PAL_HGBT_PREDICT_TAB;
create column table PAL_HGBT_PREDICT_TAB (
    "ID" integer,
    "ATT1" double,
    "ATT2" double,
    "ATT3" double,
    "ATT4" double
);
insert into PAL_HGBT_PREDICT_TAB values (1, 1.0, 10.0, 100, 1);
insert into PAL_HGBT_PREDICT_TAB values (2, 1.1, 10.1, 100, 1);
insert into PAL_HGBT_PREDICT_TAB values (3, 1.2, 10.2, 100, 1);
insert into PAL_HGBT_PREDICT_TAB values (4, 1.3, 10.4, 100, 1);
insert into PAL_HGBT_PREDICT_TAB values (5, 1.2, 10.3, 100, 3);
insert into PAL_HGBT_PREDICT_TAB values (6, 4.0, 40.0, 400, 3);
insert into PAL_HGBT_PREDICT_TAB values (7, 4.1, 40.1, 400, 3);
insert into PAL_HGBT_PREDICT_TAB values (8, 4.2, 40.2, 400, 3);
insert into PAL_HGBT_PREDICT_TAB values (9, 4.3, 40.4, 400, 3);
insert into PAL_HGBT_PREDICT_TAB values (10,4.2, 40.3, 400, 3);
```

```
drop table #PAL_PARAMETER_TAB;
create local temporary column table #PAL_PARAMETER_TAB (
    "PARAM_NAME" varchar(100),
    "INT_VALUE" integer,
    "double_VALUE" double,
    "STRING_VALUE" varchar(100)
);
insert into #PAL_PARAMETER_TAB values ('THREAD_RATIO', null, 0.5, null);
insert into #PAL_PARAMETER_TAB values ('VERBOSE', 0, null, null);
call _SYS_AFL.PAL_HGBT_PREDICT(PAL_HGBT_PREDICT_TAB, PAL_HGBT_MODEL_TAB,
#PAL_PARAMETER_TAB, ?);
```

**Expected Result**

RESULT:

| | ID | SCORE | CONFIDENCE |
|---|---|---|---|
| 1 | 1 | A | 0.8526735907154195 |
| 2 | 2 | A | 0.8526735907154195 |
| 3 | 3 | A | 0.8526735907154195 |
| 4 | 4 | A | 0.8526735907154195 |
| 5 | 5 | A | 0.7513939223190946 |
| 6 | 6 | B | 0.7031187459849326 |
| 7 | 7 | B | 0.7031187459849326 |
| 8 | 8 | B | 0.7031187459849326 |
| 9 | 9 | B | 0.8305487410573289 |
| 10 | 10 | B | 0.7031187459849326 |

**Example 2: Regression prediction**

```
set schema DM_PAL;
drop table PAL_HGBT_PREDICT_TAB;
create column table PAL_HGBT_PREDICT_TAB (
    "ID" integer,
    "ATT1" double,
    "ATT2" double,
    "ATT3" double,
    "ATT4" double
);
insert into PAL_HGBT_PREDICT_TAB values (1, 19.76, 6235, 100.0, 100.0);
insert into PAL_HGBT_PREDICT_TAB values (2, 17.85, 46230, 43.67, 84.53);
insert into PAL_HGBT_PREDICT_TAB values (3, 19.96, 7360, 65.51, 81.57);
insert into PAL_HGBT_PREDICT_TAB values (4, 16.80, 28715, 45.16, 93.33);
insert into PAL_HGBT_PREDICT_TAB values (5, 18.20, 21934, 49.20, 83.07);
insert into PAL_HGBT_PREDICT_TAB values (6, 16.71, 1337, 74.84, 94.99);
insert into PAL_HGBT_PREDICT_TAB values (7, 18.81, 17881, 70.66, 92.34);
insert into PAL_HGBT_PREDICT_TAB values (8, 20.74, 2319, 63.93, 95.08);
insert into PAL_HGBT_PREDICT_TAB values (9, 16.56, 18040, 14.45, 61.24);
insert into PAL_HGBT_PREDICT_TAB values (10, 18.55, 1147, 68.58, 97.90);
drop table #PAL_PARAMETER_TAB;
create local temporary column table #PAL_PARAMETER_TAB (
    "PARAM_NAME" varchar(100),
    "INT_VALUE" integer,
    "double_VALUE" double,
    "STRING_VALUE" varchar(100)
);
insert into #PAL_PARAMETER_TAB values ('THREAD_RATIO', null, 0.5, null);
insert into #PAL_PARAMETER_TAB values ('VERBOSE', 0, null, null);
call _SYS_AFL.PAL_HGBT_PREDICT(PAL_HGBT_PREDICT_TAB, PAL_HGBT_MODEL_TAB,
#PAL_PARAMETER_TAB, ?);
```

**Expected Result**

RESULT:

| | ID | SCORE | CONFIDENCE |
|---|---|---|---|
| 1 | 1 | 24.150756184709067 | ? |
| 2 | 2 | 19.235129769811287 | ? |
| 3 | 3 | 21.89492924174713 | ? |
| 4 | 4 | 18.52528898700058 | ? |
| 5 | 5 | 19.235129769811287 | ? |
| 6 | 6 | 19.416408449577126 | ? |
| 7 | 7 | 20.107270089162903 | ? |
| 8 | 8 | 24.150756184709067 | ? |
| 9 | 9 | 17.915141731669735 | ? |
| 10 | 10 | 21.0617752825179 | ? |

## 5.2.4.1 Early Stop of Hybrid Gradient Boosting Tree

Early stop is a technic to stop training before the model gets too complicated and cause overfitting.

Basically it monitors generalization performance of the model on another independent data set other than the training data set.

To achieve this, another data set called validation data set is needed. HGBT gets this data set by splitting the input data set into two parts.

And the generalization performance is evaluated by loss function on validation set.

### Relevant Parameters

- VALIDATION_SET_RATE
  Sampling rate of validation set
- STRATIFIED_VALIDATION_SET
  Whether to apply stratified method while sampling
- TOLERANT_ITER_NUM
  Number of successive deteriorating iterations before stopping.

### Example

```
set schema DM_PAL;
drop table PAL_HGBT_DATA_TAB;
create column table PAL_HGBT_DATA_TAB (
    "ATT1" double,
```

```
    "ATT2" double,
    "ATT3" double,
    "ATT4" double,
    "LABEL" nvarchar(50)
);
insert into PAL_HGBT_DATA_TAB values (1.0, 10.0, 100, 1.0, 'A');
insert into PAL_HGBT_DATA_TAB values (1.1, 10.1, 100, 1.0, 'A');
insert into PAL_HGBT_DATA_TAB values (1.2, 10.2, 100, 1.0, 'A');
insert into PAL_HGBT_DATA_TAB values (1.3, 10.4, 100, 1.0, 'A');
insert into PAL_HGBT_DATA_TAB values (1.2, 10.3, 100, 1.0, 'A');
insert into PAL_HGBT_DATA_TAB values (4.0, 40.0, 400, 4.0, 'B');
insert into PAL_HGBT_DATA_TAB values (4.1, 40.1, 400, 4.0, 'B');
insert into PAL_HGBT_DATA_TAB values (4.2, 40.2, 400, 4.0, 'B');
insert into PAL_HGBT_DATA_TAB values (4.3, 40.4, 400, 4.0, 'B');
insert into PAL_HGBT_DATA_TAB values (4.2, 40.3, 400, 4.0, 'A');
insert into PAL_HGBT_DATA_TAB values (9.0, 90.0, 900, 2.0, 'A');
insert into PAL_HGBT_DATA_TAB values (9.1, 90.1, 900, 1.0, 'B');
insert into PAL_HGBT_DATA_TAB values (9.2, 90.2, 900, 2.0, 'B');
insert into PAL_HGBT_DATA_TAB values (9.3, 90.4, 900, 1.0, 'B');
insert into PAL_HGBT_DATA_TAB values (9.2, 90.3, 900, 1.0, 'B');
insert into PAL_HGBT_DATA_TAB values (1.0, 10.0, 100, 1.0, 'A');
insert into PAL_HGBT_DATA_TAB values (1.1, 10.1, 100, 1.0, 'A');
insert into PAL_HGBT_DATA_TAB values (1.2, 10.2, 100, 1.0, 'A');
insert into PAL_HGBT_DATA_TAB values (1.3, 10.4, 100, 1.0, 'A');
insert into PAL_HGBT_DATA_TAB values (1.2, 10.3, 100, 1.0, 'A');
drop table PAL_PARAMETER_TAB;
create column table PAL_PARAMETER_TAB (
    "PARAM_NAME" nvarchar(100),
    "INT_VALUE" integer,
    "DOUBLE_VALUE" double,
    "STRING_VALUE" nvarchar(100)
);
insert into PAL_PARAMETER_TAB values ('VALIDATION_SET_RATE', null, 0.3, null);
insert into PAL_PARAMETER_TAB values ('STRATIFIED_VALIDATION_SET', 1, null,
null);
insert into PAL_PARAMETER_TAB values ('TOLERANT_ITER_NUM', 3, null, null);
do begin
    LT_DATA_TAB = select * from PAL_HGBT_DATA_TAB;
    LT_PARAM_TAB = select * from PAL_PARAMETER_TAB;
    call _SYS_AFL.PAL_HGBT(:LT_DATA_TAB, :LT_PARAM_TAB, LT_MODEL_TAB,
LT_IMP_TAB, LT_CONFUSE_TAB, LT_STAT_TAB, LT_CV_TAB);
    select * from :LT_MODEL_TAB;
    select * from :LT_IMP_TAB;
    select * from :LT_CONFUSE_TAB;
    select * from :LT_STAT_TAB;
    select * from :LT_CV_TAB;
    insert into PAL_HGBT_MODEL_TAB select * from :LT_MODEL_TAB;
end;
```

### 5.2.4.2 Feature Grouping for Hybrid Gradient Boosting Tree

It is common that data set contains sparse features, which means they have a large part of insignificant data (zero or nearly zero).

A set of features can also be sparse, which means at most one of them contains significant data in each data row. It usually happens in features that are measured in similar sense.

For example 3 features A, B and C appearing like following:

| A | B | C |
|---|---|---|
| 1.1 | 0.0 | 0.0 |
| 0.0 | 0.0 | 2.5 |
| 0.0 | 0.0 | 0.0 |
| 0.0 | -10.2 | 0.0 |

In above case, feature A, B, and C can group up as one that for each row only one data is registered. It can both reduce memory usage and accelerate the training process.

It is quite a complicated algorithm to find exact set of features that satisfy the requirement of feature grouping. HGBT employs a greedy algorithm that can find such sets approximately.

The requirement of features that can group up also can be relaxed that some violations can be accepted.

## Relevant Parameters

- FEATURE_GROUPING
  Indicates whether activate feature grouping preprocessing.
- TOLERANT_RATE
  Sets the maximum ratio of rows that can violate the requirement.

## Example

```
set schema DM_PAL;
drop table PAL_HGBT_DATA_TAB;
create column table PAL_HGBT_DATA_TAB (
    "ATT1" double,
    "ATT2" double,
    "ATT3" double,
    "ATT4" double,
    "LABEL" nvarchar(50)
);
insert into PAL_HGBT_DATA_TAB values (1.0, 0.0, 0.0, 1.0, 'A');
insert into PAL_HGBT_DATA_TAB values (1.1, 0.0, 0.0, 1.0, 'A');
insert into PAL_HGBT_DATA_TAB values (0.0, 10.2, 0.0, 1.0, 'A');
insert into PAL_HGBT_DATA_TAB values (1.3, 0.0, 0.0, 1.0, 'A');
insert into PAL_HGBT_DATA_TAB values (1.2, 0.0, 0.0, 1.0, 'A');
insert into PAL_HGBT_DATA_TAB values (0.0, 40.0, 400, 0.0, 'B');
insert into PAL_HGBT_DATA_TAB values (4.1, 0.0, 400, 0.0, 'B');
insert into PAL_HGBT_DATA_TAB values (4.2, 0.0, 400, 0.0, 'B');
insert into PAL_HGBT_DATA_TAB values (4.3, 0.0, 400, 0.0, 'B');
insert into PAL_HGBT_DATA_TAB values (0.0, 40.3, 400, 0.0, 'A');
insert into PAL_HGBT_DATA_TAB values (0.0, 90.0, 900, 0.0, 'A');
insert into PAL_HGBT_DATA_TAB values (0.0, 90.1, 0.0, 1.0, 'B');
insert into PAL_HGBT_DATA_TAB values (0.0, 90.2, 0.0, 2.0, 'B');
insert into PAL_HGBT_DATA_TAB values (0.0, 90.4, 0.0, 1.0, 'B');
insert into PAL_HGBT_DATA_TAB values (0.0, 90.3, 900, 0.0, 'B');
insert into PAL_HGBT_DATA_TAB values (1.0, 0.0, 100, 0.0, 'A');
insert into PAL_HGBT_DATA_TAB values (1.1, 0.0, 100, 0.0, 'A');
```

```
insert into PAL_HGBT_DATA_TAB values (1.2, 0.0, 100, 0.0, 'A');
insert into PAL_HGBT_DATA_TAB values (0.0, 10.4, 100, 0.0, 'A');
insert into PAL_HGBT_DATA_TAB values (1.2, 0.0, 100, 0.0, 'A');
drop table PAL_PARAMETER_TAB;
create column table PAL_PARAMETER_TAB (
    "PARAM_NAME" nvarchar(100),
    "INT_VALUE" integer,
    "DOUBLE_VALUE" double,
    "STRING_VALUE" nvarchar(100)
);
insert into PAL_PARAMETER_TAB values ('FEATURE_GROUPING', 1, null, null);
insert into PAL_PARAMETER_TAB values ('TOLERANT_RATE', null, 0.01, null);
do begin
    LT_DATA_TAB = select * from PAL_HGBT_DATA_TAB;
    LT_PARAM_TAB = select * from PAL_PARAMETER_TAB;
    call _SYS_AFL.PAL_HGBT(:LT_DATA_TAB, :LT_PARAM_TAB, LT_MODEL_TAB,
LT_IMP_TAB, LT_CONFUSE_TAB, LT_STAT_TAB, LT_CV_TAB);
    select * from :LT_MODEL_TAB;
    select * from :LT_IMP_TAB;
    select * from :LT_CONFUSE_TAB;
    select * from :LT_STAT_TAB;
    select * from :LT_CV_TAB;
    insert into PAL_HGBT_MODEL_TAB select * from :LT_MODEL_TAB;
end;
```

### 5.2.4.3 Histogram Splitting Method for Hybrid Gradient Boosting Tree

One optimization while HGBT splitting nodes uses histogram to accelerate the training process.

It is an approximate algorithm that not only reduces the time cost but also reduces the memory cost.

To be specific, while HGBT tries to split a node in tree, it first builds histogram of that node by put feature values into bins, then evaluates splitting points by these bins. Because the number of bins is usually a lot fewer than the number of data in node, this method can accelerate the splitting process a lot. Though building the histogram still needs to visit all data in node, but it is a faster process because it only involves scanning and adding things up. Another optimization in building the histogram is that histogram of one node can always be built by subtracting histogram of its sibling from histogram of its parent. So, we can always choose to build the histogram of node that contains less data and build histogram of its sibling by subtraction, which costs less time.

#### Relevant Parameters

- SPLIT_METHOD
  To use histogram splitting method, you need to set SPLIT_METHOD='histogram'
- MAX_BIN_NUM
  This method sets the maximum number of bins while building histogram during splitting.
  As mentioned before, histogram splitting method is an approximate algorithm that does not evaluate all potential splitting points, which brings one important setting the number of bins while building histogram. Parameter MAX_BIN_NUM is exactly for this purpose. The bigger number of bins, the more potential splitting points are evaluated, and the more time is needed. The default value of MAX_BIN_NUM is 256. It is suggested to use this default value first, then adjust it by the fitting result of model accordingly.

When it comes to categorical features, though histogram splitting method cannot be applied to them directly, HGBT will combine sparse categories if the number of categories is more than MAX_BIN_NUM and reduce the number of categories after all.

## Example

```
set schema DM_PAL;
drop table PAL_HGBT_DATA_TAB;
create column table PAL_HGBT_DATA_TAB (
    "ATT1" double,
    "ATT2" double,
    "ATT3" double,
    "ATT4" double,
    "LABEL" nvarchar(50)
);
insert into PAL_HGBT_DATA_TAB values (1.0, 10.0, 100, 1.0, 'A');
insert into PAL_HGBT_DATA_TAB values (1.1, 10.1, 100, 1.0, 'A');
insert into PAL_HGBT_DATA_TAB values (1.2, 10.2, 100, 1.0, 'A');
insert into PAL_HGBT_DATA_TAB values (1.3, 10.4, 100, 1.0, 'A');
insert into PAL_HGBT_DATA_TAB values (1.2, 10.3, 100, 1.0, 'A');
insert into PAL_HGBT_DATA_TAB values (4.0, 40.0, 400, 4.0, 'B');
insert into PAL_HGBT_DATA_TAB values (4.1, 40.1, 400, 4.0, 'B');
insert into PAL_HGBT_DATA_TAB values (4.2, 40.2, 400, 4.0, 'B');
insert into PAL_HGBT_DATA_TAB values (4.3, 40.4, 400, 4.0, 'B');
insert into PAL_HGBT_DATA_TAB values (4.2, 40.3, 400, 4.0, 'A');
insert into PAL_HGBT_DATA_TAB values (9.0, 90.0, 900, 2.0, 'A');
insert into PAL_HGBT_DATA_TAB values (9.1, 90.1, 900, 1.0, 'B');
insert into PAL_HGBT_DATA_TAB values (9.2, 90.2, 900, 2.0, 'B');
insert into PAL_HGBT_DATA_TAB values (9.3, 90.4, 900, 1.0, 'B');
insert into PAL_HGBT_DATA_TAB values (9.2, 90.3, 900, 1.0, 'B');
drop table PAL_PARAMETER_TAB;
create column table PAL_PARAMETER_TAB (
    "PARAM_NAME" nvarchar(100),
    "INT_VALUE" integer,
    "DOUBLE_VALUE" double,
    "STRING_VALUE" nvarchar(100)
);
insert into PAL_PARAMETER_TAB values ('SPLIT_METHOD', null, null, 'histogram');
-- set splitting method
insert into PAL_PARAMETER_TAB values ('MAX_BIN_NUM', 256, null, null); -- the
same value as default, can be omitted
drop table PAL_HGBT_MODEL_TAB; -- for predict followed
create column table PAL_HGBT_MODEL_TAB (
    "ROW_INDEX" integer,
    "TREE_INDEX" integer,
    "MODEL_CONTENT" nclob
);
do begin
    LT_DATA_TAB = select * from PAL_HGBT_DATA_TAB;
    LT_PARAM_TAB = select * from PAL_PARAMETER_TAB;
    call _SYS_AFL.PAL_HGBT(:LT_DATA_TAB, :LT_PARAM_TAB, LT_MODEL_TAB,
LT_IMP_TAB, LT_CONFUSE_TAB, LT_STAT_TAB, LT_CV_TAB);
    select * from :LT_MODEL_TAB;
    select * from :LT_IMP_TAB;
    select * from :LT_CONFUSE_TAB;
    select * from :LT_STAT_TAB;
    select * from :LT_CV_TAB;
    insert into PAL_HGBT_MODEL_TAB select * from :LT_MODEL_TAB;
end;
```

## 5.2.5 KNN

K-Nearest Neighbor (KNN) is a memory-based classification or regression method with no explicit training phase. It assumes similar instances should have similar labels or values.

In the prediction phase, given a query sample x, its top K nearest samples are found in the training set first, then the label or value of x is assigned with some metric using the K nearest neighbors. In order to speed up the search, the KD-tree searching method is provided.

### Prerequisites

- The first column of the training data and prediction data is an ID column.
- The training and prediction data does not contain null value.

### _SYS_AFL.PAL_KNN

This is a classification function using the KNN algorithm.

**Overview of All Tables**

| Position | Table Type Name | Parameter Type |
|---|---|---|
| 1 | `<TRAINING DATA table>` | IN |
| 2 | `<PREDICTION DATA table>` | IN |
| 3 | `<PARAMETER table>` | IN |
| 4 | `<RESULT table>` | OUT |
| 5 | `<STATISTIC table>` | OUT |

**Input Tables**

| Table | Column | Reference | Data Type | Description |
|---|---|---|---|---|
| TRAINING DATA | 1st column | ID1 | INTEGER, BIGINT, VARCHAR, or NVARCHAR | ID of training data |
| | Other columns | None | INTEGER, DOUBLE, DECIMAL(p, s), VARCHAR or NVARCHAR | Feature data and dependent variable |

| Table | Column | Reference | Data Type | Description |
|---|---|---|---|---|
| PREDICTION DATA | 1st column | ID2 | INTEGER, BIGINT, VARCHAR, or NVARCHAR | ID of prediction data |
| | Feature columns | None | INTEGER, DOUBLE, DECIMAL(p, s), VARCHAR or NVARCHAR | Feature data |

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| K_NEAREST_NEIGHBOURS | INTEGER | 1 | Number of nearest neighbors (k). | |
| THREAD_RATIO | DOUBLE | 0.0 | Specifies the ratio of total number of threads that can be used by this function. The range of this parameter is from 0 to 1, where 0 means only using 1 thread, and 1 means using at most all the currently available threads. Values outside this range are ignored and this function heuristically determines the number of threads to use. | |
| ATTRIBUTE_NUM | INTEGER | Number of all attributes | Number of attributes. | |
| FUNCTIONALITY | INTEGER | 0 | Specifies the prediction type:<br>• 0: Classification<br>• 1: Regression | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| DEPENDENT_VARIABLE | NVARCHAR | No default value | specifies the dependent variable by name, or the second column in data table is considered to be the dependent variable. | |
| CATEGORICAL_VARIABLE | NVARCHAR | No default value | Indicates whether or not a column data is actually corresponding to a category variable even the data type of this column is INTEGER.<br><br>By default, VARCHAR or NVARCHAR is category variable, and INTEGER or DOUBLE is continuous variable. | |
| STRING_VARIABLE | NVARCHAR | No default value | Indicates a string column storing not categorical data. Levenshtein distance is used to calculate similarity between two strings. Ignored if it is not a string column. | Not valid if DISTANCE_LEVEL is set to 6 (COSINE). |
| VARIABLE_WEIGHT | NVARCHAR for variable name and DOUBLE for weight value | No default value | Specifies the weight of a variable participating in distance calculation. The value must be greater or equal to 0. Defaults to 1 for variables not specified. | |
| VOTING_TYPE | INTEGER | 1 | Voting type:<br>• 0: Majority voting<br>• 1: Distance-weighted voting | Only available while performing classification. |
| AGGREGATE_TYPE | INTEGER | 1 | Aggregate type:<br>• 0: Average<br>• 1: Distance-weighted average | Only available while performing regression. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| STAT_INFO | INTEGER | 1 | • 0: STATISTIC table will be empty.<br>• 1: Statistic information is displayed in the STATISTIC table. | |
| DISTANCE_LEVEL | INTEGER | 2 | Computes the distance between the train data and the test data point.<br>• 1: Manhattan distance<br>• 2: Euclidean distance<br>• 3: Minkowski distance<br>• 4: Chebyshev distance<br>• 6: Cosine distance | |
| MINKOWSKI_POWER | DOUBLE | 3.0 | When you use the Minkowski distance, this parameter controls the value of power. | Only valid when `DISTANCE_LEVEL` is 3. |
| CATEGORY_WEIGHTS | DOUBLE | 0.707 | Represents the weight of category attributes. The value must be greater or equal to 0. | |
| METHOD | INTEGER | 0 | Searching method.<br>• 0: Brute force searching<br>• 1: KD-tree searching | When `DISTANCE_LEVEL` is 6, using KD-tree searching will not have much help. |

## Output Tables

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| RESULT | 1st column | ID2 data type | ID2 column name | Prediction data ID |
| | 2nd column | NVARCHAR | TARGET | Predicted label or value |

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| STATISTICS | 1st column | ID2 data type | TEST_ + ID2 column name | Prediction data ID |
| | 2nd column | INTEGER | K | K number |
| | 3rd column | ID1 data type | TRAIN_ + ID1 column name | Training data ID |
| | 4th column | DOUBLE | DISTANCE | Distance |

### Examples

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

#### Example 1: classification

```
set schema DM_PAL;
drop table PAL_KNN_DATA_TAB;
create column table PAL_KNN_DATA_TAB (
    "ID" integer,
    "X1" integer,
    "X2" double,
    "X3" varchar(100),
    "TYPE" integer
);
insert into PAL_KNN_DATA_TAB values (0, 2, 1, 'A', 1);
insert into PAL_KNN_DATA_TAB values (1, 3, 10, 'A', 10);
insert into PAL_KNN_DATA_TAB values (2, 3, 10, 'B', 10);
insert into PAL_KNN_DATA_TAB values (3, 3, 10, 'C', 1);
insert into PAL_KNN_DATA_TAB values (4, 1, 1000, 'C', 1);
insert into PAL_KNN_DATA_TAB values (5, 1, 1000, 'A', 10);
insert into PAL_KNN_DATA_TAB values (6, 1, 1000, 'B', 99);
insert into PAL_KNN_DATA_TAB values (7, 1, 999, 'A', 99);
insert into PAL_KNN_DATA_TAB values (8, 1, 999, 'B', 10);
insert into PAL_KNN_DATA_TAB values (9, 1, 1000, 'C', 10);
drop table PAL_KNN_CLASSDATA_TAB;
create column table PAL_KNN_CLASSDATA_TAB(
    "ID" integer,
    "X1" integer,
    "X2" double,
    "X3" varchar(100)
);
insert into PAL_KNN_CLASSDATA_TAB values (0, 2, 1, 'A');
insert into PAL_KNN_CLASSDATA_TAB values (1, 1, 10, 'C');
insert into PAL_KNN_CLASSDATA_TAB values (2, 1, 11, 'B');
insert into PAL_KNN_CLASSDATA_TAB values (3, 3, 15000, 'C');
insert into PAL_KNN_CLASSDATA_TAB values (4, 2, 1000, 'C');
insert into PAL_KNN_CLASSDATA_TAB values (5, 1, 1001, 'A');
insert into PAL_KNN_CLASSDATA_TAB values (6, 1, 999, 'A');
insert into PAL_KNN_CLASSDATA_TAB values (7, 3, 999, 'B');
drop table #PAL_PARAMETER_TAB;
create local temporary column table #PAL_PARAMETER_TAB (
    "PARAM_NAME" nvarchar(256),
    "INT_VALUE" integer,
    "DOUBLE_VALUE" double,
    "STRING_VALUE" nvarchar(1000)
```

```
);
insert into #PAL_PARAMETER_TAB values ('THREAD_RATIO', null, 0.1, null);
insert into #PAL_PARAMETER_TAB values ('DEPENDENT_VARIABLE', null, null, 'TYPE');
insert into #PAL_PARAMETER_TAB values ('CATEGORICAL_VARIABLE', null, null, 'X1');
insert into #PAL_PARAMETER_TAB values ('K_NEAREST_NEIGHBOURS', 3, null, null);
insert into #PAL_PARAMETER_TAB values ('VOTING_TYPE', 0, null, null);
insert into #PAL_PARAMETER_TAB values ('METHOD', 1, null, null);
insert into #PAL_PARAMETER_TAB values ('STRING_VARIABLE', null, null, 'X3');
insert into #PAL_PARAMETER_TAB values ('VARIABLE_WEIGHT', null, 0.5, 'X3');
call _SYS_AFL.PAL_KNN(PAL_KNN_DATA_TAB, PAL_KNN_CLASSDATA_TAB,
"#PAL_PARAMETER_TAB", ?, ?);
```

**Expected Result**

RESULT:

| | ID | TARGET |
|---|---|---|
| 1 | 0 | 10 |
| 2 | 1 | 10 |
| 3 | 2 | 10 |
| 4 | 3 | 1 |
| 5 | 4 | 1 |
| 6 | 5 | 1 |
| 7 | 6 | 10 |
| 8 | 7 | 99 |

STATISTIC:

| | TEST_ID | K | TRAIN_ID | DISTANCE |
|---|---|---|---|---|
| 1 | 0 | 1 | 0 | 0 |
| 2 | 0 | 2 | 1 | 9.999848988597778 |
| 3 | 0 | 3 | 2 | 10.414 |
| 4 | 1 | 1 | 3 | 0.9998489885977783 |
| 5 | 1 | 2 | 1 | 1.414 |
| 6 | 1 | 3 | 2 | 1.414 |
| 7 | 2 | 1 | 2 | 1.9998489885977784 |
| 8 | 2 | 2 | 1 | 2.4139999999999997 |
| 9 | 2 | 3 | 3 | 2.4139999999999997 |
| 10 | 3 | 1 | 4 | 14,000.999848988598 |
| 11 | 3 | 2 | 9 | 14,000.999848988598 |
| 12 | 3 | 3 | 6 | 14,001.414 |
| 13 | 4 | 1 | 4 | 0.9998489885977783 |
| 14 | 4 | 2 | 9 | 0.9998489885977783 |
| 15 | 4 | 3 | 6 | 1.414 |
| 16 | 5 | 1 | 5 | 1 |
| 17 | 5 | 2 | 4 | 1.9998489885977784 |
| 18 | 5 | 3 | 6 | 1.9998489885977784 |
| 19 | 6 | 1 | 7 | 0 |
| 20 | 6 | 2 | 8 | 0.9998489885977783 |
| 21 | 6 | 3 | 5 | 1 |
| 22 | 7 | 1 | 8 | 0.9998489885977783 |
| 23 | 7 | 2 | 7 | 1.414 |
| 24 | 7 | 3 | 6 | 1.9998489885977784 |

**Example 2: regression**

```
set schema DM_PAL;
drop table PAL_KNN_DATA_TAB;
create column table PAL_KNN_DATA_TAB (
    "ID" integer,
    "X1" integer,
```

```
    "X2" double,
    "X3" varchar(100),
    "VALUE" double
);
insert into PAL_KNN_DATA_TAB values (0, 2, 1, 'A', 1);
insert into PAL_KNN_DATA_TAB values (1, 3, 10, 'A', 10);
insert into PAL_KNN_DATA_TAB values (2, 3, 10, 'B', 10);
insert into PAL_KNN_DATA_TAB values (3, 3, 10, 'C', 1);
insert into PAL_KNN_DATA_TAB values (4, 1, 1000, 'C', 1);
insert into PAL_KNN_DATA_TAB values (5, 1, 1000, 'A', 10);
insert into PAL_KNN_DATA_TAB values (6, 1, 1000, 'B', 99);
insert into PAL_KNN_DATA_TAB values (7, 1, 999, 'A', 99);
insert into PAL_KNN_DATA_TAB values (8, 1, 999, 'B', 10);
insert into PAL_KNN_DATA_TAB values (9, 1, 1000, 'C', 10);
drop table PAL_KNN_CLASSDATA_TAB;
create column table PAL_KNN_CLASSDATA_TAB(
    "ID" integer,
    "X1" integer,
    "X2" double,
    "X3" VARCHAR(100)
);
insert into PAL_KNN_CLASSDATA_TAB values (0, 2, 1, 'A');
insert into PAL_KNN_CLASSDATA_TAB values (1, 1, 10, 'C');
insert into PAL_KNN_CLASSDATA_TAB values (2, 1, 11, 'B');
insert into PAL_KNN_CLASSDATA_TAB values (3, 3, 15000, 'C');
insert into PAL_KNN_CLASSDATA_TAB values (4, 2, 1000, 'C');
insert into PAL_KNN_CLASSDATA_TAB values (5, 1, 1001, 'A');
insert into PAL_KNN_CLASSDATA_TAB values (6, 1, 999, 'A');
insert into PAL_KNN_CLASSDATA_TAB values (7, 3, 999, 'B');
drop table #PAL_PARAMETER_TAB;
create local temporary column table #PAL_PARAMETER_TAB (
    "PARAM_NAME" nvarchar(256),
    "INT_VALUE" integer,
    "DOUBLE_VALUE" double,
    "STRING_VALUE" nvarchar(1000)
);
insert into #PAL_PARAMETER_TAB values ('THREAD_RATIO', null, 0.1, null);
insert into #PAL_PARAMETER_TAB values ('DEPENDENT_VARIABLE', null, null,
'VALUE');
insert into #PAL_PARAMETER_TAB values ('CATEGORICAL_VARIABLE', null, null, 'X1');
insert into #PAL_PARAMETER_TAB values ('K_NEAREST_NEIGHBOURS', 3, null, null);
insert into #PAL_PARAMETER_TAB values ('AGGREGATE_TYPE', 0, null, null);
insert into #PAL_PARAMETER_TAB values ('METHOD', 1, null, null);
insert into #PAL_PARAMETER_TAB values ('FUNCTIONALITY', 1, null, null);
insert into #PAL_PARAMETER_TAB values ('STRING_VARIABLE', null, null, 'X3');
insert into #PAL_PARAMETER_TAB values ('VARIABLE_WEIGHT', null, 0.5, 'X3');
call _SYS_AFL.PAL_KNN(PAL_KNN_DATA_TAB, PAL_KNN_CLASSDATA_TAB,
"#PAL_PARAMETER_TAB", ?, ?);
```

### Expected Result

RESULT:

| | ID | TARGET |
|---|---|---|
| 1 | 0 | 7 |
| 2 | 1 | 7 |
| 3 | 2 | 7 |
| 4 | 3 | 36.666666666666664 |
| 5 | 4 | 36.666666666666664 |
| 6 | 5 | 36.666666666666664 |
| 7 | 6 | 39.666666666666664 |
| 8 | 7 | 69.33333333333333 |

STATISTIC:

| | TEST_ID | K | TRAIN_ID | DISTANCE |
|---|---|---|---|---|
| 1 | 0 | 1 | 0 | 0 |
| 2 | 0 | 2 | 1 | 9.999848988597778 |
| 3 | 0 | 3 | 2 | 10.414 |
| 4 | 1 | 1 | 3 | 0.9998489885977783 |
| 5 | 1 | 2 | 1 | 1.414 |
| 6 | 1 | 3 | 2 | 1.414 |
| 7 | 2 | 1 | 2 | 1.9998489885977784 |
| 8 | 2 | 2 | 1 | 2.4139999999999997 |
| 9 | 2 | 3 | 3 | 2.4139999999999997 |
| 10 | 3 | 1 | 4 | 14,000.999848988598 |
| 11 | 3 | 2 | 9 | 14,000.999848988598 |
| 12 | 3 | 3 | 6 | 14,001.414 |
| 13 | 4 | 1 | 4 | 0.9998489885977783 |
| 14 | 4 | 2 | 9 | 0.9998489885977783 |
| 15 | 4 | 3 | 6 | 1.414 |
| 16 | 5 | 1 | 5 | 1 |
| 17 | 5 | 2 | 4 | 1.9998489885977784 |
| 18 | 5 | 3 | 6 | 1.9998489885977784 |
| 19 | 6 | 1 | 7 | 0 |
| 20 | 6 | 2 | 8 | 0.9998489885977783 |
| 21 | 6 | 3 | 5 | 1 |
| 22 | 7 | 1 | 8 | 0.9998489885977783 |
| 23 | 7 | 2 | 7 | 1.414 |
| 24 | 7 | 3 | 6 | 1.9998489885977784 |

## KNN with Interpretation

It is same as KNN except that it gives interpretation of prediction results.

### Prerequisites

- The first column of the training data and prediction data is an ID column.
- The training and prediction data does not contain null value.

### _SYS_AFL.PAL_KNN_INTERPRET

This is a classification function using KNN algorithm and it gives interpretation of prediction results.

### Overviews of All Tables

| Position | Table type name | Parameter type |
|---|---|---|
| 1 | `<TRAINING DATA table>` | IN |
| 2 | `<PREDICTION DATA table>` | IN |
| 3 | `<PARAMETER table>` | IN |
| 4 | `<RESULT table>` | OUT |
| 5 | `<STATISTIC table>` | OUT |

### Input Table

| Table | Column | Reference | Data type | description |
|---|---|---|---|---|
| TRAINING DATA | 1st column | ID1 | INTEGER, BIGINT, VARCHAR, or NVARCHAR | ID of training data |
| | Other columns | None | INTEGER, DOUBLE, DECIMAL(p, s), VARCHAR, or NVARCHAR | Feature data and dependent variable |
| PREDICTION DATA | 1st column | ID2 | INTEGER, BIGINT, VARCHAR, or NVARCHAR | ID of prediction data |
| | Feature columns | None | INTEGER, DOUBLE, DECIMAL(p, s), VARCHAR, or NVARCHAR | Feature data |

### Parameter Table

> **Note**
>
> The following parameters are only for interpretation functionality. Other parameters are the same as KNN.

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use default value.

| Name | Data type | Default value | Description | Dependency |
|---|---|---|---|---|
| SAMPLESIZE | INTEGER | 0 | Number of sampled combinations of features. If it equals to 0, then it will be determined by algorithm heuristically. | It would be better to set it to a number greater than the number of features. |
| TOP_K_ATTRIBUTIONS | INTEGER | 10 | Outputs the attributions of top k features which contribute the most. | |
| SEED | INTEGER | 0 | The seed for random number generating.<br>• 0: current time<br>• Others: the seed | |

### Output Table

| Table | Column | Data type | Column name | Description |
|---|---|---|---|---|
| RESULT | 1st column | ID2 data type | ID2 column name | Prediction data ID |
| | 2nd column | NVARCHAR | TARGET | Predicted label or value |
| | 3rd column | NVARCHAR | REASON_CODE | Interpretation of result |
| STATISTICS | 1st column | ID2 data type | TEST_ + ID2 column name | Prediction data ID |
| | 2nd column | INTEGER | K | K number |
| | 3rd column | ID1 data type | TRAIN_ + ID1 column name | Training data ID |
| | 4th column | DOUBLE | DISTANCE | Distance |

### Examples

Assume that:

- `DM_PAL` is a schema belonging to USER1; and
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

Example 1: classification

```
set schema DM_PAL;
drop table PAL_KNN_DATA_TAB;
create column table PAL_KNN_DATA_TAB (
    "ID" integer,
    "X1" integer,
    "X2" double,
    "X3" varchar(100),
    "TYPE" integer
);
insert into PAL_KNN_DATA_TAB values (0, 2, 1, 'A', 1);
insert into PAL_KNN_DATA_TAB values (1, 3, 10, 'A', 10);
insert into PAL_KNN_DATA_TAB values (2, 3, 10, 'B', 10);
insert into PAL_KNN_DATA_TAB values (3, 3, 10, 'C', 1);
insert into PAL_KNN_DATA_TAB values (4, 1, 1000, 'C', 1);
insert into PAL_KNN_DATA_TAB values (5, 1, 1000, 'A', 10);
insert into PAL_KNN_DATA_TAB values (6, 1, 1000, 'B', 99);
insert into PAL_KNN_DATA_TAB values (7, 1, 999, 'A', 99);
insert into PAL_KNN_DATA_TAB values (8, 1, 999, 'B', 10);
insert into PAL_KNN_DATA_TAB values (9, 1, 1000, 'C', 10);
drop table PAL_KNN_CLASSDATA_TAB;
create column table PAL_KNN_CLASSDATA_TAB(
    "ID" integer,
    "X1" integer,
    "X2" double,
    "X3" varchar(100)
);
insert into PAL_KNN_CLASSDATA_TAB values (0, 2, 1, 'A');
insert into PAL_KNN_CLASSDATA_TAB values (1, 1, 10, 'C');
insert into PAL_KNN_CLASSDATA_TAB values (2, 1, 11, 'B');
insert into PAL_KNN_CLASSDATA_TAB values (3, 3, 15000, 'C');
insert into PAL_KNN_CLASSDATA_TAB values (4, 2, 1000, 'C');
insert into PAL_KNN_CLASSDATA_TAB values (5, 1, 1001, 'A');
insert into PAL_KNN_CLASSDATA_TAB values (6, 1, 999, 'A');
insert into PAL_KNN_CLASSDATA_TAB values (7, 3, 999, 'B');
drop table #PAL_PARAMETER_TAB;
create local temporary column table #PAL_PARAMETER_TAB (
    "PARAM_NAME" nvarchar(256),
    "INT_VALUE" integer,
    "DOUBLE_VALUE" double,
    "STRING_VALUE" nvarchar(1000)
);
insert into #PAL_PARAMETER_TAB values ('THREAD_RATIO', null, 0.1, null);
insert into #PAL_PARAMETER_TAB values ('DEPENDENT_VARIABLE', null, null, 'TYPE');
insert into #PAL_PARAMETER_TAB values ('CATEGORICAL_VARIABLE', null, null, 'X1');
insert into #PAL_PARAMETER_TAB values ('K_NEAREST_NEIGHBOURS', 3, null, null);
insert into #PAL_PARAMETER_TAB values ('VOTING_TYPE', 0, null, null);
insert into #PAL_PARAMETER_TAB values ('METHOD', 1, null, null);
insert into #PAL_PARAMETER_TAB values ('STRING_VARIABLE', null, null, 'X3');
insert into #PAL_PARAMETER_TAB values ('VARIABLE_WEIGHT', null, 0.5, 'X3');
insert into #PAL_PARAMETER_TAB values ('TOP_K_ATTRIBUTIONS', 3, null, null);
insert into #PAL_PARAMETER_TAB values ('SEED', 1, null, null);
insert into #PAL_PARAMETER_TAB values ('SAMPLESIZE', 10, null, null);
call _SYS_AFL.PAL_KNN_INTERPRET(PAL_KNN_DATA_TAB, PAL_KNN_CLASSDATA_TAB,
"#PAL_PARAMETER_TAB", ?, ?);
```

**Expected Results**

RESULT:

| | ID | TARGET | REASON_CODE |
|---|---|---|---|
| 1 | 0 | 10 | [{"attr":"X1","pct":56.5,"val":0.086666666666666614},{"attr":"X3","... |
| 2 | 1 | 10 | [{"attr":"X1","pct":75.699999999999989,"val":0.186666666666666... |
| 3 | 2 | 10 | [{"attr":"X1","pct":85.5,"val":0.17666666666666658},{"attr":"X3","p... |
| 4 | 3 | 1 | [{"attr":"X3","pct":55.300000000000004,"val":0.069999999999999... |
| 5 | 4 | 1 | [{"attr":"X3","pct":55.300000000000004,"val":0.069999999999999... |
| 6 | 5 | 1 | [{"attr":"X1","pct":100.0,"val":0.066666666666666638}] |
| 7 | 6 | 10 | [{"attr":"X3","pct":62.5,"val":0.083333333333333315},{"attr":"X2","... |
| 8 | 7 | 99 | [{"attr":"X1","pct":66.099999999999994,"val":0.259999999999999... |

STATISTIC:

| | TEST_ID | K | TRAIN_ID | DISTANCE |
|---|---|---|---|---|
| 1 | 0 | 1 | 0 | 0 |
| 2 | 0 | 2 | 1 | 9.999848988597778 |
| 3 | 0 | 3 | 2 | 10.013727177257... |
| 4 | 1 | 1 | 3 | 0.9998489885977... |
| 5 | 1 | 2 | 1 | 1.4998489885977... |
| 6 | 1 | 3 | 2 | 1.4998489885977... |
| 7 | 2 | 1 | 2 | 1.9998489885977... |
| 8 | 2 | 2 | 1 | 2.1178829773476... |
| 9 | 2 | 3 | 3 | 2.1178829773476... |
| 10 | 3 | 1 | 4 | 14,000.999848988... |
| 11 | 3 | 2 | 9 | 14,000.999848988... |
| 12 | 3 | 3 | 6 | 14,000.999857917... |
| 13 | 4 | 1 | 4 | 0.9998489885977... |
| 14 | 4 | 2 | 9 | 0.9998489885977... |
| 15 | 4 | 3 | 6 | 1.4998489885977... |
| 16 | 5 | 1 | 5 | 1 |
| 17 | 5 | 2 | 4 | 1.118033988749895 |
| 18 | 5 | 3 | 6 | 1.118033988749895 |
| 19 | 6 | 1 | 7 | 0 |
| 20 | 6 | 2 | 8 | 0.5 |

Example 2: regression

```
set schema DM_PAL;
drop table PAL_KNN_DATA_TAB;
create column table PAL_KNN_DATA_TAB (
    "ID" integer,
    "X1" integer,
    "X2" double,
    "X3" varchar(100),
    "VALUE" double
);
insert into PAL_KNN_DATA_TAB values (0, 2, 1, 'A', 1);
insert into PAL_KNN_DATA_TAB values (1, 3, 10, 'A', 10);
insert into PAL_KNN_DATA_TAB values (2, 3, 10, 'B', 10);
insert into PAL_KNN_DATA_TAB values (3, 3, 10, 'C', 1);
insert into PAL_KNN_DATA_TAB values (4, 1, 1000, 'C', 1);
```

```
insert into PAL_KNN_DATA_TAB values (5, 1, 1000, 'A', 10);
insert into PAL_KNN_DATA_TAB values (6, 1, 1000, 'B', 99);
insert into PAL_KNN_DATA_TAB values (7, 1, 999, 'A', 99);
insert into PAL_KNN_DATA_TAB values (8, 1, 999, 'B', 10);
insert into PAL_KNN_DATA_TAB values (9, 1, 1000, 'C', 10);
drop table PAL_KNN_CLASSDATA_TAB;
create column table PAL_KNN_CLASSDATA_TAB(
    "ID" integer,
    "X1" integer,
    "X2" double,
    "X3" varchar(100)
);
insert into PAL_KNN_CLASSDATA_TAB values (0, 2, 1, 'A');
insert into PAL_KNN_CLASSDATA_TAB values (1, 1, 10, 'C');
insert into PAL_KNN_CLASSDATA_TAB values (2, 1, 11, 'B');
insert into PAL_KNN_CLASSDATA_TAB values (3, 3, 15000, 'C');
insert into PAL_KNN_CLASSDATA_TAB values (4, 2, 1000, 'C');
insert into PAL_KNN_CLASSDATA_TAB values (5, 1, 1001, 'A');
insert into PAL_KNN_CLASSDATA_TAB values (6, 1, 999, 'A');
insert into PAL_KNN_CLASSDATA_TAB values (7, 3, 999, 'B');
drop table #PAL_PARAMETER_TAB;
create local temporary column table #PAL_PARAMETER_TAB (
    "PARAM_NAME" nvarchar(256),
    "INT_VALUE" integer,
    "DOUBLE_VALUE" double,
    "STRING_VALUE" nvarchar(1000)
);
insert into #PAL_PARAMETER_TAB values ('THREAD_RATIO', null, 0.1, null);
insert into #PAL_PARAMETER_TAB values ('DEPENDENT_VARIABLE', null, null,
'VALUE');
insert into #PAL_PARAMETER_TAB values ('CATEGORICAL_VARIABLE', null, null, 'X1');
insert into #PAL_PARAMETER_TAB values ('K_NEAREST_NEIGHBOURS', 3, null, null);
insert into #PAL_PARAMETER_TAB values ('AGGREGATE_TYPE', 0, null, null);
insert into #PAL_PARAMETER_TAB values ('METHOD', 1, null, null);
insert into #PAL_PARAMETER_TAB values ('FUNCTIONALITY', 1, null, null);
insert into #PAL_PARAMETER_TAB values ('STRING_VARIABLE', null, null, 'X3');
insert into #PAL_PARAMETER_TAB values ('VARIABLE_WEIGHT', null, 0.5, 'X3');
insert into #PAL_PARAMETER_TAB values ('TOP_K_ATTRIBUTIONS', 3, null, null);
insert into #PAL_PARAMETER_TAB values ('SEED', 1, null, null);
insert into #PAL_PARAMETER_TAB values ('SAMPLESIZE', 8, null, null);
call _SYS_AFL.PAL_KNN_INTERPRET(PAL_KNN_DATA_TAB, PAL_KNN_CLASSDATA_TAB,
"#PAL_PARAMETER_TAB", ?, ?);
```

*Expected result*

RESULT:

| | ID | TARGET | REASON_CODE |
|---|---|---|---|
| 1 | 0 | 7 | [{"attr":"X1","pct":89.099999999999994,"val":-21.10666666... |
| 2 | 1 | 7 | [{"attr":"X1","pct":85.299999999999997,"val":-20.20666666... |
| 3 | 2 | 7 | [{"attr":"X1","pct":89.299999999999997,"val":-22.25666666... |
| 4 | 3 | 36.6666666666... | [{"attr":"X1","pct":73.099999999999994,"val":9.4599999999... |
| 5 | 4 | 36.6666666666... | [{"attr":"X1","pct":73.099999999999994,"val":9.4599999999... |
| 6 | 5 | 36.6666666666... | [{"attr":"X1","pct":77.5,"val":9.1899999999999959},{"attr":"X... |
| 7 | 6 | 39.6666666666... | [{"attr":"X1","pct":76.5,"val":17.790000000000003},{"attr":"X... |
| 8 | 7 | 69.3333333333... | [{"attr":"X1","pct":66.900000000000006,"val":24.909999999... |

STATISTIC:

| | TEST_ID | K | TRAIN_ID | DISTANCE |
|---|---|---|---|---|
| 1 | 0 | 1 | 0 | 0 |
| 2 | 0 | 2 | 1 | 9.999848988597778 |
| 3 | 0 | 3 | 2 | 10.013727177257... |
| 4 | 1 | 1 | 3 | 0.9998489885977... |
| 5 | 1 | 2 | 1 | 1.4998489885977... |
| 6 | 1 | 3 | 2 | 1.4998489885977... |
| 7 | 2 | 1 | 2 | 1.9998489885977... |
| 8 | 2 | 2 | 1 | 2.1178829773476... |
| 9 | 2 | 3 | 3 | 2.1178829773476... |
| 10 | 3 | 1 | 4 | 14,000.999848988... |
| 11 | 3 | 2 | 9 | 14,000.999848988... |
| 12 | 3 | 3 | 6 | 14,000.999857917... |
| 13 | 4 | 1 | 4 | 0.9998489885977... |
| 14 | 4 | 2 | 9 | 0.9998489885977... |
| 15 | 4 | 3 | 6 | 1.4998489885977... |
| 16 | 5 | 1 | 5 | 1 |
| 17 | 5 | 2 | 4 | 1.118033988749895 |
| 18 | 5 | 3 | 6 | 1.118033988749895 |
| 19 | 6 | 1 | 7 | 0 |
| 20 | 6 | 2 | 8 | 0.5 |

## KNN Model Evaluation and Parameter Selection

A function specific for model evaluation and parameter selection of KNN algorithm.

### Prerequisites

- The input data does not contain null value.

## _SYS_AFL.PAL_KNN_CV

This is a model evaluation and parameter selection function specifically for the KNN algorithm. Unlike other functions that support model evaluation and parameter selection, it does not have training functionality. For more information, see the section on model evaluation and parameter selection.

### Overview of All Tables

| Position | Table Type Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<STATISTIC table>` | OUT |
| 4 | `<OPTIMAL_PARAM table>` | OUT |

### Input Tables

| Table | Column | Data Type | Description |
|---|---|---|---|
| DATA | 1st column | INTEGER, BIGINT, VARCHAR, or NVARCHAR | Record ID. If this column is absent, the HAS_ID parameter must be set to 0. |
| | Other columns | INTEGER, DOUBLE, DECIMAL(p, s), VARCHAR, or NVARCHAR | Feature data and dependent variable. |

### Parameter Table

Mandatory Parameters

The following parameters are mandatory and must be given a value.

| Name | Data Type | Description | Dependency |
| --- | --- | --- | --- |
| RESAMPLING_METHOD | NVARCHAR | Specifies resampling method for model evaluation or parameter selection.<br>• cv<br>• cv_sha<br>• cv_hyperband<br>• stratified_cv<br>• stratified_cv_sha<br>• stratified_cv_hyperband<br>• bootstrap<br>• bootstrap_sha<br>• bootstrap_hyperband<br>• stratified_bootstrap<br>• stratified_bootstrap_sha<br>• stratified_bootstrap_hyperband | Options like stratified_* can only be applied to classification algorithms. Options like *_sha and *_hyperband can only be used in parameter selection, can not be used in model evaluation. |
| EVALUATION_METRIC | NVARCHAR | Specifies the evaluation metric for model evaluation or parameter selection.<br>• ACCURACY<br>• F1_SCORE<br>• RMSE | ACCURACY and F1_SCORE are only available while performing classification and RMSE is only available while performing regression. |

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| THREAD_RATIO | DOUBLE | 0.0 | Specifies the ratio of total number of threads that can be used by this function. The range of this parameter is from 0 to 1, where 0 means only using one thread, and 1 means using all currently available threads. Values outside this range are ignored and this function heuristically determines the number of threads to use. | |
| HAS_ID | INTEGER | 1 | Specifies if the first column is the ID column. | |
| ATTRIBUTE_NUM | INTEGER | Number of all attributes | Number of attributes. | |
| FUNCTIONALITY | INTEGER | 0 | Specifies the prediction type:<br>• 0: Classification<br>• 1: Regression | |
| DEPENDENT_VARIABLE | NVARCHAR | No default value | Specifies the dependent variable by name. If no column is specified, the first column other than ID column (if exists) is considered to be the dependent variable. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| CATEGORICAL_VARIABLE | NVARCHAR | No default value | Indicates whether or not a column data is actually corresponding to a category variable even the data type of this column is INTEGER.<br><br>By default, VARCHAR or NVARCHAR is category variable, and INTEGER or DOUBLE is continuous variable. | |
| STRING_VARIABLE | NVARCHAR | No default value | Indicates a string column storing not categorical data. Levenshtein distance is used to calculate similarity between two strings. Ignored if it is not a string column. | Not valid if DISTANCE_LEVEL is set to 6 (COSINE). |
| VARIABLE_WEIGHT | NVARCHAR for variable name and DOUBLE for weight value | No default value | Specifies the weight of a variable participating in distance calculation. The value must be greater or equal to 0. Defaults to 1 for variables not specified. | |
| K_NEAREST_NEIGHBOURS | INTEGER | 1 | Number of nearest neighbors (K). | |
| VOTING_TYPE | INTEGER | 1 | Voting type:<br>• 0: Majority voting<br>• 1: Distance-weighted voting | Only available while performing classification. |
| AGGREGATE_TYPE | INTEGER | 1 | Aggregate type:<br>• 0: Average<br>• 1: Distance-weighted average | Only available while performing regression. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| DISTANCE_LEVEL | INTEGER | 2 | Computes the distance between the train data and the test data point.<br>• 1: Manhattan distance<br>• 2: Euclidean distance<br>• 3: Minkowski distance<br>• 4: Chebyshev distance<br>• 6: Cosine distance | |
| MINKOWSKI_POWER | DOUBLE | 3.0 | When you use the Minkowski distance, this parameter controls the value of power. | Only valid when `DISTANCE_LEVEL` is 3 or value 3 is one of the candidate values of DISTANCE_LEVEL. |
| CATEGORY_WEIGHTS | DOUBLE | 0.707 | Represents the weight of category attributes. The value must be greater or equal to 0. | |
| METHOD | INTEGER | 0 | Searching method.<br>• 0: Brute force searching<br>• 1: KD-tree searching | |
| FOLD_NUM | INTEGER | No default value | Specifies the fold number for the cross validation method. The value should be greater than 1. | Mandatory and valid only when RESAMPLING_METHOD is set to cv or stratified_cv. |
| REPEAT_TIMES | INTEGER | 1 | Specifies the number of repeat times for resampling. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| PARAM_SEARCH_STRATEGY | NVARCHAR | No default value | Specifies the parameter search method:<br>• grid<br>• random | When *_sha is specified in RESAMPLING_METHOD, PARAM_SEARCH_STRATEGY is mandatory. When *_hyperband is specified in RESAMPLING_METHOD, PARAM_SEARCH_STRATEGY is set to random whatever other value is provided. Otherwise if this parameter is not set, only model evaluation is activated. |
| RANDOM_SEARCH_TIMES | INTEGER | No default value | Specifies the number of times to randomly select candidate parameters for selection. The value should be greater or equal to 1. | Mandatory and valid when PARAM_SEARCH_STRATEGY is set to random. |
| REDUCTION_RATE | DOUBLE | 3.0 | Specifies reduction rate in SHA or Hyperband method. | Only valid when RESAMPLING_METHOD is specified as *_sha or *_hyberband. |
| MIN_RESOURCE_RATE | DOUBLE | 0.0 | Specifies the minimum resource rate that should be used in SHA or Hyperband iteration. | Only valid when RESAMPLING_METHOD is specified as *_sha or *_hyperband. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| AGGRESSIVE_ELIMINATION | INTEGER | 0 | Specifies whether to apply aggressive elimination while using SHA method. Aggressive elimination happens when the data size and parameters size to be searched does not match and there are still bunch of parameters to be searched while data size reaches its upper limits. If aggressive elimination is applied, lower bound of limit of data size will be used multiple times first to reduce number of parameters.<br>• 0: does not apply aggressive elimination<br>• 1: applies aggressive elimination | Only valid when RESAMPLING_METHOD is specified as *_sha. |
| SEED | INTEGER | 0 | Specifies the seed for random generation. Use system time when 0 is specified. | |
| TIMEOUT | INTEGER | 0 | Specifies maximum running time for model evaluation or parameter selection, in seconds. No timeout when 0 is specified. | |
| PROGRESS_INDICATOR_ID | NVARCHAR | No default value | Sets an ID of progress indicator for model evaluation or parameter selection. No progress indicator is active if no value is provided. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| DIS-TANCE_LEVEL_VAL-UES | NVARCHAR | No default value | Specifies values of DISTANCE_LEVEL to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| MINKOWSKI_POWER_VALUES | NVARCHAR | No default value | Specifies values of MINKOWSKI_POWER to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified and value 3 is one of the candidate values of DISTANCE_LEVEL. |
| MINKOWSKI_POWER_RANGE | NVARCHAR | No default value | Specifies the range of MINKOWSKI_POWER to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified and value 3 is one of the candidate values of DISTANCE_LEVEL. |
| CATE-GORY_WEIGHTS_VAL-UES | NVARCHAR | No default value | Specifies values of parameter CATE-GORY_WEIGHTS to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| CATE-GORY_WEIGHTS_RANGE | NVARCHAR | No default value | Specifies range of parameter CATE-GORY_WEIGHTS to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| K_NEAREST_NEIGH-BOURS _VALUES | NVARCHAR | No default value | Specifies val-ues of K_NEAR-EST_NEIGHBOURS to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| K_NEAREST_NEIGH-BOURS _RANGE | NVARCHAR | No default value | Specifies the range of K_NEAR-EST_NEIGHBOURS to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| VOTING_TYPE _VAL-UES | NVARCHAR | No default value | Specifies values of VOTING_TYPE to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified and performing classifica-tion. |
| AGGRE-GATE_TYPE_VALUES | NVARCHAR | | Specifies values of parameter AGGRE-GATE_TYPE to be se-lected. | Only valid when PARAM_SEARCH_STRATEGY is specified and performing regression. |

### Output Tables

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| STATISTICS | 1st column | VARCHAR or NVARCHAR | STAT_NAME | Statistic names. |
| | 2nd column | VARCHAR or NVARCHAR | STAT_VALUE | Statistic values. |
| OPTIMAL_PARAM | 1st column | NVARCHAR | PARAM_NAME | Optimal parameters selected. |
| | 2nd column | INTEGER | INT_VALUE | |
| | 3rd column | DOUBLE | DOUBLE_VALUE | |
| | 4th column | NVARCHAR | STRING_VALUE | |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

**Model Evaluation Example:**

```
set schema DM_PAL;
drop table PAL_KNN_CV_DATA_TAB;
create column table PAL_KNN_CV_DATA_TAB(
    "ID" integer,
    "LABEL" integer,
    "DATA1" double,
    "DATA2" double
);
insert into PAL_KNN_CV_DATA_TAB values (0, 2, 1, 1);
insert into PAL_KNN_CV_DATA_TAB values (1, 3, 10, 10);
insert into PAL_KNN_CV_DATA_TAB values (2, 3, 10, 11);
insert into PAL_KNN_CV_DATA_TAB values (3, 3, 10, 10);
insert into PAL_KNN_CV_DATA_TAB values (4, 1, 100, 100);
insert into PAL_KNN_CV_DATA_TAB values (5, 1, 100, 101);
insert into PAL_KNN_CV_DATA_TAB values (6, 1, 100, 99);
insert into PAL_KNN_CV_DATA_TAB values (7, 1, 99, 99);
insert into PAL_KNN_CV_DATA_TAB values (8, 1, 99, 100);
insert into PAL_KNN_CV_DATA_TAB values (9, 1, 100, 100);
drop table #PAL_PARAMETER_TAB;
create local temporary column table #PAL_PARAMETER_TAB(
    "PARAM_NAME" nvarchar(256),
    "INT_VALUE" integer,
    "DOUBLE_VALUE" double,
    "STRING_VALUE" nvarchar(1000)
);
-- set KNN parameter
insert into #PAL_PARAMETER_TAB values ('THREAD_RATIO', null, 1, null);
insert into #PAL_PARAMETER_TAB values ('METHOD', 1, null, null);
insert into #PAL_PARAMETER_TAB values ('K_NEAREST_NEIGHBOURS', 3, null, null);
insert into #PAL_PARAMETER_TAB values ('AGGREGATE_TYPE', 1, null, null);
insert into #PAL_PARAMETER_TAB values ('DISTANCE_LEVEL', 2, null, null);
insert into #PAL_PARAMETER_TAB values ('FUNCTIONALITY', 1, null, null);
-- specify model evaluation settings
```

```
insert into #PAL_PARAMETER_TAB values ('RESAMPLING_METHOD', null, null, 'cv');
insert into #PAL_PARAMETER_TAB values ('FOLD_NUM', 5, null, null);
insert into #PAL_PARAMETER_TAB values ('EVALUATION_METRIC', null, null, 'RMSE');
insert into #PAL_PARAMETER_TAB values ('REPEAT_TIMES', 2, null, null);
insert into #PAL_PARAMETER_TAB values ('SEED', 1, null, null);
insert into #PAL_PARAMETER_TAB values ('PROGRESS_INDICATOR_ID', null, null,
'TEST');
call _SYS_AFL.PAL_KNN_CV(PAL_KNN_CV_DATA_TAB, "#PAL_PARAMETER_TAB", ?, ?);
```

**Expected Results**

STATISTIC:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | timeout | FALSE |
| 2 | TEST_1_RMSE | 0.643375, 0, 0.0253377, 0, 0 |
| 3 | TEST_2_RMSE | 0, 0, 0.707106, 0, 0.0253377 |
| 4 | TEST_RMSE.... | 0.140115 |
| 5 | TEST_RMSE.... | 0.0798769 |
| 6 | EVAL_RESU... | {"candidates":[{"TEST_RMSE":[[0.643375966... |

**Parameter Selection Example:**

```
set schema DM_PAL;
drop table PAL_KNN_CV_DATA_TAB;
create column table PAL_KNN_CV_DATA_TAB(
    "ID" INTEGER,
    "LABEL" INTEGER,
    "DATA1" double,
    "DATA2" double
);
insert into PAL_KNN_CV_DATA_TAB values (0, 2, 1, 1);
insert into PAL_KNN_CV_DATA_TAB values (1, 3, 10, 10);
insert into PAL_KNN_CV_DATA_TAB values (2, 3, 10, 11);
insert into PAL_KNN_CV_DATA_TAB values (3, 3, 10, 10);
insert into PAL_KNN_CV_DATA_TAB values (4, 1, 100, 100);
insert into PAL_KNN_CV_DATA_TAB values (5, 1, 100, 101);
insert into PAL_KNN_CV_DATA_TAB values (6, 1, 100, 99);
insert into PAL_KNN_CV_DATA_TAB values (7, 1, 99, 99);
insert into PAL_KNN_CV_DATA_TAB values (8, 1, 99, 100);
insert into PAL_KNN_CV_DATA_TAB values (9, 1, 100, 100);
drop table #PAL_PARAMETER_TAB;
create local temporary column table #PAL_PARAMETER_TAB(
    "PARAM_NAME" nvarchar(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" double,
    "STRING_VALUE" nvarchar(1000)
);
-- set KNN parameter
insert into #PAL_PARAMETER_TAB values ('THREAD_RATIO', null, 1, null);
insert into #PAL_PARAMETER_TAB values ('METHOD', 1, null, null);
-- specify parameter selection settings
insert into #PAL_PARAMETER_TAB values ('RESAMPLING_METHOD', null, null,
'stratified_bootstrap');
insert into #PAL_PARAMETER_TAB values ('EVALUATION_METRIC', null, null,
'ACCURACY');
insert into #PAL_PARAMETER_TAB values ('REPEAT_TIMES', 2, null, null);
```

```
insert into #PAL_PARAMETER_TAB values ('PARAM_SEARCH_STRATEGY', null, null,
'grid');
insert into #PAL_PARAMETER_TAB values ('SEED', 1, null, null);
insert into #PAL_PARAMETER_TAB values ('PROGRESS_INDICATOR_ID', null, null,
'TEST');
insert into #PAL_PARAMETER_TAB values ('K_NEAREST_NEIGHBOURS_VALUES', null,
null, '{1, 2, 3, 10}');
insert into #PAL_PARAMETER_TAB values ('VOTING_TYPE_VALUES', null, null, '{0,
1}');
insert into #PAL_PARAMETER_TAB values ('DISTANCE_LEVEL_VALUES', null, null, '{1,
2, 3, 4, 6}');
insert into #PAL_PARAMETER_TAB values ('MINKOWSKI_POWER_RANGE', null, null,
'[1.0, 0.5, 5.0]');
call _SYS_AFL.PAL_KNN_CV(PAL_KNN_CV_DATA_TAB, "#PAL_PARAMETER_TAB", ?, ?);
```

### Expected Results

STATISTIC:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | timeout | FALSE |
| 2 | TEST_1_ACCURACY | 1 |
| 3 | TEST_2_ACCURACY | 1 |
| 4 | TEST_ACCURACY.... | 1 |
| 5 | TEST_ACCURACY.... | 0 |
| 6 | EVAL_RESULTS_1 | {"candidates":[{"TEST_ACCURACY":[[1.0],[1.0]],"TEST_ACC... |
| 7 | EVAL_RESULTS_2 | 3333333,"TEST_ACCURACY.VAR":0.0022222222222222... |
| 8 | EVAL_RESULTS_3 | BOURS=3;DISTANCE_LEVEL=2;"},{"TEST_ACCURACY":[[1.0... |
| 9 | EVAL_RESULTS_4 | :[[1.0],[1.0]],"TEST_ACCURACY.MEAN":1.0,"TEST_ACCURA... |
| 10 | EVAL_RESULTS_5 | ACY":[[0.60],[0.6666666666666666]],"TEST_ACCURACY.M... |
| 11 | EVAL_RESULTS_6 | 333333333,"TEST_ACCURACY.VAR":0.002222222222222... |
| 12 | EVAL_RESULTS_7 | "TEST_ACCURACY":[[1.0],[1.0]],"TEST_ACCURACY.MEAN":... |
| 13 | EVAL_RESULTS_8 | E=1;K_NEAREST_NEIGHBOURS=1;DISTANCE_LEVEL=3;MI... |
| 14 | EVAL_RESULTS_9 | ],"TEST_ACCURACY.MEAN":1.0,"TEST_ACCURACY.VAR":0.... |
| 15 | EVAL_RESULTS_10 | ACY":[[1.0],[1.0]],"TEST_ACCURACY.MEAN":1.0,"TEST_ACC... |
| 16 | EVAL_RESULTS_11 | REST_NEIGHBOURS=2;DISTANCE_LEVEL=3;MINKOWSKI_... |
| 17 | EVAL_RESULTS_12 | ACCURACY.MEAN":1.0,"TEST_ACCURACY.VAR":0.0,"para... |
| 18 | EVAL_RESULTS_13 | ],[0.6666666666666666]],"TEST_ACCURACY.MEAN":0.633... |
| 19 | EVAL_RESULTS_14 | HBOURS=2;DISTANCE_LEVEL=3;MINKOWSKI_POWER=3.... |
| 20 | EVAL_RESULTS_15 | AN":1.0,"TEST_ACCURACY.VAR":0.0,"parameters":"VOTIN... |
| 21 | EVAL_RESULTS_16 | SKI_POWER=4;"},{"TEST_ACCURACY":[[1.0],[1.0]],"TEST_AC... |
| 22 | EVAL_RESULTS_17 | DISTANCE_LEVEL=3;MINKOWSKI_POWER=4.5;"},{"TEST_A... |
| 23 | EVAL_RESULTS_18 | ST_ACCURACY.VAR":0.0,"parameters":"VOTING_TYPE=0;K... |
| 24 | EVAL_RESULTS_19 | 5;"}],"version":"1.0"} |

OPTIMAL PARAMETERS:

| | PARAM_NAME | INT_VALUE | DOUBLE_VALUE | STRING_VALUE |
|---|---|---|---|---|
| 1 | VOTING_TYPE | 0 | ? | ? |
| 2 | K_NEAREST_NEIGHBOURS | 1 | ? | ? |
| 3 | DISTANCE_LEVEL | 1 | ? | ? |

## 5.2.6 Linear Discriminant Analysis

The implementation of linear discriminant analysis (LDA) in PAL includes three procedures: PAL_LINEAR_DISCRIMINANT_ANALYSIS, PAL_LINEAR_DISCRIMINANT_ANALYSIS_CLASSIFY and PAL_LINEAR_DISCRIMINANT_ANALYSIS_PROJECT.

The main procedure is PAL_LINEAR_DISCRIMINANT_ANALYSIS. It performs LDA of a given dataset X with label Y and returns:

- A classifier which can be used in PAL_LINEAR_DISCRIMINANT_ANALYSIS_CLASSIFY to classify further unlabeled data.
- A projection model which can be used in PAL_LINEAR_DISCRIMINANT_ANALYSIS_PROJECT to reduce the dimension of dataset X by projection. The projected data can be used for visualization or further classification.
- Empirical prior of each class and some other basic information.

Suppose that you are given an N×D (dataset) matrix X with an N×1 (label) vector Y, each row $x^{(i)}$ of X is a D-dimensional sample belonging to class $y_i$ and the total number of classes is C. Linear discriminant analysis (LDA) assumes that the samples within each class k obey normal distribution with different means $\mu_k$ but same covariance matrix $\Sigma$:

$P(x|y=k) \sim N(\mu_k, \Sigma)$,

i.e. $$P(x|y=k)=\frac{1}{(2\pi)^{D/2}|\Sigma|^{1/2}}\exp\left\{-\frac{1}{2}\left(x-\mu_k\right)^T\Sigma^{-1}\left(x-\mu_k\right)\right\}$$

Under this modeling assumption, you can fit the model parameters $\mu_1, \ldots, \mu_C$ and $\Sigma$ by estimating the training dataset using, for example, empirical estimation, maximum likelihood estimation, or maximum posteriori estimation.

In PAL_LINEAR_DISCRIMINANT_ANALYSIS, empirical estimation is used. The sample mean of each class is computed as:

$$\hat{\mu}_k=\frac{\sum_{i=1}^{N} M_{ik}x^{(i)}}{\sum_{i=1}^{N} M_{ik}},$$

where M is an N×C class membership matrix, $M_{ik}=1$ if $y_i=k$, otherwise $M_{ik}=0$.

Then the sample covariance is computed by first subtracting the sample mean of each class from the samples of that class, then taking the unbiased empirical covariance matrix of the result:

$$\hat{\Sigma}=\frac{\sum_{i=1}^{N}\sum_{k=1}^{C} M_{ik}\left(x^{(i)}-\hat{\mu}_k\right)\left(x^{(i)}-\hat{\mu}_k\right)^T}{N-C}.$$

After fitting the model, LDA classifies an unlabeled x by the following procedure:

Compute the posterior probability of class k for unlabeled sample x as:

$$P(y=k|x)=\frac{P(y=k)P(x|y=k)}{P(x)} \sim P(y=k)P(x|y=k),$$

Where $P(y=k)$ is the prior, $P(x)=\sum_{k=1}^{C} P(y=k)P(x|y=k)$.

Then classify x to class $\hat{y}$ such that $\hat{y}=\mathrm{argmax}_k P(y=k|x)$.

Define:

$$\delta_k(x) = \log\left(P(y=k|x)\right)$$

$$= \log\left(P(y=k)\right) + \log\left(P(x|y=k)\right)$$
$$= \log\left(P(y=k)\right) - \frac{1}{2}\log(|\hat{\Sigma}|) - \frac{1}{2}\left(x-\hat{\mu}_k\right)^T\hat{\Sigma}^{-1}\left(x-\hat{\mu}_k\right)+c,$$

Where c is a constant. It is sufficient to find $\hat{k}$ such that $\delta_{\hat{k}}(x)$ is the largest. Due to equivalence covariance matrix assumption of LDA, the second order term of x and some other terms are all the same in each $\delta_k(x)$ and can be eliminated, which leaves a linear score function of x:

$$\tilde{\delta}_k(x) = \hat{\mu}_k^T\hat{\Sigma}^{-1}x - \frac{1}{2}\hat{\mu}_k^T\hat{\Sigma}^{-1}\hat{\mu}_k + \log\left(P(y=k)\right).$$

As in PAL_LINEAR_DISCRIMINANT_ANALYSIS, the coefficients of each score function are computed and stored, and then these coefficients are used in PAL_LINEAR_DISCRIMINANT_ANALYSIS_CLASSIFY to make classification.

With mild additional computation (mainly a singular value decomposition of a `C×D` matrix), PAL_LINEAR_DISCRIMINANT_ANALYSIS can also return an optimal projection matrix V which can be used to reduce the dimension of dataset X in PAL_LINEAR_DISCRIMINANT_ANALYSIS_PROJECT by project X to the subspace span by the columns of V as $\hat{X}=V^TX$, followed by a centering.

The projection matrix V is optimal in the sense that it maximize the "separation" `S(V)` between classes after the data is projected:

$V = \mathrm{argmax}_V S(V)$,

where `S(V)` is defined as the ratio of the covariance between the classes to the covariance within the classes:

$$S(V) = \frac{\sigma^2_{between}}{\sigma^2_{within}} = \frac{\det(V^T\Sigma_B V)}{\det(V^T\Sigma_W V)},$$

where

$$\Sigma_B = \frac{\sum_{k=1}^{C} N_k\left(\hat{\mu}_k-\hat{\mu}\right)\left(\hat{\mu}_k-\hat{\mu}\right)^T}{N-C}, \quad \Sigma_W = \hat{\Sigma},$$

$$N_i = \sum_{i=1}^{N} M_{ik}, \quad \hat{\mu} = \frac{\sum_{i=1}^{N} x^{(i)}}{N}.$$

It should be mentioned that the reduced dimension K should be less than `(C-1)` because $\Sigma_B$ is at most rank `(C-1)`.

In some cases, the number of features of each sample exceeds the number of samples in each class, this will cause the covariance estimates $\hat{\Sigma}$ to be rank deficient, and so cannot be inverted. PAL_LINEAR_DISCRIMINANT_ANALYSIS gives some options to handle these circumstances:

- Use regularized covariance $\hat{\Sigma}_\gamma = (1-\gamma)\hat{\Sigma} + \gamma\,\mathrm{diag}(\hat{\Sigma})$
- Use diagonal of $\hat{\Sigma}$
- Use pseudo inverse when inverting $\hat{\Sigma}$

By default, the algorithm uses regularized $\hat{\Sigma}_\gamma$ and chooses the minimum γ such that $\hat{\Sigma}_\gamma$ is invertible.

In case of missing data (null in dataset X), PAL_LINEAR_DISCRIMINANT_ANALYSIS used the mean of the corresponding class to fill in the missing value.

In case of missing label (null in label Y), PAL_LINEAR_DISCRIMINANT_ANALYSIS simply ignores the sample and does not use it to fit.

In case of missing data, PAL_LINEAR_DISCRIMINANT_ANALYSIS_CLASSIFY and PAL_LINEAR_DISCRIMINANT_ANALYSIS_PROJECT use zero to fill in the missing value.

In case of missing ID, PAL_LINEAR_DISCRIMINANT_ANALYSIS_CLASSIFY and PAL_LINEAR_DISCRIMINANT_ANALYSIS_PROJECT simply ignore the corresponding data and do not process it.

> **Note**
>
> Due to numerical reason, the signs of discriminants in result projection matrix might be different from those in the versions prior to SAP HANA Platform 2.0 SPS 02. However, this will not impact the correctness of the result since different signs of a discriminant are considered equivalent in linear discriminant analysis.

## Prerequisites

- The input data columns are of INTEGER or DOUBLE data type.
- For each class label, there should be at least two rows with different feature values.

## _SYS_AFL.PAL_LINEAR_DISCRIMINANT_ANALYSIS

This function performs LDA of a given dataset and returns:

- A classifier which can be used in PAL_LINEAR_DISCRIMINANT_ANALYSIS_CLASSIFY to classify further unlabeled data.
- A projection model which can be used in PAL_LINEAR_DISCRIMINANT_ANALYSIS_PROJECT to reduce the dimension of dataset.
- Empirical prior of each class and some other basic information.

**Overview of All Tables**

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<BASIC_INFORMATION table>` | OUT |
| 4 | `<PRIORS table>` | OUT |
| 5 | `<CLASSIFIER table>` | OUT |
| 6 | `<PROJECTION_INFORMATION table>` | OUT |

| Position | Table Name | Parameter Type |
|---|---|---|
| 7 | `<PROJECTION_MODEL table>` | OUT |

### Input Table

| Table | Column | Reference for Output Table | Column Data Type | Description |
|---|---|---|---|---|
| DATA | Feature columns | FEATURES | INTEGER or DOUBLE | Attribute data |
| | Last column | CLASS | INTEGER, VARCHAR, or NVARCHAR | Class |

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| REGULARIZATION_TYPE | INTEGER | 0 | Specifies regularization type:<br>• 0: Uses regularized $\widehat{\Sigma}$<br>• 1: Uses diagonal of $\widehat{\Sigma}$<br>• 2: Uses pseudo inverse when inverting $\widehat{\Sigma}$ | |
| REGULARIZATION_AMOUNT | DOUBLE | Minimum $\gamma$ such that $\widehat{\Sigma}_\gamma$ is invertible | Parameter $\gamma$ in [0,1] used to regularize $\widehat{\Sigma}$:<br>$\widehat{\Sigma}_\gamma=(1-\gamma)\widehat{\Sigma}+\gamma\mathrm{diag}(\widehat{\Sigma})$ | Only valid when REGULARIZATION_TYPE is 0. |
| DO_PROJECTION | INTEGER | 1 | Indicates whether to compute the projection model or not. If this is set to 0, the projection model and projection info output tables will be empty. | |

### Output Tables

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| BASIC_INFORMATION | 1st column | NVARCHAR(100) | NAME | Property name |
| | 2nd column | DOUBLE | VALUE | Property value |
| PRIORS | 1st column | CLASS (in input table) data type | CLASS (in input table) type | Class |
| | 2nd column | DOUBLE | PRIOR | Empirical prior of the class |
| CLASSIFIER | 1st column | CLASS (in input table) data type | CLASS (in input table) type | Class |
| | Coefficient columns | DOUBLE | COEFF_ + FEATURES (in input table) column name | Coefficients of the class' linear score function |
| | Last column | DOUBLE | INTERCEPT | Intercept of the class's linear score function |
| PROJECTION_INFORMATION | 1st column | NVARCHAR (100) | DISCRIMINANT_ID | Discriminant ID |
| | 2nd column | DOUBLE | SD | Standard deviations of the discriminant |
| | 3rd column | DOUBLE | VAR_PROP | Variance proportion to the total variance explained by each discriminant |
| | 4th column | DOUBLE | CUM_VAR_PROP | Cumulative variance proportion to the total variance explained by each discriminant |
| PROJECTION_MODEL | 1st column | NVARCHAR (100) | NAME | Discriminant ID and overall mean |
| | Projection model columns | DOUBLE | FEATURES (in input table) column name | Projection matrix and overall mean |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;

- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_LINEAR_DISCRIMINANT_ANALYSIS_DATA_TBL;
CREATE COLUMN TABLE PAL_LINEAR_DISCRIMINANT_ANALYSIS_DATA_TBL ("X1" DOUBLE, "X2"
DOUBLE, "X3" DOUBLE, "X4" DOUBLE, "CLASS" NVARCHAR(100));
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_DATA_TBL VALUES (5.1, 3.5, 1.4,
0.2, 'Iris-setosa');
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_DATA_TBL VALUES (4.9, 3.0, 1.4,
0.2, 'Iris-setosa');
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_DATA_TBL VALUES (4.7, 3.2, 1.3,
0.2, 'Iris-setosa');
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_DATA_TBL VALUES (4.6, 3.1, 1.5,
0.2, 'Iris-setosa');
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_DATA_TBL VALUES (5.0, 3.6, 1.4,
0.2, 'Iris-setosa');
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_DATA_TBL VALUES (5.4, 3.9, 1.7,
0.4, 'Iris-setosa');
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_DATA_TBL VALUES (4.6, 3.4, 1.4,
0.3, 'Iris-setosa');
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_DATA_TBL VALUES (5.0, 3.4, 1.5,
0.2, 'Iris-setosa');
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_DATA_TBL VALUES (4.4, 2.9, 1.4,
0.2, 'Iris-setosa');
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_DATA_TBL VALUES (4.9, 3.1, 1.5,
0.1, 'Iris-setosa');
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_DATA_TBL VALUES (7.0, 3.2, 4.7,
1.4, 'Iris-versicolor');
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_DATA_TBL VALUES (6.4, 3.2, 4.5,
1.5, 'Iris-versicolor');
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_DATA_TBL VALUES (6.9, 3.1, 4.9,
1.5, 'Iris-versicolor');
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_DATA_TBL VALUES (5.5, 2.3, 4.0,
1.3, 'Iris-versicolor');
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_DATA_TBL VALUES (6.5, 2.8, 4.6,
1.5, 'Iris-versicolor');
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_DATA_TBL VALUES (5.7, 2.8, 4.5,
1.3, 'Iris-versicolor');
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_DATA_TBL VALUES (6.3, 3.3, 4.7,
1.6, 'Iris-versicolor');
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_DATA_TBL VALUES (4.9, 2.4, 3.3,
1.0, 'Iris-versicolor');
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_DATA_TBL VALUES (6.6, 2.9, 4.6,
1.3, 'Iris-versicolor');
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_DATA_TBL VALUES (5.2, 2.7, 3.9,
1.4, 'Iris-versicolor');
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_DATA_TBL VALUES (6.3, 3.3, 6.0,
2.5, 'Iris-virginica');
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_DATA_TBL VALUES (5.8, 2.7, 5.1,
1.9, 'Iris-virginica');
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_DATA_TBL VALUES (7.1, 3.0, 5.9,
2.1, 'Iris-virginica');
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_DATA_TBL VALUES (6.3, 2.9, 5.6,
1.8, 'Iris-virginica');
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_DATA_TBL VALUES (6.5, 3.0, 5.8,
2.2, 'Iris-virginica');
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_DATA_TBL VALUES (7.6, 3.0, 6.6,
2.1, 'Iris-virginica');
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_DATA_TBL VALUES (4.9, 2.5, 4.5,
1.7, 'Iris-virginica');
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_DATA_TBL VALUES (7.3, 2.9, 6.3,
1.8, 'Iris-virginica');
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_DATA_TBL VALUES (6.7, 2.5, 5.8,
1.8, 'Iris-virginica');
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_DATA_TBL VALUES (7.2, 3.6, 6.1,
2.5, 'Iris-virginica');
```

```
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL ("PARAM_NAME"
VARCHAR(256), "INT_VALUE" INTEGER, "DOUBLE_VALUE" DOUBLE, "STRING_VALUE"
VARCHAR(1000));
INSERT INTO #PAL_PARAMETER_TBL VALUES ('DO_PROJECTION', 1, NULL, NULL);
DROP TABLE PAL_LDA_PRIORS_TBL;
DROP TABLE PAL_LDA_CLASSIFIER_TBL;
DROP TABLE PAL_LDA_PROJECTION_MODEL_TBL;
CREATE COLUMN TABLE PAL_LDA_PRIORS_TBL ("CLASS" NVARCHAR(100), "PRIOR" DOUBLE);
CREATE COLUMN TABLE PAL_LDA_CLASSIFIER_TBL ("CLASS" NVARCHAR(100), "COEFF_X1"
DOUBLE, "COEFF_X2" DOUBLE, "COEFF_X3" DOUBLE, "COEFF_X4" DOUBLE, "INTERCEPT"
DOUBLE);
CREATE COLUMN TABLE PAL_LDA_PROJECTION_MODEL_TBL ("NAME" NVARCHAR(100), "X1"
DOUBLE, "X2" DOUBLE, "X3" DOUBLE, "X4" DOUBLE);
CALL _SYS_AFL.PAL_LINEAR_DISCRIMINANT_ANALYSIS
(PAL_LINEAR_DISCRIMINANT_ANALYSIS_DATA_TBL, #PAL_PARAMETER_TBL, ?,
PAL_LDA_PRIORS_TBL, PAL_LDA_CLASSIFIER_TBL, ?, PAL_LDA_PROJECTION_MODEL_TBL)
WITH OVERVIEW;
```

**Expected Result**

BASIC_INFORMATION:

| | NAME | VALUE |
|---|---|---|
| 1 | CLASS_NUM | 3 |
| 2 | SAMPLE_NUM | 30 |
| 3 | SAMPLE_USED | 30 |

PRIORS:

| | CLASS | PRIOR |
|---|---|---|
| 1 | Iris-setosa | 0.3333333333333333 |
| 2 | Iris-versicolor | 0.3333333333333333 |
| 3 | Iris-virginica | 0.3333333333333333 |

CLASSIFIER:

| | CLASS | COEFF_X1 | COEFF_X2 | COEFF_X3 | COEFF_X4 | INTERCEPT |
|---|---|---|---|---|---|---|
| 1 | Iris-setosa | 23.907390632287107 | 51.75400132383591 | -34.641901675376346 | -49.06340729470452 | -113.2354779103408 |
| 2 | Iris-versicolor | 0.5110341225331645 | 15.652078170461506 | 15.209568000877203 | -4.8610177851685705 | -53.89819005848877 |
| 3 | Iris-virginica | -14.72963611162715 | 4.981955283916806 | 42.51148604627835 | 12.315007016931364 | -94.14356404144554 |

PROJECTION_INFORMATION:

| | DISCRIMINANT_ID | SD | VAR_PROP | CUM_VAR_PROP |
|---|---|---|---|---|
| 1 | DISCRIMINANT_1 | 76.5323138785947 | 0.9972664360287256 | 0.9972664360287256 |
| 2 | DISCRIMINANT_2 | 0.20977942132484018 | 0.0027335639712744208 | 1 |

PROJECTION_MODEL:

| | NAME | X1 | X2 | X3 | X4 |
|---|---|---|---|---|---|
| 1 | DISCRIMINANT_1 | -1.9079777058779852 | -2.3995159207489465 | 3.8461535534662143 | 3.112216100808173 |
| 2 | DISCRIMINANT_2 | -3.0467941065859625 | 4.575495547710883 | 2.7572709979338272 | -2.6330368172217273 |
| 3 | OVERALL_MEAN | 5.843333333333334 | 3.04 | 3.863333333333333 | 1.213333333333333 |

## _SYS_AFL.PAL_LINEAR_DISCRIMINANT_ANALYSIS_CLASSIFY

This function uses the classifier returned by PAL_LINEAR_DISCRIMINANT_ANALYSIS to classify unlabeled data.

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PRIORS table>` | IN |
| 3 | `<CLASSIFIER table>` | IN |
| 4 | `<PARAMETER table>` | IN |
| 5 | `<RESULT table>` | OUT |

### Input Tables

| Table | Column | Reference for Output Table | Data Type | Description |
|---|---|---|---|---|
| DATA | 1st column | ID | INTEGER | ID |
| | Feature columns | FEATURES | INTEGER or DOUBLE | Attribute data |
| PRIORS | 1st column | | INTEGER, VARCHAR, or NVARCHAR | Class |
| | 2nd column | | DOUBLE | Prior of the class.<br>You can use specified priors or the empirical priors output by PAL_LINEAR_DISCRIMINANT_ANALYSIS. |
| CLASSIFIER | 1st column | CLASS | INTEGER, VARCHAR, or NVARCHAR | Class |

| Table | Column | Reference for Output Table | Data Type | Description |
|---|---|---|---|---|
| | Other columns | | DOUBLE | Coefficients of the class' linear score function.<br>The number of coefficient columns should be equal to the number of variables in the input data. |
| | Last column | | DOUBLE | Intercept of the class' linear score function |

### Parameter Table

Mandatory Parameters

None.

Optional Parameter

The following parameter is optional. If it is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| VERBOSE_OUTPUT | INTEGER | 0 | • 0: Only outputs score of the predicted class.<br>• 1: Outputs score of all classes. |

### Output Table

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| RESULT | 1st column | ID (in input table) data type | ID (in input table) column name | ID |
| | 2nd column | CLASS (in input table) data type | CLASS (in input table) column name | Class |
| | 3rd column | DOUBLE | SCORE | Score of the class |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;

- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_LINEAR_DISCRIMINANT_ANALYSIS_CLASSIFY_DATA_TBL;
CREATE COLUMN TABLE PAL_LINEAR_DISCRIMINANT_ANALYSIS_CLASSIFY_DATA_TBL ("ID"
INTEGER, "X1" DOUBLE, "X2" DOUBLE, "X3" DOUBLE, "X4" DOUBLE);
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_CLASSIFY_DATA_TBL VALUES (1, 5.1,
3.5, 1.4, 0.2);
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_CLASSIFY_DATA_TBL VALUES (2, 4.9,
3.0, 1.4, 0.2);
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_CLASSIFY_DATA_TBL VALUES (3, 4.7,
3.2, 1.3, 0.2);
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_CLASSIFY_DATA_TBL VALUES (4, 4.6,
3.1, 1.5, 0.2);
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_CLASSIFY_DATA_TBL VALUES (5, 5.0,
3.6, 1.4, 0.2);
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_CLASSIFY_DATA_TBL VALUES (6, 5.4,
3.9, 1.7, 0.4);
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_CLASSIFY_DATA_TBL VALUES (7, 4.6,
3.4, 1.4, 0.3);
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_CLASSIFY_DATA_TBL VALUES (8, 5.0,
3.4, 1.5, 0.2);
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_CLASSIFY_DATA_TBL VALUES (9, 4.4,
2.9, 1.4, 0.2);
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_CLASSIFY_DATA_TBL VALUES (10, 4.9,
3.1, 1.5, 0.1);
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_CLASSIFY_DATA_TBL VALUES (11, 7.0,
3.2, 4.7, 1.4);
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_CLASSIFY_DATA_TBL VALUES (12, 6.4,
3.2, 4.5, 1.5);
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_CLASSIFY_DATA_TBL VALUES (13, 6.9,
3.1, 4.9, 1.5);
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_CLASSIFY_DATA_TBL VALUES (14, 5.5,
2.3, 4.0, 1.3);
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_CLASSIFY_DATA_TBL VALUES (15, 6.5,
2.8, 4.6, 1.5);
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_CLASSIFY_DATA_TBL VALUES (16, 5.7,
2.8, 4.5, 1.3);
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_CLASSIFY_DATA_TBL VALUES (17, 6.3,
3.3, 4.7, 1.6);
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_CLASSIFY_DATA_TBL VALUES (18, 4.9,
2.4, 3.3, 1.0);
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_CLASSIFY_DATA_TBL VALUES (19, 6.6,
2.9, 4.6, 1.3);
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_CLASSIFY_DATA_TBL VALUES (20, 5.2,
2.7, 3.9, 1.4);
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_CLASSIFY_DATA_TBL VALUES (21, 6.3,
3.3, 6.0, 2.5);
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_CLASSIFY_DATA_TBL VALUES (22, 5.8,
2.7, 5.1, 1.9);
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_CLASSIFY_DATA_TBL VALUES (23, 7.1,
3.0, 5.9, 2.1);
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_CLASSIFY_DATA_TBL VALUES (24, 6.3,
2.9, 5.6, 1.8);
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_CLASSIFY_DATA_TBL VALUES (25, 6.5,
3.0, 5.8, 2.2);
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_CLASSIFY_DATA_TBL VALUES (26, 7.6,
3.0, 6.6, 2.1);
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_CLASSIFY_DATA_TBL VALUES (27, 4.9,
2.5, 4.5, 1.7);
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_CLASSIFY_DATA_TBL VALUES (28, 7.3,
2.9, 6.3, 1.8);
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_CLASSIFY_DATA_TBL VALUES (29, 6.7,
2.5, 5.8, 1.8);
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_CLASSIFY_DATA_TBL VALUES (30, 7.2,
3.6, 6.1, 2.5);
```

```
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL ("PARAM_NAME"
VARCHAR(256), "INT_VALUE" INTEGER, "DOUBLE_VALUE" DOUBLE, "STRING_VALUE"
VARCHAR(1000));
INSERT INTO #PAL_PARAMETER_TBL VALUES ('VERBOSE_OUTPUT', 0, NULL, NULL);
CALL _SYS_AFL.PAL_LINEAR_DISCRIMINANT_ANALYSIS_CLASSIFY
(PAL_LINEAR_DISCRIMINANT_ANALYSIS_CLASSIFY_DATA_TBL, PAL_LDA_PRIORS_TBL,
PAL_LDA_CLASSIFIER_TBL, #PAL_PARAMETER_TBL, ?);
```

**Expected Result**

RESULT:

| | ID | CLASS | SCORE |
|---|---|---|---|
| 1 | 1 | Iris-setosa | 130.4212628546132 |
| 2 | 2 | Iris-setosa | 99.76278406623784 |
| 3 | 3 | Iris-setosa | 108.79629637208... |
| 4 | 4 | Iris-setosa | 94.30177684139765 |
| 5 | 5 | Iris-setosa | 133.20592392376... |
| 6 | 6 | Iris-setosa | 138.08982861227... |
| 7 | 7 | Iris-setosa | 108.38582667661... |
| 8 | 8 | Iris-setosa | 119.39093349146... |
| 9 | 9 | Iris-setosa | 82.63368861771069 |
| 10 | 10 | Iris-setosa | 106.38033476055... |
| 11 | 11 | Iris-versicolor | 63.34663136093894 |
| 12 | 12 | Iris-versicolor | 59.51199550872676 |
| 13 | 13 | Iris-versicolor | 64.28613195329808 |
| 14 | 14 | Iris-versicolor | 38.33261400162666 |
| 15 | 15 | Iris-versicolor | 54.823224452883... |
| 16 | 16 | Iris-versicolor | 53.86564391180264 |
| 17 | 17 | Iris-versicolor | 63.58191173517818 |
| 18 | 18 | Iris-versicolor | 30.40280908008943 |
| 19 | 19 | Iris-versicolor | 57.41173923921635 |
| 20 | 20 | Iris-versicolor | 42.43307645444673 |

## _SYS_AFL.PAL_LINEAR_DISCRIMINANT_ANALYSIS_PROJECT

This function uses the projection model returned by PAL_LINEAR_DISCRIMINANT_ANALYSIS to reduce the dimension of data.

**Overview of All Tables**

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PROJECTION_MODEL table>` | IN |

| Position | Table Name | Parameter Type |
|---|---|---|
| 3 | `<PARAMETER table>` | IN |
| 4 | `<RESULT table>` | OUT |

### Input Tables

| Table | Column | Reference for Output Table | Data Type | Description | Constraint |
|---|---|---|---|---|---|
| DATA | 1st column | ID | INTEGER | ID | |
| | Feature columns | FEATURES | INTEGER or DOUBLE | Attribute data | |
| PROJECTION_MODEL | 1st column | | NVARCHAR | Discriminant ID and overall mean | |
| | Projection model columns | | DOUBLE | Projection matrix and overall mean | The number of columns that store the coefficients should be equal to the number of variables in the input data plus one. |

### Parameter Table

Mandatory Parameters

None.

Optional Parameter

The following parameter is optional. If it is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| DISCRIMINANT_NUMBER | INTEGER | Number of discriminants in projection model | Specifies the number of discriminants to be retained.<br>Value range: 0 < DISCRIMINANT_NUMBER ≤ number of discriminants in projection model. |

### Output Table

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| RESULT | 1st column | ID (in input table) data type | ID (in input table) column name | ID |
| | Projected data column | DOUBLE | DISCRIMINANT_ + i (i sequences from 1 to the number of columns in FEATURES) | Data after projection |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_LINEAR_DISCRIMINANT_ANALYSIS_PROJECT_DATA_TBL;
CREATE COLUMN TABLE PAL_LINEAR_DISCRIMINANT_ANALYSIS_PROJECT_DATA_TBL ("ID"
INTEGER, "X1" DOUBLE, "X2" DOUBLE, "X3" DOUBLE, "X4" DOUBLE);
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_PROJECT_DATA_TBL VALUES (1, 5.1,
3.5, 1.4, 0.2);
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_PROJECT_DATA_TBL VALUES (2, 4.9,
3.0, 1.4, 0.2);
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_PROJECT_DATA_TBL VALUES (3, 4.7,
3.2, 1.3, 0.2);
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_PROJECT_DATA_TBL VALUES (4, 4.6,
3.1, 1.5, 0.2);
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_PROJECT_DATA_TBL VALUES (5, 5.0,
3.6, 1.4, 0.2);
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_PROJECT_DATA_TBL VALUES (6, 5.4,
3.9, 1.7, 0.4);
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_PROJECT_DATA_TBL VALUES (7, 4.6,
3.4, 1.4, 0.3);
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_PROJECT_DATA_TBL VALUES (8, 5.0,
3.4, 1.5, 0.2);
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_PROJECT_DATA_TBL VALUES (9, 4.4,
2.9, 1.4, 0.2);
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_PROJECT_DATA_TBL VALUES (10, 4.9,
3.1, 1.5, 0.1);
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_PROJECT_DATA_TBL VALUES (11, 7.0,
3.2, 4.7, 1.4);
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_PROJECT_DATA_TBL VALUES (12, 6.4,
3.2, 4.5, 1.5);
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_PROJECT_DATA_TBL VALUES (13, 6.9,
3.1, 4.9, 1.5);
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_PROJECT_DATA_TBL VALUES (14, 5.5,
2.3, 4.0, 1.3);
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_PROJECT_DATA_TBL VALUES (15, 6.5,
2.8, 4.6, 1.5);
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_PROJECT_DATA_TBL VALUES (16, 5.7,
2.8, 4.5, 1.3);
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_PROJECT_DATA_TBL VALUES (17, 6.3,
3.3, 4.7, 1.6);
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_PROJECT_DATA_TBL VALUES (18, 4.9,
2.4, 3.3, 1.0);
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_PROJECT_DATA_TBL VALUES (19, 6.6,
2.9, 4.6, 1.3);
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_PROJECT_DATA_TBL VALUES (20, 5.2,
2.7, 3.9, 1.4);
```

```
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_PROJECT_DATA_TBL VALUES (21, 6.3,
3.3, 6.0, 2.5);
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_PROJECT_DATA_TBL VALUES (22, 5.8,
2.7, 5.1, 1.9);
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_PROJECT_DATA_TBL VALUES (23, 7.1,
3.0, 5.9, 2.1);
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_PROJECT_DATA_TBL VALUES (24, 6.3,
2.9, 5.6, 1.8);
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_PROJECT_DATA_TBL VALUES (25, 6.5,
3.0, 5.8, 2.2);
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_PROJECT_DATA_TBL VALUES (26, 7.6,
3.0, 6.6, 2.1);
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_PROJECT_DATA_TBL VALUES (27, 4.9,
2.5, 4.5, 1.7);
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_PROJECT_DATA_TBL VALUES (28, 7.3,
2.9, 6.3, 1.8);
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_PROJECT_DATA_TBL VALUES (29, 6.7,
2.5, 5.8, 1.8);
INSERT INTO PAL_LINEAR_DISCRIMINANT_ANALYSIS_PROJECT_DATA_TBL VALUES (30, 7.2,
3.6, 6.1, 2.5);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL ("PARAM_NAME"
VARCHAR(256), "INT_VALUE" INTEGER, "DOUBLE_VALUE" DOUBLE, "STRING_VALUE"
VARCHAR(1000));
INSERT INTO #PAL_PARAMETER_TBL VALUES ('DISCRIMINANT_NUMBER', 2, NULL, NULL);
CALL _SYS_AFL.PAL_LINEAR_DISCRIMINANT_ANALYSIS_PROJECT
(PAL_LINEAR_DISCRIMINANT_ANALYSIS_PROJECT_DATA_TBL,
PAL_LDA_PROJECTION_MODEL_TBL, #PAL_PARAMETER_TBL, ?);
```

**Expected Result**

RESULT:

| | ID | DISCRIMINANT_1 | DISCRIMINANT_2 | DISCRIMINANT_3 | DISCRIMINANT_4 |
|---|---|---|---|---|---|
| 1 | 1 | -12.313584464365933 | 0.2455779877169264 | ? | ? |
| 2 | 2 | -10.732230962815864 | -1.432810964821324 | ? | ? |
| 3 | 3 | -11.215153961136679 | -0.18408013375533... | ? | ? |
| 4 | 4 | -10.015173887780742 | 0.21450392171893... | ? | ? |
| 5 | 5 | -12.362738285853032 | 1.0078069531466118 | ? | ? |
| 6 | 6 | -12.069494858227408 | 1.4623119107612936 | ? | ? |
| 7 | 7 | -10.80842240927123 | 1.0481218045166472 | ? | ? |
| 8 | 8 | -11.49821974635662 | 0.3684349433978178 | ? | ? |
| 9 | 9 | -9.538290517801977 | -0.36696346629943... | ? | ? |
| 10 | 10 | -10.898788809624955 | -0.43623062853467... | ? | ? |
| 11 | 11 | 1.2080787180988937 | -0.97662936659410... | ? | ? |
| 12 | 12 | 1.894856241013259 | 0.03668921604852... | ? | ? |
| 13 | 13 | 2.719280401535646 | -0.84134899284200... | ? | ? |
| 14 | 14 | 3.2260805080827577 | -2.191170216486469 | ? | ? |
| 15 | 15 | 3.048480194071659 | -1.82246131390103... | ? | ? |
| 16 | 16 | 3.567803783265793 | 0.8658542350186957 | ? | ? |
| 17 | 17 | 2.926154740300224 | 1.0870686993428063 | ? | ? |
| 18 | 18 | 0.5049432218658461 | -1.04572285115096... | ? | ? |
| 19 | 19 | 1.9952876112473295 | -1.142983806344199 | ? | ? |
| 20 | 20 | 2.765273706280767 | 0.01403545305812... | ? | ? |

> **Note**
>
> The wrapper procedure generation approach to call PAL functions is still supported. For more information, refer to the older versions (prior to SAP HANA Platform 2.0 SPS 02) of this document.

## 5.2.7 Logistic Regression (with Elastic Net Regularization)

Logistic regression models the relationship between a dichotomous dependent variable (also known as explained variable) and one or more continuous or categorical independent variables (also known as explanatory variables). It models the log odds of the dependent variable as a linear combination of the independent variables.

This function can only handle binary-class classification problems. For multiple-class classification problems, refer to *Multi-Class Logistic Regression*.

Considering a training data set with `n` samples and `m` explanatory variables, the logistic regression model is made by:

$$h_{(\theta_0,\theta)}(x) = 1/(1 + \exp(-(\theta_0+\theta^T x)))$$

Where $\theta_0$ is the intercept, $\theta$ represents coefficients $\theta_1, \ldots, \theta_m$ and $\theta^T x = \theta_1 x_1 + \ldots + \theta_m x_m$

Assuming that there are only two class labels, {0,1}, you can get the below formula:

$$P(y = 1 \mid x; (\theta_0,\theta)) = h_{(\theta_0,\theta)}(x)$$

$$P(y = 0 \mid x; (\theta_0,\theta)) = 1 - h_{(\theta_0,\theta)}(x)$$

And combine them into:

$$P(y \mid x;(\theta_0,\theta)) = h_{(\theta_0,\theta)}(x)^y (1 - h_{(\theta_0,\theta)}(x))^{1-y}$$

Here $\theta_0, \theta_1, \ldots, \theta_m$ can be obtained through the Maximum Likelihood Estimation (MLE) method.

The likelihood function is:

$$L(\theta_0,\theta) = \prod_{i=1}^{n} p(y^{(i)} \mid x^{(i)}; (\theta_0,\theta))$$

The log-likelihood function is:

$$\begin{aligned}
&\ell(\theta_0,\theta) \\
&= \ln\left(\prod_{i=1}^{n} p(y^{(i)} \mid x^{(i)}; (\theta_0,\theta))\right) \\
&= \ln\left(\prod_{i=1}^{n} \left(h_{(\theta_0,\theta)}(x^{(i)})\right)^{y^{(i)}} \left(1-h_{(\theta_0,\theta)}(x^{(i)})\right)^{1-y^{(i)}}\right) \\
&= \sum_{i=1}^{n} \left(y^{(i)} \ln\left(h_{(\theta_0,\theta)}\left(x^{(i)}\right)\right) + (1-y^{(i)}) \ln\left(1-h_{(\theta_0,\theta)}(x^{(i)})\right)\right)
\end{aligned}$$

Newton method, L-BFGS and Cyclical Coordinate Descend (primarily for elastic net penalized object function) are provided to minimize objective function which is opposite in sign with log likelihood function. For fast convergence, Newton method and L-BFGS are preferred.

Elastic net regularization seeks to find coefficients which can minimize:

$$\min_{(\theta_0,\theta)\in R^{m+1}} \left(-\frac{l(\theta_0,\theta)}{n} + \lambda * P_\alpha(\theta)\right)$$

Where $P_\alpha(\theta) = (1-\alpha)\frac{1}{2}\|\theta\|_{l_2}^2 + \alpha\|\theta\|_{l_1}$.

Here $\alpha \in [0,1]$ and $\lambda \geq 0$. If $\alpha=0$, we have the ridge regularization; if $\alpha=1$, we have the LASSO regularization.

Parameter selection process is time-consuming when the training data set is large or when the parameter candidate search space is large. Current algorithm supports successive-halving and hyperband algorithm to accelerate this search process. Both algorithms exploit the intermediate evaluation results from training results for each hyper-parameter candidate and eliminate some of them accordingly. So by early-stopping the training process for some hyper-parameters and eliminating them from candidate sets the whole parameter selection process is boosted. The evaluation metric for final optimal hyper-parameter selected by these two algorithms may be sub-optimal comparing to non-early-stopping algorithm, but theoretically their gaps can be arbitrarily small when enough resource budget is allocated to training process of each candidate. Currently logistic regression only supports iteration number or epoch number (for SGD algorithm) as resource. With more iteration steps allocated per candidate or smaller reduction rate applied will improve the accuracy of optimal result, but in the meantime consumes longer parameter selection time; Otherwise it will boost run-time performance with the sacrifice of accuracy.

For details of successive-halving and hyperband algorithm, refer to Successive Halving and Hyperband for Model Evaluation and Parameter Selection [page 2240].

Procedure _SYS_AFL.PAL_LOGISTIC_REGRESSION_PREDICT is used to predict the labels for the testing data.

## Prerequisites

- Data is numeric, or categorical.
- Given m independent variables, there must be at least m+1 records available for analysis.

## _SYS_AFL.PAL_LOGISTIC_REGRESSION

This is a logistic regression function.

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<RESULT table>` | OUT |
| 4 | `<PMMLMODEL table>` | OUT |
| 5 | `<STATISTIC table>` | OUT |
| 6 | `<OPTIMAL PARAM table>` | OUT |

### Input Table

| Table | Column | Column Data Type | Description |
|---|---|---|---|
| DATA | Feature columns | INTEGER, DOUBLE, DECIMAL(p, s), VARCHAR, or NVARCHAR | Independent variables |
| | Type column | INTEGER, VARCHAR, or NVARCHAR | Dependent Variables<br>**Note**: For INTEGER dependent variables, it must take value 0 or 1. |

### Parameter Table

Mandatory Parameters

The following parameters are mandatory and must be given a value.

| Name | Data Type | Description | Dependency |
|---|---|---|---|
| CLASS_MAP0 | VARCHAR | Specifies the dependent variable value which will be mapped to 0. | Only valid when the column type of dependent variables is VARCHAR or NVARCHAR. |
| CLASS_MAP1 | VARCHAR | Specifies the dependent variable value which will be mapped to 1. | Only valid when the column type of dependent variables is VARCHAR or NVARCHAR. |

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| METHOD | INTEGER | -1 | • -1: Auto-selection, based on given data and parameters.<br>• 0: Newton iteration method.<br>• 2: Cyclical coordinate descent method to fit elastic net regularized logistic regression.<br>• 3: LBFGS method (recommended when having many independent variables).<br>• 4: Stochastic gradient descent method (recommended when dealing with very large dataset).<br>• 6: Proximal gradient descent method to fit elastic net regularized logistic regression. | |
| THREAD_RATIO | DOUBLE | 1.0 | Specifies the ratio of total number of threads that can be used by this function. The value range is from 0 to 1, where 0 means only using one thread, and 1 means using at most all the currently available threads. Values outside the range will be ignored and this function heuristically determines the number of threads to use. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| MAX_PASS_NUMBER | INTEGER | 1 | The maximum number of passes over the data. | Only valid when `METHOD` is 4. |
| SGD_BATCH_NUMBER | INTEGER | 1 | The batch number of Stochastic gradient descent. | Only valid when `METHOD` is 4. |
| PRECOMPUTE | INTEGER | 1 | Whether to compute Gram matrix:<br>• 0: No;<br>• 1: Yes | Only valid when `METHOD` is 2. |
| ENET_LAMBDA | DOUBLE | 0.0 | Penalized weight. The value should be equal to or greater than 0. | Only valid when `METHOD` is 0, 2, 3, or 6.<br>**Note**<br>If you want to solve Ridge penalty problem for `METHOD` 0, set ENET_ALPHA to 0.0. |
| ENET_ALPHA | DOUBLE | 1.0 | The elastic net mixing parameter. The value range is between 0 and 1 inclusively.<br>• 0: Ridge penalty<br>• 1: LASSO penalty | Only valid when `METHOD` is 0, 2, 3, or 6.<br>**Note**<br>`METHOD` 0 only solves Ridge penalty problem. |
| HANDLE_MISSING | INTEGER | 1 | Whether to handle missing values:<br>• 0: No<br>• 1: Yes | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| MAX_ITERATION | INTEGER | 100, 1000, or 100000 | When METHOD is 0 or 3, the convex optimizer may return suboptimal results after the maximum number of iterations. When METHOD is 2, if convergence is not reached after the maximum number of passes over training data, an error will be generated. | When `METHOD` is 2, `MAX_ITERATION` defaults to 100000; when `METHOD` is 6, it defaults to 1000; otherwise it defaults to 100. |
| HAS_ID | INTEGER | 0 | Specifies whether the input data table has an ID column.<br>• 0: No<br>• 1: Yes | |
| STANDARDIZE | INTEGER | 1 | Controls whether to standardize the data to have zero mean and unit variance.<br>• 0: No<br>• 1: Yes | |
| EXIT_THRESHOLD | DOUBLE | 1.0e-6 or 1.0e-7 | Convergence threshold for exiting iterations. | When METHOD is 2, EXIT_THRESHOLD defaults to 1.0E-7; Otherwise it defaults to 1.0E-6. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| EPSILON | DOUBLE | 1.0e-5 or 1.0e-6 | The parameter determines the accuracy with which the solution is to be found.<br>When `METHOD` is 3, the condition is: ||g|| < EPSILON * max {1, ||x||}, where g is gradient of objective function, x is solve of current iteration, and ||·|| denotes the L2 norm;<br>When `METHOD` is 0, the condition is: ||x- x'|| < EPSILON * sqrt(n), where x is the solve of current iteration, x' is the previous iteration, and n is the number of features. | When `METHOD` is 0, `EPSILON` defaults to 1.0e-6. When `METHOD` is 3, `EPSILON` defaults to 1.0e-5. |
| LBFGS_M | INTEGER | 6 | Number of past updates needed to be kept. | Only available when `METHOD` is 3. |
| STAT_INF | INTEGER | 0 | • 0: Does not proceed statistical inference<br>• 1: Proceeds statistical inference | |
| PMML_EXPORT | INTEGER | 0 | • 0: Does not export logistic regression model in PMML.<br>• 1: Exports logistic regression model in PMML in single row.<br>• 2: Exports logistic regression model in PMML in several rows, and the minimum length of each row is 5000 characters. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| SELECTED_FEATURES | VARCHAR | No default value | A string to specify the features that will be processed. The pattern is "$X_1,...,X_n$", where $X_i$ is the corresponding column name in the data table. If this parameter is not specified, all the features will be processed. | Only used when you need to indicate the needed features. |
| DEPENDENT_VARIABLE | VARCHAR | No default value | Column name in the data table used as dependent variable. | Only used when you need to indicate the dependence. |
| CATEGORICAL_VARIABLE | VARCHAR | No default value | Column name in the data table used as category variable.<br><br>By default, STRING is category variable, and INTEGER or DOUBLE is continuous variable. | |

## Model Evaluation and Parameter Selection

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| RESAMPLING_METHOD | NVARCHAR | No default value | Specifies the resampling method for model evaluation or parameter selection.<br>• cv<br>• cv_sha<br>• cv_hyperband<br>• bootstrap<br>• bootstrap_sha<br>• bootstrap_hyperband<br>• stratified_cv<br>• stratified_cv_sha<br>• stratified_cv_hyperband<br>• stratified_bootstrap<br>• stratified_bootstrap_sha<br>• stratified_bootstrap_hyperband<br>If no value is specified for this parameter, neither model evaluation nor parameter selection is activated. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| RESOURCE | NVARCHAR | No default value | Resource type used in successive-halving and hyperband algorithm for parameter selection.<br>• MAX_ITERATION<br>• MAX_PASS_NUMBER | Mandatory and valid only when RESAMPLING_METHOD parameter has values with sha or hyperband suffix. The value MAX_PASS_NUMBER is only valid when actual used method is Stochastic Gradient Descent; MAX_ITERATION is used for other methods. When the user set METHOD as -1 (Auto Selection), the value to use depends on actual used methods; For current implementation users should always choose MAX_ITERATION in this case. |
| MAX_RESOURCE | INTEGER | No default value | Maximum allowed resource budget for single hyper-parameter candidate. Valid value must be larger than 0. | Mandatory and valid only when RESOURCE parameter is set. |
| MIN_RESOURCE_RATE | DOUBLE | 0 | The rate between minimum allowed resource and maximum allowed resource. Valid value is between [0, 1). | Mandatory and valid only when RESOURCE parameter is set. |
| REDUCTION_RATE | DOUBLE | 3.0 | This parameter specifies the reduction rate of available size of hyper-parameter candidates. For each round, the available parameter candidate size will be divided by value of this parameter. Thus valid value for this parameter must be larger than 1.0. | Valid only when RESOURCE parameter is set. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| AGGRESSIVE_ELIMINATION | INTEGER | 0 | Boolean value indicates whether to perform aggressive elimination behavior for successive-halving algorithm. When set to 1, it will eliminate more parameter candidates than expected, which is defined by REDUCTION_RATE parameter; This can accelerate the run-time performance but may result in sub-optimal hyper-parameter candidate.<br>• 0: Does not perform aggressive elimination;<br>• 1: Performs aggressive elimiination | Valid only when RESAMPLING_METHOD parameter has value with suffix sha. |
| PS_VERBOSE | INTEGER | 1 | Boolean value indicates whether to output optimal hyper-parameter and all involved hyper-parameter candidates related evaluation statistics in Statistics output table.<br>• 0: Does not output evaluation statistics<br>• 1: Outputs evaluation statistics | Valid only when RESAMPLING_METHOD is set. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| EVALUATION_METRIC | NVARCHAR | No default value | Specifies the evaluation metric for model evaluation or parameter selection.<br>• ACCURACY<br>• F1_SCORE<br>• AUC<br>• NLL | |
| FOLD_NUM | INTEGER | No default value | Specifies the fold number for the cross validation method. | Mandatory and valid only when RESAMPLING_METHOD is set to cv, stratified_cv, cv_sha, cv_hyperband, stratified_cv_sha, or stratified_cv_hyperband. |
| REPEAT_TIMES | INTEGER | 1 | Specifies the number of repeat times for resampling. | |
| PARAM_SEARCH_STRATEGY | NVARCHAR | No default value | Specifies the method to activate parameter selection.<br>• grid<br>• random | This parameter is only valid when RESAMPLING_METHOD does not have value with suffix hyperband; Otherwise, this parameter is always considered to have value random. |
| RANDOM_SEARCH_TIMES | INTEGER | No default value | Specifies the number of times to randomly select candidate parameters for selection. | Mandatory and valid when PARAM_SEARCH_STRATEGY is set to random. |
| SEED | INTEGER | 0 | Specifies the seed for random generation. Use system time when 0 is specified. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| TIMEOUT | INTEGER | 0 | Specifies maximum running time for model evaluation or parameter selection, in seconds. No timeout when 0 is specified. | |
| PROGRESS_INDICATOR_ID | NVARCHAR | No default value | Sets an ID of progress indicator for model evaluation or parameter selection. No progress indicator is active if no value is provided. | |
| ENET_LAMBDA_VALUES | NVARCHAR | No default value | Specifies values of ENET_LAMBDA to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| ENET_LAMBDA_RANGE | NVARCHAR | No default value | Specifies the range of ENET_LAMBDA to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| ENET_ALPHA_VALUES | NVARCHAR | No default value | Specifies values of ENET_ALPHA to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| ENET_ALPHA_RANGE | NVARCHAR | No default value | Specifies the range of ENET_ALPHA to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |

### Output Tables

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| RESULT | 1st column | NVARCHAR(1000) | VARIABLE_NAME | Variable name |
| | 2nd column | DOUBLE | COEFFICIENT | Value of the coefficients |
| | 3rd column | DOUBLE | Z_SCORE | (When STAT_INF = 1) Zscore of each coefficient parameter |
| | 4th column | DOUBLE | P_VALUE | (When STAT_INF = 1) Pvalue of each coefficient parameter |
| PMML | 1st column | INTEGER | ROW_INDEX | ID |

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| | 2nd column | NVARCHAR(5000) | MODEL_CONTENT | Logistic regression model in PMML format |
| STATISTIC | 1st column | NVARCHAR(256) | STAT_NAME | AIC, obj, log-likelihood, iter.<br>Note that iter means the number of iterations. |
| | 2nd column | NVARCHAR(1000) | STAT_VALUE | AIC value, objective function value, log-likelihood value, iteration value |
| OPTIMAL_PARAM | 1st column | NVARCHAR(256) | PARAM_NAME | |
| | 2nd column | INTEGER | INT_VALUE | |
| | 3rd column | DOUBLE | DOUBLE_VALUE | |
| | 4th column | NVARCHAR(1000) | STRING_VALUE | |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

**Example 1: Fitting logistic regression model without penalties**

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_LOGISTICR_DATA_TBL;
CREATE COLUMN TABLE PAL_LOGISTICR_DATA_TBL ("V1" VARCHAR (50),"V2" DOUBLE,"V3"
INTEGER,"CATEGORY" INTEGER);
INSERT INTO PAL_LOGISTICR_DATA_TBL VALUES ('B',2.62,0,1);
INSERT INTO PAL_LOGISTICR_DATA_TBL VALUES ('B',2.875,0,1);
INSERT INTO PAL_LOGISTICR_DATA_TBL VALUES ('A',2.32,1,1);
INSERT INTO PAL_LOGISTICR_DATA_TBL VALUES ('A',3.215,2,0);
INSERT INTO PAL_LOGISTICR_DATA_TBL VALUES ('B',3.44,3,0);
INSERT INTO PAL_LOGISTICR_DATA_TBL VALUES ('B',3.46,0,0);
INSERT INTO PAL_LOGISTICR_DATA_TBL VALUES ('A',3.57,1,0);
INSERT INTO PAL_LOGISTICR_DATA_TBL VALUES ('B',3.19,2,0);
INSERT INTO PAL_LOGISTICR_DATA_TBL VALUES ('A',3.15,3,0);
INSERT INTO PAL_LOGISTICR_DATA_TBL VALUES ('B',3.44,0,0);
INSERT INTO PAL_LOGISTICR_DATA_TBL VALUES ('B',3.44,1,0);
INSERT INTO PAL_LOGISTICR_DATA_TBL VALUES ('A',4.07,3,0);
INSERT INTO PAL_LOGISTICR_DATA_TBL VALUES ('A',3.73,1,0);
INSERT INTO PAL_LOGISTICR_DATA_TBL VALUES ('B',3.78,2,0);
INSERT INTO PAL_LOGISTICR_DATA_TBL VALUES ('B',5.25,2,0);
INSERT INTO PAL_LOGISTICR_DATA_TBL VALUES ('A',5.424,3,0);
INSERT INTO PAL_LOGISTICR_DATA_TBL VALUES ('A',5.345,0,0);
INSERT INTO PAL_LOGISTICR_DATA_TBL VALUES ('B',2.2,1,1);
INSERT INTO PAL_LOGISTICR_DATA_TBL VALUES ('B',1.615,2,1);
```

```
INSERT INTO PAL_LOGISTICR_DATA_TBL VALUES ('A',1.835,0,1);
INSERT INTO PAL_LOGISTICR_DATA_TBL VALUES ('B',2.465,3,0);
INSERT INTO PAL_LOGISTICR_DATA_TBL VALUES ('A',3.52,1,0);
INSERT INTO PAL_LOGISTICR_DATA_TBL VALUES ('A',3.435,0,0);
INSERT INTO PAL_LOGISTICR_DATA_TBL VALUES ('B',3.84,2,0);
INSERT INTO PAL_LOGISTICR_DATA_TBL VALUES ('B',3.845,3,0);
INSERT INTO PAL_LOGISTICR_DATA_TBL VALUES ('A',1.935,1,1);
INSERT INTO PAL_LOGISTICR_DATA_TBL VALUES ('B',2.14,0,1);
INSERT INTO PAL_LOGISTICR_DATA_TBL VALUES ('B',1.513,1,1);
INSERT INTO PAL_LOGISTICR_DATA_TBL VALUES ('A',3.17,3,1);
INSERT INTO PAL_LOGISTICR_DATA_TBL VALUES ('B',2.77,0,1);
INSERT INTO PAL_LOGISTICR_DATA_TBL VALUES ('B',3.57,0,1);
INSERT INTO PAL_LOGISTICR_DATA_TBL VALUES ('A',2.78,3,1);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('METHOD',3,NULL,NULL);
-- INSERT INTO #PAL_PARAMETER_TBL VALUES ('EXIT_THRESHOLD',NULL,0.000001,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('THREAD_RATIO',NULL,0.1,NULL);
-- INSERT INTO #PAL_PARAMETER_TBL VALUES ('MAX_ITERATION',1000,NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('CATEGORICAL_VARIABLE',NULL,NULL,'V3');
-- parameter SGD_BATCH_NUMBER is valid when METHOD is 4.
--INSERT INTO #PAL_PARAMETER_TBL VALUES ('SGD_BATCH_NUMBER',3,NULL,NULL);
--INSERT INTO #PAL_PARAMETER_TBL VALUES ('HANDLE_MISSING', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('STAT_INF',0,NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PMML_EXPORT',0,NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('HAS_ID',0,NULL,NULL);
DROP TABLE PAL_LOGISTICR_RESULT_TBL;
CREATE COLUMN TABLE PAL_LOGISTICR_RESULT_TBL ("VARIABLE_NAME"
NVARCHAR(1000),"COEFFICIENT" DOUBLE,"Z_SCORE" DOUBLE,"P_VALUE" DOUBLE);
DROP TABLE PAL_LOGISTICR_PMMLMODEL_TBL;
CREATE COLUMN TABLE PAL_LOGISTICR_PMMLMODEL_TBL ("ROW_INDEX"
INTEGER,"MODEL_CONTENT" NVARCHAR(5000));
CALL _SYS_AFL.PAL_LOGISTIC_REGRESSION
(PAL_LOGISTICR_DATA_TBL,"#PAL_PARAMETER_TBL",PAL_LOGISTICR_RESULT_TBL,PAL_LOGISTI
CR_PMMLMODEL_TBL,?,?) WITH OVERVIEW;
```

**Expected Result**

RESULT:

| | VARIABLE_NAME | COEFFICIENT | Z_SCORE | P_VALUE |
|---|---|---|---|---|
| 1 | __PAL_INTERCEPT__ | 17.04602917075882 | ? | ? |
| 2 | V1__PAL_DELIMIT__A | 0 | ? | ? |
| 3 | V1__PAL_DELIMIT__B | -1.4648882403529... | ? | ? |
| 4 | V2 | -4.8201132270187... | ? | ? |
| 5 | V3__PAL_DELIMIT__0 | 0 | ? | ? |
| 6 | V3__PAL_DELIMIT__1 | -2.7942067036962... | ? | ? |
| 7 | V3__PAL_DELIMIT__2 | -4.8088016689396... | ? | ? |
| 8 | V3__PAL_DELIMIT__3 | -2.7810517149633... | ? | ? |
| 9 | {"CONTENT":"{\"impute_... | ? | ? | ? |

PMML:

empty

STATISTIC:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | AIC | 26.832334860522344 |
| 2 | obj | 0.23175523219566166 |
| 3 | log-likelih... | -7.416167430261173 |
| 4 | iter | 15 |
| 5 | method | lbfgs |
| 6 | solution st... | Converged |

**Example 2: Fitting logistic regression model with elastic net penalties**

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_ENET_LOGISTICR_DATA_TBL;
CREATE COLUMN TABLE PAL_ENET_LOGISTICR_DATA_TBL ("V1" DOUBLE, "V2" INTEGER, "V3"
DOUBLE, "V4" INTEGER, "V5" INTEGER, "V6" VARCHAR(50), "CATEGORY" INTEGER);
INSERT INTO PAL_ENET_LOGISTICR_DATA_TBL VALUES (4,1, 0.86,0,0, 'blue', 1);
INSERT INTO PAL_ENET_LOGISTICR_DATA_TBL VALUES (8,0, 0.45,1,2, 'blue', 1);
INSERT INTO PAL_ENET_LOGISTICR_DATA_TBL VALUES (7,1, 0.99,1,2, 'yellow', 0);
INSERT INTO PAL_ENET_LOGISTICR_DATA_TBL VALUES (12,1, 0.84,2,2, 'red', 1);
INSERT INTO PAL_ENET_LOGISTICR_DATA_TBL VALUES (6,1, 0.85,2,2, 'red', 0);
INSERT INTO PAL_ENET_LOGISTICR_DATA_TBL VALUES (9,0, 0.67,3,3, 'yellow', 0);
INSERT INTO PAL_ENET_LOGISTICR_DATA_TBL VALUES (10,1, 0.91,2,2 ,'yellow', 0);
INSERT INTO PAL_ENET_LOGISTICR_DATA_TBL VALUES (14,0, 0.29, 0,0, 'red', 1);
INSERT INTO PAL_ENET_LOGISTICR_DATA_TBL VALUES (7,0, 0.88, 1,0, 'yellow', 1);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('METHOD', 2, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('CATEGORICAL_VARIABLE', NULL, NULL,
'V2');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('CATEGORICAL_VARIABLE', NULL, NULL,
'V4');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('CATEGORICAL_VARIABLE', NULL, NULL,
'V5');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PMML_EXPORT', 0, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('ENET_LAMBDA', NULL, 0.03613, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('ENET_ALPHA', NULL, 0.7, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('HAS_ID', 0, NULL, NULL);
DROP TABLE PAL_ENET_LOGISTICR_RESULT_TBL;
CREATE COLUMN TABLE PAL_ENET_LOGISTICR_RESULT_TBL ("VARIABLE_NAME"
NVARCHAR(1000), "COEFFICIENT" DOUBLE, "Z_SCORE" DOUBLE, "P_VALUE" DOUBLE);
DROP TABLE PAL_ENET_LOGISTICR_PMMLMODEL_TBL;
CREATE COLUMN TABLE PAL_ENET_LOGISTICR_PMMLMODEL_TBL ("ROW_INDEX" INTEGER,
"MODEL_CONTENT" NVARCHAR(5000));
CALL _SYS_AFL.PAL_LOGISTIC_REGRESSION (PAL_ENET_LOGISTICR_DATA_TBL,
"#PAL_PARAMETER_TBL", PAL_ENET_LOGISTICR_RESULT_TBL,
PAL_ENET_LOGISTICR_PMMLMODEL_TBL, ?, ?) WITH OVERVIEW;
```

**Expected Results**

RESULT:

| | VARIABLE_NAME | COEFFICIENT | Z_SCORE | P_VALUE |
|---|---|---|---|---|
| 1 | __PAL_INTERCEPT__ | 2.431473168946126 | ? | ? |
| 2 | V1 | 0.11462816560515546 | ? | ? |
| 3 | V2__PAL_DELIMIT__0 | 0 | ? | ? |
| 4 | V2__PAL_DELIMIT__1 | -0.199970714085047... | ? | ? |
| 5 | V3 | -1.879197175047733 | ? | ? |
| 6 | V4__PAL_DELIMIT__0 | 0 | ? | ? |
| 7 | V4__PAL_DELIMIT__1 | 0.4784814444825556 | ? | ? |
| 8 | V4__PAL_DELIMIT__2 | -0.2958230117715817 | ? | ? |
| 9 | V4__PAL_DELIMIT__3 | -0.7617839700687163 | ? | ? |
| 10 | V5__PAL_DELIMIT__0 | 0 | ? | ? |
| 11 | V5__PAL_DELIMIT__2 | -1.2568222002162774 | ? | ? |
| 12 | V5__PAL_DELIMIT__3 | -0.7614147724722 | ? | ? |
| 13 | V6__PAL_DELIMIT__b... | 0 | ? | ? |
| 14 | V6__PAL_DELIMIT__r... | 0 | ? | ? |
| 15 | V6__PAL_DELIMIT__y... | -1.518992712741706 | ? | ? |
| 16 | {"CONTENT":"{\"imp... | ? | ? | ? |
| 17 | {"CONTENT":"MPUT... | ? | ? | ? |

PMML:

empty

STATISTIC:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | AIC | 25.750907234963872 |
| 2 | obj | 0.5026056001164475 |
| 3 | log-likelih... | -2.875453617481936 |
| 4 | iter | 5 |
| 5 | method | cyclical coordinate ... |
| 6 | solution st... | Converged |

**Example 3: Model evaluation**

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_ENET_LOGISTICR_DATA_TBL;
CREATE COLUMN TABLE PAL_ENET_LOGISTICR_DATA_TBL ("V1" DOUBLE, "V2" INTEGER, "V3"
DOUBLE, "V4" INTEGER, "V5" INTEGER, "V6" VARCHAR(50), "CATEGORY" INTEGER);
INSERT INTO PAL_ENET_LOGISTICR_DATA_TBL VALUES (4,1, 0.86,0,0, 'blue', 1);
INSERT INTO PAL_ENET_LOGISTICR_DATA_TBL VALUES (8,0, 0.45,1,2, 'blue', 1);
INSERT INTO PAL_ENET_LOGISTICR_DATA_TBL VALUES (7,1, 0.99,1,2, 'yellow', 0);
```

```
INSERT INTO PAL_ENET_LOGISTICR_DATA_TBL VALUES (12,1, 0.84,2,2, 'red', 1);
INSERT INTO PAL_ENET_LOGISTICR_DATA_TBL VALUES (6,1, 0.85,2,2, 'red', 0);
INSERT INTO PAL_ENET_LOGISTICR_DATA_TBL VALUES (9,0, 0.67,3,3, 'yellow', 0);
INSERT INTO PAL_ENET_LOGISTICR_DATA_TBL VALUES (10,1, 0.91,2,2 ,'yellow', 0);
INSERT INTO PAL_ENET_LOGISTICR_DATA_TBL VALUES (14,0, 0.29, 0,0, 'red', 1);
INSERT INTO PAL_ENET_LOGISTICR_DATA_TBL VALUES (7,0, 0.88, 1,0, 'yellow', 1);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('METHOD', 6, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('CATEGORICAL_VARIABLE', NULL, NULL,
'V2');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('CATEGORICAL_VARIABLE', NULL, NULL,
'V4');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('CATEGORICAL_VARIABLE', NULL, NULL,
'V5');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PMML_EXPORT', 0, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('ENET_LAMBDA', NULL, 0.03613, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('ENET_ALPHA', NULL, 0.7, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('HAS_ID', 0, NULL, NULL);
--model evaluation parameters
INSERT INTO #PAL_PARAMETER_TBL VALUES ('RESAMPLING_METHOD', NULL, NULL, 'cv');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('EVALUATION_METRIC', NULL, NULL,
'F1_SCORE');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FOLD_NUM', 2, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('REPEAT_TIMES', 2, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SEED', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PROGRESS_INDICATOR_ID', NULL, NULL,
'TEST');
DROP TABLE PAL_ENET_LOGISTICR_RESULT_TBL;
CREATE COLUMN TABLE PAL_ENET_LOGISTICR_RESULT_TBL ("VARIABLE_NAME"
NVARCHAR(1000), "COEFFICIENT" DOUBLE, "Z_SCORE" DOUBLE, "P_VALUE" DOUBLE);
DROP TABLE PAL_ENET_LOGISTICR_PMMLMODEL_TBL;
CREATE COLUMN TABLE PAL_ENET_LOGISTICR_PMMLMODEL_TBL ("ROW_INDEX" INTEGER,
"MODEL_CONTENT" NVARCHAR(5000));
CALL _SYS_AFL.PAL_LOGISTIC_REGRESSION (PAL_ENET_LOGISTICR_DATA_TBL,
"#PAL_PARAMETER_TBL", PAL_ENET_LOGISTICR_RESULT_TBL,
PAL_ENET_LOGISTICR_PMMLMODEL_TBL, ?, ?) WITH OVERVIEW;
```

### Expected Result

STATISTICS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | timeout | FALSE |
| 2 | TEST_1_F1_SCORE | 0.583333, 0.2 |
| 3 | TEST_2_F1_SCORE | 0.4, 0.733333 |
| 4 | TEST_F1_SCORE.... | 0.479166 |
| 5 | TEST_F1_SCORE.... | 0.0532175 |
| 6 | EVAL_RESULTS_1 | {"candidat... |
| 7 | AIC | 25.7659095... |
| 8 | obj | 0.50389331... |
| 9 | log-likelihood | -2.8829547... |
| 10 | iter | 124 |
| 11 | method | proximal g... |
| 12 | solution status | Converged |

**Example 4: Parameter selection**

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_ENET_LOGISTICR_DATA_TBL;
CREATE COLUMN TABLE PAL_ENET_LOGISTICR_DATA_TBL ("V1" DOUBLE, "V2" INTEGER, "V3"
DOUBLE, "V4" INTEGER, "V5" INTEGER, "V6" VARCHAR(50), "CATEGORY" INTEGER);
INSERT INTO PAL_ENET_LOGISTICR_DATA_TBL VALUES (4,1, 0.86,0,0, 'blue', 1);
INSERT INTO PAL_ENET_LOGISTICR_DATA_TBL VALUES (8,0, 0.45,1,2, 'blue', 1);
INSERT INTO PAL_ENET_LOGISTICR_DATA_TBL VALUES (7,1, 0.99,1,2, 'yellow', 0);
INSERT INTO PAL_ENET_LOGISTICR_DATA_TBL VALUES (12,1, 0.84,2,2, 'red', 1);
INSERT INTO PAL_ENET_LOGISTICR_DATA_TBL VALUES (6,1, 0.85,2,2, 'red', 0);
INSERT INTO PAL_ENET_LOGISTICR_DATA_TBL VALUES (9,0, 0.67,3,3, 'yellow', 0);
INSERT INTO PAL_ENET_LOGISTICR_DATA_TBL VALUES (10,1, 0.91,2,2 ,'yellow', 0);
INSERT INTO PAL_ENET_LOGISTICR_DATA_TBL VALUES (14,0, 0.29, 0,0, 'red', 1);
INSERT INTO PAL_ENET_LOGISTICR_DATA_TBL VALUES (7,0, 0.88, 1,0, 'yellow', 1);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('METHOD', 6, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('CATEGORICAL_VARIABLE', NULL, NULL,
'V2');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('CATEGORICAL_VARIABLE', NULL, NULL,
'V4');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('CATEGORICAL_VARIABLE', NULL, NULL,
'V5');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PMML_EXPORT', 0, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('HAS_ID', 0, NULL, NULL);
--parameters to be selected
INSERT INTO #PAL_PARAMETER_TBL VALUES ('ENET_LAMBDA_VALUES', NULL, NULL,
'{0.01,0.02,0.03167}');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('ENET_ALPHA_VALUES', NULL, NULL,
'{0.15,0.2,0.7}');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('RESAMPLING_METHOD', NULL, NULL, 'cv');
--parameter selection parameters
```

```
INSERT INTO #PAL_PARAMETER_TBL VALUES ('EVALUATION_METRIC', NULL, NULL,
'F1_SCORE');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FOLD_NUM', 2, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('REPEAT_TIMES', 2, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SEED', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PARAM_SEARCH_STRATEGY', NULL, NULL,
'grid');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PROGRESS_INDICATOR_ID', NULL, NULL,
'TEST');
DROP TABLE PAL_ENET_LOGISTICR_RESULT_TBL;
CREATE COLUMN TABLE PAL_ENET_LOGISTICR_RESULT_TBL ("VARIABLE_NAME"
NVARCHAR(1000), "COEFFICIENT" DOUBLE, "Z_SCORE" DOUBLE, "P_VALUE" DOUBLE);
DROP TABLE PAL_ENET_LOGISTICR_PMMLMODEL_TBL;
CREATE COLUMN TABLE PAL_ENET_LOGISTICR_PMMLMODEL_TBL ("ROW_INDEX" INTEGER,
"MODEL_CONTENT" NVARCHAR(5000));
CALL _SYS_AFL.PAL_LOGISTIC_REGRESSION (PAL_ENET_LOGISTICR_DATA_TBL,
"#PAL_PARAMETER_TBL", PAL_ENET_LOGISTICR_RESULT_TBL,
PAL_ENET_LOGISTICR_PMMLMODEL_TBL, ?, ?) WITH OVERVIEW;
```

**EXPECTED RESULT**

STATISTICS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | timeout | FALSE |
| 2 | TEST_1_F1_SCORE | 0.583333, 0.2 |
| 3 | TEST_2_F1_SCORE | 0.4, 0.733333 |
| 4 | TEST_F1_SCORE.... | 0.479166 |
| 5 | TEST_F1_SCORE.... | 0.0532175 |
| 6 | EVAL_RESULTS_1 | {"candidat... |
| 7 | EVAL_RESULTS_2 | 333333333... |
| 8 | AIC | 22.1294059... |
| 9 | obj | 0.27305300... |
| 10 | log-likelihood | -1.0647029... |
| 11 | iter | 172 |
| 12 | method | proximal g... |
| 13 | solution status | Converged |

OPTIMAL PARAMETERS:

| | PARAM_NAME | INT_VALUE | DOUBLE_VALUE | STRING_VALUE |
|---|---|---|---|---|
| 1 | ENET_ALPHA | ? | 0.7 | ? |
| 2 | ENET_LAMBDA | ? | 0.01 | ? |

**Example 5: Parameter Selection with Successive-Halving**

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_ENET_LOGISTICR_DATA_TBL;
CREATE COLUMN TABLE PAL_ENET_LOGISTICR_DATA_TBL ("V1" DOUBLE, "V2" INTEGER, "V3"
DOUBLE, "V4" INTEGER, "V5" INTEGER, "V6" VARCHAR(50), "CATEGORY" INTEGER);
```

```
INSERT INTO PAL_ENET_LOGISTICR_DATA_TBL VALUES (4,1, 0.86,0,0, 'blue', 1);
INSERT INTO PAL_ENET_LOGISTICR_DATA_TBL VALUES (8,0, 0.45,1,2, 'blue', 1);
INSERT INTO PAL_ENET_LOGISTICR_DATA_TBL VALUES (7,1, 0.99,1,2, 'yellow', 0);
INSERT INTO PAL_ENET_LOGISTICR_DATA_TBL VALUES (12,1, 0.84,2,2, 'red', 1);
INSERT INTO PAL_ENET_LOGISTICR_DATA_TBL VALUES (6,1, 0.85,2,2, 'red', 0);
INSERT INTO PAL_ENET_LOGISTICR_DATA_TBL VALUES (9,0, 0.67,3,3, 'yellow', 0);
INSERT INTO PAL_ENET_LOGISTICR_DATA_TBL VALUES (10,1, 0.91,2,2 ,'yellow', 0);
INSERT INTO PAL_ENET_LOGISTICR_DATA_TBL VALUES (14,0, 0.29, 0,0, 'red', 1);
INSERT INTO PAL_ENET_LOGISTICR_DATA_TBL VALUES (7,0, 0.88, 1,0, 'yellow', 1);
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL (
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR (1000)
);
INSERT INTO PAL_PARAMETER_TBL VALUES ('METHOD', 6, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('CATEGORICAL_VARIABLE', NULL, NULL, 'V2');
INSERT INTO PAL_PARAMETER_TBL VALUES ('CATEGORICAL_VARIABLE', NULL, NULL, 'V4');
INSERT INTO PAL_PARAMETER_TBL VALUES ('CATEGORICAL_VARIABLE', NULL, NULL,
'V5');
INSERT INTO PAL_PARAMETER_TBL VALUES ('PMML_EXPORT', 0, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('HAS_ID', 0, NULL, NULL);
--parameters to be selected
INSERT INTO PAL_PARAMETER_TBL VALUES ('ENET_LAMBDA_VALUES', NULL, NULL,
'{0.01,0.02,0.03167}');
INSERT INTO PAL_PARAMETER_TBL VALUES ('ENET_ALPHA_VALUES', NULL, NULL,
'{0.15,0.2,0.7}');
INSERT INTO PAL_PARAMETER_TBL VALUES ('PARAM_SEARCH_STRATEGY', NULL, NULL,
'grid');
--parameter selection parameters
INSERT INTO PAL_PARAMETER_TBL VALUES ('EVALUATION_METRIC', NULL, NULL,
'F1_SCORE');
INSERT INTO PAL_PARAMETER_TBL VALUES ('RESAMPLING_METHOD', NULL, NULL,
'cv_sha');
INSERT INTO PAL_PARAMETER_TBL VALUES ('RESOURCE', NULL, NULL, 'MAX_ITERATION');
INSERT INTO PAL_PARAMETER_TBL VALUES ('MAX_RESOURCE', 1000, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('MIN_RESOURCE_RATE', NULL, 0.4, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('REDUCTION_RATE', NULL, 2.0, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('FOLD_NUM', 2, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('REPEAT_TIMES', 2, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('SEED', 2022, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('PROGRESS_INDICATOR_ID', NULL, NULL,
'TEST');
DROP TABLE PAL_ENET_LOGISTICR_STATISTIC_TBL;
CREATE COLUMN TABLE PAL_ENET_LOGISTICR_STATISTIC_TBL ("STAT_NAME"
NVARCHAR(256),"STAT_VALUE" NVARCHAR(1000));
DROP TABLE PAL_ENET_LOGISTICR_OPT_PARAM_TBL;
CREATE COLUMN TABLE PAL_ENET_LOGISTICR_OPT_PARAM_TBL ("PARAM_NAME"
NVARCHAR(256),"INT_VALUE" INTEGER, "DOUBLE_VALUE" DOUBLE, "STRING_VALUE"
NVARCHAR(1000));
DO BEGIN
lt_data = SELECT * FROM PAL_ENET_LOGISTICR_DATA_TBL;
lt_para = SELECT * FROM PAL_PARAMETER_TBL;
CALL _SYS_AFL.PAL_LOGISTIC_REGRESSION
(:lt_data,:lt_para,lt_result,lt_pmml_model,lt_stat,lt_opt_para);
INSERT INTO PAL_ENET_LOGISTICR_STATISTIC_TBL
SELECT * FROM :lt_stat;
INSERT INTO PAL_ENET_LOGISTICR_OPT_PARAM_TBL
SELECT * FROM :lt_opt_para;
END;
SELECT * FROM PAL_ENET_LOGISTICR_STATISTIC_TBL;
SELECT * FROM PAL_ENET_LOGISTICR_OPT_PARAM_TBL;
```

Expected Results

STATISTICS:

| | STAT_NAME | |
|---|---|---|
| 1 | timeout | FALSE |
| 2 | TEST_1_F1_SCORE | 0.285714, 0.5 |
| 3 | TEST_2_F1_SCORE | 0.583333, 0.5 |
| 4 | EVAL_RESULTS_1 | {"candidates":[{"TEST_F1_SCORE":[[[0.5833333333333333,0.50],[0.5833333333333333,0.20]],[[0.2857142857142858,0.50],[0.583333 |
| 5 | EVAL_RESULTS_2 | rs":"ENET_ALPHA=0.2;ENET_LAMBDA=0.02;"},{"TEST_F1_SCORE":[[[0.5833333333333333,0.20],[0.5833333333333333,0.50]]],"TEST |
| 6 | AIC | 22.129405953486888 |
| 7 | obj | 0.1183003307492715 |
| 8 | log-likelihood | -1.0647029767434435 |
| 9 | iter | 172 |
| 10 | method | proximal gradient descent |
| 11 | solution status | Converged |

OPTIMAL_PARAMETER:

| | PARAM_NAME | INT_VALUE | DOUBLE_VALUE | STRING_VALUE |
|---|---|---|---|---|
| 1 | ENET_ALPHA | *NULL* | 0.7 | *NULL* |
| 2 | ENET_LAMBDA | *NULL* | 0.01 | *NULL* |

# _SYS_AFL.PAL_LOGISTIC_REGRESSION_PREDICT

This function performs predication with logistic regression result.

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<COEFFICIENT table>` | IN |
| 3 | `<PARAMETER table>` | IN |
| 4 | `<RESULT table>` | OUT |

### Input Tables

| Table | Column | Reference for Output Table | Data Type | Description |
|---|---|---|---|---|
| DATA | 1st column | ID | INTEGER | Data item ID |
| | Feature columns | | INTEGER, DOUBLE, DECIMAL(p, s), VARCHAR, or NVARCHAR | Independent variables |
| COEFFICIENT | 1st column | | INTEGER, VARCHAR, or NVARCHAR | ID |

| Table | Column | Reference for Output Table | Data Type | Description |
|---|---|---|---|---|
| | 2nd column | | INTEGER, DOUBLE, VARCHAR, or NVARCHAR | COEFFICIENT or PMML model.<br>**Note**: VARCHAR, NVARCHAR, and CLOB types are only valid for PMML model. |
| | 3rd column | | DOUBLE | Only valid for non-PMML model. |
| | 4th column | | DOUBLE | Only valid for non-PMML model. |

### Parameter Table

Mandatory Parameters

The following parameters are mandatory and must be given a value.

| Name | Data Type | Description | Dependency |
|---|---|---|---|
| CLASS_MAP0 | VARCHAR | The same value as the parameter of PAL_LOGISTIC_REGRESSION. | Only valid when the column type of dependent variable is VARCHAR or NVARCHAR. |
| CLASS_MAP1 | VARCHAR | The same value as the parameter of PAL_LOGISTIC_REGRESSION. | Only valid when the column type of dependent variable is VARCHAR or NVARCHAR. |

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| THREAD_RATIO | DOUBLE | 1.0 | Specifies the ratio of total number of threads that can be used by this function. The value range is from 0 to 1, where 0 means only using 1 thread, and 1 means using at most all the currently available threads. Values outside the range will be ignored and this function heuristically determines the number of threads to use. |

### Output Table

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| RESULT | 1st column | ID (in input table) data type | ID (in input table) column name | ID |
| | 2nd column | DOUBLE | SCORE | Value $Y_i$ |
| | 3rd column | VARCHAR | CLASS | Category |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_FLOGISTICR_PREDICTDATA_TBL;
CREATE COLUMN TABLE PAL_FLOGISTICR_PREDICTDATA_TBL ("ID" INTEGER, "V1"
VARCHAR(5000),"V2" DOUBLE, "V3" INTEGER);
INSERT INTO PAL_FLOGISTICR_PREDICTDATA_TBL VALUES (0,'B',2.62,0);
INSERT INTO PAL_FLOGISTICR_PREDICTDATA_TBL VALUES (1,'B',2.875,0);
INSERT INTO PAL_FLOGISTICR_PREDICTDATA_TBL VALUES (2,'A',2.32,1);
INSERT INTO PAL_FLOGISTICR_PREDICTDATA_TBL VALUES (3,'A',3.215,2);
INSERT INTO PAL_FLOGISTICR_PREDICTDATA_TBL VALUES (4,'B',3.44,3);
INSERT INTO PAL_FLOGISTICR_PREDICTDATA_TBL VALUES (5,'B',3.46,0);
INSERT INTO PAL_FLOGISTICR_PREDICTDATA_TBL VALUES (6,'A',3.57,1);
INSERT INTO PAL_FLOGISTICR_PREDICTDATA_TBL VALUES (7,'B',3.19,2);
INSERT INTO PAL_FLOGISTICR_PREDICTDATA_TBL VALUES (8,'A',3.15,3);
INSERT INTO PAL_FLOGISTICR_PREDICTDATA_TBL VALUES (9,'B',3.44,0);
INSERT INTO PAL_FLOGISTICR_PREDICTDATA_TBL VALUES (10,'B',3.44,1);
INSERT INTO PAL_FLOGISTICR_PREDICTDATA_TBL VALUES (11,'A',4.07,3);
INSERT INTO PAL_FLOGISTICR_PREDICTDATA_TBL VALUES (12,'A',3.73,1);
INSERT INTO PAL_FLOGISTICR_PREDICTDATA_TBL VALUES (13,'B',3.78,2);
INSERT INTO PAL_FLOGISTICR_PREDICTDATA_TBL VALUES (14,'B',5.25,2);
INSERT INTO PAL_FLOGISTICR_PREDICTDATA_TBL VALUES (15,'A',5.424,3);
```

```
INSERT INTO PAL_FLOGISTICR_PREDICTDATA_TBL VALUES (16,'A',5.345,0);
INSERT INTO PAL_FLOGISTICR_PREDICTDATA_TBL VALUES (17,'B',2.2,1);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL ("NAME" VARCHAR(50),
"INTARGS" INTEGER, "DOUBLEARGS" DOUBLE, "STRINGARGS" VARCHAR(100));
INSERT INTO #PAL_PARAMETER_TBL VALUES ('THREAD_RATIO',null,0.1,null);
CALL _SYS_AFL.PAL_LOGISTIC_REGRESSION_PREDICT(PAL_FLOGISTICR_PREDICTDATA_TBL,
PAL_LOGISTICR_RESULT_TBL, "#PAL_PARAMETER_TBL",?);
```

**Expected Result**

RESULT:

| | ID | SCORE | CLASS |
|---|---|---|---|
| 1 | 0 | 0.9503788846701562 | 1 |
| 2 | 1 | 0.8485553884340249 | 1 |
| 3 | 2 | 0.9556025386215554 | 1 |
| 4 | 3 | 0.03698927571246... | 0 |
| 5 | 4 | 0.02228944797832... | 0 |
| 6 | 5 | 0.250405491502889 | 0 |
| 7 | 6 | 0.04945455744432... | 0 |
| 8 | 7 | 0.00991421570968... | 0 |
| 9 | 8 | 0.28528825664473... | 0 |
| 10 | 9 | 0.2689318719297027 | 0 |
| 11 | 10 | 0.02200456061558... | 0 |
| 12 | 11 | 0.00471217032708... | 0 |
| 13 | 12 | 0.0234948249080189 | 0 |
| 14 | 13 | 0.00058244758818... | 0 |
| 15 | 14 | 0.00000048785106... | 0 |
| 16 | 15 | 0.00000693226734... | 0 |
| 17 | 16 | 0.00016367305056... | 0 |
| 18 | 17 | 0.8986838180554119 | 1 |

> **Note**
>
> The wrapper procedure generation approach to call PAL functions is still supported. For more information, refer to the older versions (prior to SAP HANA Platform 2.0 SPS 02) of this document.

## Related Information

Multi-Class Logistic Regression [page 304]

## 5.2.8 Multi-Class Logistic Regression

Multi-class logistic regression is also called multinomial logistic regression, it is a classification method to solve logistic regression for multi-class problems.

Multi-class logistic regression can be deemed as an extension for binary logistic regression. Unlike binary logistic regression whose dependent variable can only have two different values, multi-class logistic regression extends dependent variable to have multiple different values. Like binary logistic regression, for each data instance, it tries to model the logit function of the probability for each choice of dependent variable as a linear combination of indepnendent variable. However, in multi-class logistic regression there are more than two different values for dependent variable, that means it needs to model independently the logit function for each possible value of dependent variable as choice. Thus, when adopting multi-class logistic regression algorithm, it implies the assumption of `independence of irrelevant alternatives`, which may not always be desirable. This assumption states the choice between two class values do not depend on the presence or absence of third class as choice.

This model consists of two procedures, one for training and one for prediction. Below will state how training and prediction are performed in theory. Let *N* be the sample size, *P* be the feature number, *K* be the label size.

In the training phase, the training sample input consists of features and label for each row. We use $\mathbf{X} \in \mathbf{R}^{N \times (P+1)}$ to represent the feature where $X_{i,p}$ is the p-th feature of the i-th sample, while $\mathbf{X}_{i,0}$ always has value 1. $\mathbf{y} \in \mathbf{R}^{N}$ is the label input vector, where $\mathbf{y}_i \in \{1, 2, ..., K\}$ is the label of the i-th sample. The output model is a coefficient weight matrix $\mathbf{W} \in \mathbf{R}^{(P+1) \times K}$. Here uses $\mathbf{W}_{0,k}$ to represent intercept coefficient corresponding to label `k`. The optimal coefficient weight will be represented as $\mathbf{W}^*$. Theoretically the optimal model is obtained from following optimization problem:

$$\mathbf{W}^* = \arg\min_{\mathbf{W}} -\frac{1}{N}\sum_{i=1}^{N}\left[\sum_{k=1}^{K} 1_{y_i=k} \cdot \mathbf{X}_i \cdot \mathbf{W}_k - \log\left(\sum_{k=1}^{k} \exp(\mathbf{X}_i \cdot \mathbf{W}_k)\right)\right]$$

In above formula, $1_{y_i=k}$ represents boolean function that return value 1 when $y_i = k$ or value 0 otherwise.

In the testing phase, the test sample set consists of features only; The input model is obtained from training phase. To differentiate from training data instance, here we use $\widetilde{\mathbf{X}_i}$ for test data instance. For each test data instance, output consists of result label and the probability of this result.

$$\widetilde{y_i} = \arg\max_{k} \exp(\widetilde{\mathbf{X}_i} \cdot \mathbf{W}_k^*)$$

$$\widetilde{c_i} = \frac{\exp(\widetilde{\mathbf{X}_i} \cdot \mathbf{W}_{\widetilde{y_i}}^*)}{\sum_{k=1}^{K} \exp(\widetilde{\mathbf{X}_i} \cdot \mathbf{W}_k^*)}$$

## _SYS_AFL.PAL_MULTICLASS_LOGISTIC_REGRESSION

### Prerequisites

- No missing or null value in the data input table
- Dependent variable column of input data table must have at least two different values

This procedure implements the training function of multi-class logistic regression. The procedure takes input for training data and configuration parameters. After finishing training, it will output fitted model plus statistics for training result. For parameter selection, it will also output selected optimal parameters.

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | DATA table | IN |
| 2 | PARAMETER table | IN |
| 3 | RESULT table | OUT |
| 4 | SEMISTRUCTURED_RESULT table | OUT |
| 5 | STATISTIC table | OUT |
| 6 | OPTIMAL_PARAM table | OUT |

### Input Table

### Data Table

| Column | Column Data Type | Description |
|---|---|---|
| "ID" | INTEGER, VARCHAR, NVARCHAR | "ID" |
| Feature columns | INTEGER, VARCHAR, NVARCHAR, DOUBLE, or DECIMAL (p, s) | Features, X |
| Label column | INTEGER, VARCHAR, NVARCHAR | Label, y |

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| MAX_ITERATION | INTEGER | 100 | Maximum number of iterations of the optimization. | |
| PMML_EXPORT | INTEGER | 0 | • 0: Does not export logistic regression model in PMML.<br>• 1: Exports logistic regression model in PMML. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| JSON_EXPORT | INTEGER | 0 | • 0: Does not export logistic regression model in JSON.<br>• 1: Exports logistic regression model in JSON.<br><br>**Note**<br>Currently either PMML or JSON format model can be exported. JSON format is preferred if both formats are to be exported. | |
| SELECTED_FEATURES | VARCHAR | No default value | A string to specify the features that will be processed. The pattern is "X1,...,Xn", where Xi is the corresponding column name in the data table. If this parameter is not specified, all the features will be processed. | Only used when you need to indicate the needed features. |
| DEPENDENT_VARIABLE | VARCHAR | No default value | Column name in the data table used as dependent variable. | Only used when you need to indicate the dependence. |
| METHOD | INTEGER | Automatically determined | • 0: lbfgs<br>• 1: cyclical coordinate descent | |
| ENET_LAMBDA | DOUBLE | 0 | Penalized weight. The value should be equal to or greater than 0. | |
| ENET_ALPHA | DOUBLE | 1 | The elastic net mixing parameter. The value range is between 0 and 1 inclusively.<br>• 0: Ridge penalty<br>• 1: LASSO penalty | METHOD 0 only solves Ridge penalty problem. |
| EXIT_THRESHOLD | DOUBLE | 1.00E-05 | Convergence threshold for exiting iterations. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| HAS_ID | INTEGER | 0 | Specifies whether the input data table has an ID column.<br>• 0: No<br>• 1: Yes | |
| CATEGORICAL_VARIABLE | VARCHAR | No default value | Indicates whether or not a column data is corresponding to a category variable even the data type of this column is INTEGER.<br>By default, `VARCHAR` or `NVARCHAR` is category variable, and `INTEGER` or `DOUBLE` is continuous variable. | |
| STANDARDIZE | INTEGER | 1 | Controls whether to standardize the data to have zero mean and unit variance.<br>• 0: No<br>• 1: Yes | |
| STAT_INF | INTEGER | 0 | Controls whether to proceed statistical inference.<br>• 0: Does not proceed statistical inference.<br>• 1: Proceeds statistical inference. | |
| THREAD_RATIO | DOUBLE | 1 | Specifies the ratio of total number of threads that can be used by this function. The value range is from 0 to 1, where 0 means only using one thread, and 1 means using at most all the currently available threads. Values outside the range will be ignored and this function heuristically determines the number of threads to be used. | |

## Model Evaluation and Parameter Selection

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| RESAMPLING_METHOD | NVARCHAR | No default value | Specifies the resampling method for model evaluation or parameter selection.<br>• cv<br>• stratified_cv<br>• bootstrap<br>• stratified_bootstrap<br>• cv_sha<br>• cv_hyperband<br>• bootstrap_sha<br>• bootstrap_hyperband<br>• stratified_cv_sha<br>• stratified_cv_hyperband<br>• stratified_bootstrap_sha<br>• stratified_bootstrap_hyperband<br>If no value is specified for this parameter, neither model evaluation nor parameter selection is activated. | |
| RESOURCE | NVARCHAR | No default value | Resource type used in successive-halving and hyperband algorithm for parameter selection.<br>• MAX_ITERATION | Mandatory and valid only when RESAMPLING_METHOD parameter has values with sha or hyperband suffix. |
| MAX_RESOURCE | INTEGER | No default value | Maximum allowed resource budget for single hyper-parameter candidate. Valid value must be larger than 0. | Mandatory and valid only when RESOURCE parameter is set. |
| MIN_RESOURCE_RATE | DOUBLE | 0 | The rate between minimum allowed resource and maximum allowed resource. Valid value is between [0, 1). | Valid only when RESOURCE parameter is set. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| REDUCTION_RATE | DOUBLE | 3.0 | This parameter specifies the reduction rate of available size of hyper-parameter candidates. For each round, the available parameter candidate size will be divided by value of this parameter. Thus , valid value for this parameter must be larger than 1.0. | Valid only when RESOURCE parameter is set. |
| AGGRESSIVE_ELIMINATION | INTEGER | 0 | Boolean value indicates whether to perform aggressive elimination behavior for successive-halving algorithm. When set to 1, it will eliminate more parameter candidates than expected, which is defined by REDUCTION_RATE parameter; This can accelerate the run-time performance but may result in sub-optimal hyper-parameter candidate.<br>• 0: Does not perform aggressive elimination;<br>• 1: Performs aggressive elimination | Valid only when RESAMPLING_METHOD parameter has value with suffix sha. |
| PS_VERBOSE | INTEGER | 1 | Boolean value indicates whether output optimal hyper-parameter and all involved hyper-parameter candidates related evaluation statistics in Statistics output table.<br>• 0: Does not output evaluation statistics<br>• 1. Outputs evaluation statistics | Valid only when RESAMPLING_METHOD parameter is set. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| EVALUATION_METRIC | NVARCHAR | No default value | Specifies the evaluation metric for model evaluation or parameter selection.<br>• ACCURACY<br>• F1_SCORE<br>• AUC<br>• NLL | |
| FOLD_NUM | INTEGER | No default value | Specifies the fold number for a cross validation method. | Mandatory and valid only when RESAMPLING_METHOD is set to cv, stratified_cv, cv_sha, cv_hyperband, stratified_cv_sha, or stratified_cv_hyperband. |
| REPEAT_TIMES | INTEGER | 1 | Specifies the number of repeat times for resampling. | |
| PARAM_SEARCH_STRATEGY | NVARCHAR | No default value | Specifies the method to activate parameter selection.<br>• grid<br>• random | This parameter is only valid when RESAMPLING_METHOD does not have value with suffix hyperband; Otherwise, this parameter is always considered to have value random. |
| RANDOM_SEARCH_TIMES | INTEGER | No default value | Specifies the number of times to randomly select candidate parameters for selection. | Mandatory and valid when PARAM_SEARCH_STRATEGY is set to random. |
| SEED | INTEGER | 0 | Specifies the seed for random generation. Use system time when 0 is specified. | |
| TIMEOUT | INTEGER | 0 | Specifies maximum running time for model evaluation or parameter selection, in seconds. No timeout when 0 is specified. | |
| PROGRESS_INDICATOR_ID | NVARCHAR | No default value | Sets an ID of progress indicator for model evaluation or parameter selection. No progress indicator is active if no value is provided. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| ENET_LAMBDA_VALUES | NVARCHAR | No default value | Specifies values of ENET_LAMBDA to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| ENET_LAMBDA_RANGE | NVARCHAR | No default value | Specifies the range of ENET_LAMBDA to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| ENET_ALPHA_VALUES | NVARCHAR | No default value | Specifies values of ENET_ALPHA to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| ENET_ALPHA_RANGE | NVARCHAR | No default value | Specifies the range of ENET_ALPHA to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |

## Output Tables

### RESULT Table

| Column | Data Type | Column name | Description |
|---|---|---|---|
| 1st column | NVARCHAR (1000) | VARIABLE_NAME | Similar to p, corresponding to a feature. |
| 2nd column | NVARCHAR (100) | CLASS | Similar to k, corresponding to a label |
| 3rd column | DOUBLE | COEFFICIENT | $w_{pk}$ |
| 4th column | DOUBLE | Z_SCORE | (When STAT_INF = 1) Zscore of each coefficient parameter |
| 5th column | DOUBLE | P_VALUE | (When STAT_INF = 1) PValue of each coefficient parameter |

### SEMISTRUCTURED_RESULT Table

| Column | Data Type | Column name | Description |
|---|---|---|---|
| 1st column | INTEGER | ROW_INDEX | "ID" |
| 2nd column | NVARCHAR (5000) | MODEL_CONTENT | Model in PMML or JSON format |

### STATISTIC Table

| Column | Data Type | Column name | Description |
|---|---|---|---|
| 1st column | NVARCHAR (256) | STAT_NAME | Statistics name |
| 2nd column | NVARCHAR (1000) | STAT_VALUE | Statistics value |

### OPTIMAL_PARAM Table

| Column | Data Type | Column name | Description |
|---|---|---|---|
| 1st column | NVARCHAR (256) | PARAM_NAME | |
| 2nd column | INTEGER | INT_VALUE | |
| 3rd column | DOUBLE | DOUBLE_VALUE | |
| 4th column | NVARCHAR (1000) | STRING_VALUE | |

### General Guides

This section will only describe the normal logistic model without regularization. For biased model with regularization, follow the link below.

Besides input data table, the training procedures depend on the input configuration in parameter table to specify the input and output, to define the behaviour of training and so on.

### Parameters for Specifying Input

- HAS_ID: Specifies whether the first column of input data is an ID column.
- SELECTED_FEATURES: Specifies the columns enumerating all feature columns; Otherwise by default all provided columns except for ID and DEPENDENT column will be considered as feature columns.
- DEPENDENT_VARIABLE: Specifies the dependent column; Otherwise by default the column comes after the optional ID column will be considered as DEPENDENT column.

The ID column, if exists in input data column, always locates in first column. The dependent variable column, if not specified in parameter `DEPENDENT_VARIABLE`, always locates in the last column.

### Parameters for Controlling Output Model Format

Now, the training procedure supports three model formats:

- Key-value pair format
- PMML format
- JSON format

Among them, the key-value pair format model will always be output. Unlike PMML format and JSON format, the key-value pair format model has two extra optional statistics output for each coefficient. Output or not can be controlLed by parameter STAT_INF. The two extra statistics are Z-statistic and its p-value, they are derived from two-tailed Z-test with null hypothesis that coefficient is 0. Larger absolute value of Z-statistic indicates more significant current coefficient related to dependent variable, p-value represents the probability of obtaining test statistic that is at least as extreme as currently obtained value if null hypothesis is true.

To control the output of PMML format and JSON format, the procedure introduces following parameters:

- PMML_EXPORT
- JSON_EXPORT

Each parameter above controls corresponding format model output. By default, both formats will not be output. However, if both formats are specified to be output, only JSON format will be output.

After obtaining the training data and before committing training, there are some preprocessing steps to be done:

- Encode categorical variable
- Standardize input data

Since categorical variable cannot be used directly in training procedure, they must be first converted to numerical data before feeding into the training algorithm. Currently the only supported encoding method is one-hot encoding, which is built into the training procedure. Users don't need to be aware of this process except that user must tell procedure which variable is categorical variable. Only INTEGER type independent categorical variable needs to be specified. DOUBLE type variable is always treated as continuous variable and VARCHAR or NVARCHAR is always treated as categorical variable. INTEGER type variable is by default treated as continuous variable, unless user lists the variable name with parameter `CATEGORICAL_VARIABLE`.

Different independent variables may have different units, in order for all independent variables to have uniform units, user may set parameter `STANDARDIZE` to standardize all independent variables. Currently by default this parameter is set.

Regarding training behaviour, current procedure implements two algorithms:

- LBFGS
- Cyclical coordinate descent

For normal logistic regression without regularization, both above algorithms can be used to do the training task. Users can use `METHOD` to choose one. Currently the default one is `LBFGS` because it normally runs much faster than `Cyclical Coordinate Descent` algorithm.

Both algorithms are iterative based algorithms, both following parameters can be applied to either algorithm:

- MAX_ITERATION
- EXIT_THRESHOLD

`MAX_ITERATION` parameter is used to limit maximum available iteration steps for the applied algorithm, which can in turn limit the time resource. `EXIT_THRESHOLD` parameter controls the accuracy of the obtained model, the smaller this parameter is, the more accurate of the obtained model, but also means that more iteration steps are needed. After obtaining the trained model, user should beware of the value with key `solution status` in STATISTIC output table. If the value is `Converged`, that means the obtained model is an ideal model based on current parameter setting; Otherwise user may consider to adjust above parameter and re-train.

In STATISTIC output table the procedure will output some metadata and results for current training. Current implementation will output following information:

- AIC: Akaike Information Criterion
- obj: objective value for the object function after final iteration of training
- iter: total iteration steps
- method: actual used algorithm in training
- solution status: converge status for current training

### Examples

#### Example: Fitting logistic model

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_MCLR_DATA_TBL;
CREATE COLUMN TABLE PAL_MCLR_DATA_TBL (
    "V1" VARCHAR (50),
    "V2" DOUBLE,
    "V3" INTEGER,
    "CATEGORY" INTEGER
);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',2.62,0,1);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',2.875,0,1);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('A',2.32,1,1);
```

```
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('A',3.215,2,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',3.44,3,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',3.46,0,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('A',3.57,1,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',3.19,2,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('A',3.15,3,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',3.44,0,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',3.44,1,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('A',4.07,3,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('A',3.73,1,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',3.78,2,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',5.25,2,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('A',5.424,3,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('A',5.345,0,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',2.2,1,1);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',1.615,2,1);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('A',1.835,0,1);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',2.465,3,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('A',3.52,1,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('A',3.435,0,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',3.84,2,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',3.845,3,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('A',1.935,1,1);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',2.14,0,1);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',1.513,1,1);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('A',3.17,3,1);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',2.77,0,1);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',3.57,0,1);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('A',2.78,3,1);
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL (
    "PARAM_NAME" NVARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR (1000)
);
INSERT INTO PAL_PARAMETER_TBL VALUES ('MAX_ITERATION',500,NULL,NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('JSON_EXPORT',1,NULL,NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('HAS_ID',0,NULL,NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('STANDARDIZE',1,NULL,NULL);

DROP TABLE PAL_MCLR_MODEL_TBL;
CREATE COLUMN TABLE PAL_MCLR_MODEL_TBL (
    "VARIABLE_NAME" NVARCHAR(1000),
    "CLASS" NVARCHAR(100),
    "COEFFICIENT" DOUBLE,
    "Z_SCORE" DOUBLE,
    "P_VALUE" DOUBLE
);
DROP TABLE PAL_MCLR_JSON_TBL;
CREATE COLUMN TABLE PAL_MCLR_JSON_TBL (
    "ROW_INDEX" INT,
    "MODEL_CONTENT" NVARCHAR(5000)
);
DROP TABLE PAL_MCLR_STATISTIC_TBL;
CREATE COLUMN TABLE PAL_MCLR_STATISTIC_TBL ("STAT_NAME"
NVARCHAR(256),"STAT_VALUE" NVARCHAR(1000));
DO BEGIN
lt_data = SELECT * FROM PAL_MCLR_DATA_TBL;
lt_para = SELECT * FROM PAL_PARAMETER_TBL;
CALL _SYS_AFL.PAL_MULTICLASS_LOGISTIC_REGRESSION
(:lt_data,:lt_para,lt_result,lt_json_model,lt_stat,lt_opt_para);
INSERT INTO PAL_MCLR_MODEL_TBL
SELECT * FROM :lt_result;
INSERT INTO PAL_MCLR_JSON_TBL
SELECT * FROM :lt_json_model;
INSERT INTO PAL_MCLR_STATISTIC_TBL
SELECT * FROM :lt_stat;
```

```
END;
SELECT * FROM PAL_MCLR_MODEL_TBL;
SELECT * FROM PAL_MCLR_JSON_TBL;
SELECT * FROM PAL_MCLR_STATISTIC_TBL;
```

### Expected Results

### Coefficient

| | VARIABLE_NAME | CLASS | COEFFICIENT | Z_SCORE | P_VALUE |
|---|---|---|---|---|---|
| 1 | __PAL_INTERCEPT__ | 1 | 16.20055608955917 | *NULL* | *NULL* |
| 2 | __PAL_INTERCEPT__ | 0 | 0 | *NULL* | *NULL* |
| 3 | V2 | 1 | -4.846572670488794 | *NULL* | *NULL* |
| 4 | V2 | 0 | 0 | *NULL* | *NULL* |
| 5 | V3 | 1 | -0.9076037804267292 | *NULL* | *NULL* |
| 6 | V3 | 0 | 0 | *NULL* | *NULL* |
| 7 | V1__PAL_KEY_DELIMIT__A | 1 | 0.23509366683932617 | *NULL* | *NULL* |
| 8 | V1__PAL_KEY_DELIMIT__A | 0 | 0 | *NULL* | *NULL* |
| 9 | V1__PAL_KEY_DELIMIT__B | 1 | -0.8751286994969815 | *NULL* | *NULL* |
| 10 | V1__PAL_KEY_DELIMIT__B | 0 | 0 | *NULL* | *NULL* |

### JSON

| | ROW_INDEX | |
|---|---|---|
| 1 | 1 | {"Metadata":{"ContinuousFeature":["V2","V3"],"CategoricalFeature":["V1"],"CategoricalValue":[["A","B"]],"Label":["1","0"]},"Coefficient":{"InterceptCoe |

### Statistics

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | AIC | 26.634071919175714 |
| 2 | obj | 0.25990737373712053 |
| 3 | iter | 10 |
| 4 | method | lbfgs |
| 5 | solution status | Converged |

## Biased Model Guides

Refer to Biased Model [page 319] of this procedure.

## Model Evaluation and Parameter Selection Guides

Refer to Model Evaluation and Parameter Selection [page 322] of this procedure.

## _SYS_AFL.PAL_MULTICLASS_LOGISTIC_REGRESSION_PREDICT

### Prerequisites

- No missing or null value in the data input table

This procedure implements the prediction function of multi-class logistic regression.

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | DATA table | IN |
| 2 | MODEL table | IN |
| 3 | PARAMETER table | IN |
| 4 | RESULT table | OUT |

### Input Tables

#### Data Table

| Column | Reference for Output Table | Data Type | Description |
|---|---|---|---|
| 1st column | "ID" | INTEGER, VARCHAR, NVARCHAR | Sample ID |
| Feature columns | | INTEGER, VARCHAR, NVARCHAR, DOUBLE, or DECIMAL (p, s) | Features, X |

#### Model Table

#### Key-Value Format

| Column | Data Type | Description |
|---|---|---|
| 1st column | VARCHAR or NVARCHAR | Similar to p, corresponding to a feature |
| 2nd column | VARCHAR or NVARCHAR | Similar to k, corresponding to a label |
| 3rd column | DOUBLE | $W_{pk}$ |

#### Semi-Structured Format

| Column | Data Type | Description |
|---|---|---|
| 1st column | INTEGER | "ID" |
| 2nd column | VARCHAR or NVARCHAR | Model in PMML or JSON format |

### Parameter Table

Mandator Parameter

None.

Optional Parameter

The following parameter is optional. If it is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| VERBOSE_OUTPUT | INTEGER | 0 | • 0: Outputs the category of the highest probability only<br>• 1: Outputs the probability of all label categories |
| IGNORE_UNKNOWN_CATEGORY | INTEGER | 1 | • 0: Procedures will report error if unknown category value is found<br>• 1: Ignores unknown category value |
| THREAD_RATIO | DOUBLE | 1.0 | Specifies the ratio of total number of threads that can be used by this function. The value range is from 0 to 1, where 0 means only using one thread, and 1 means using at most all the currently available threads. Values outside the range will be ignored and this function heuristically determines the number of threads to be used. |

### Output Tables

### RESULT Table

| Column | Data Type | Column Name | Description |
|---|---|---|---|
| 1st column | ID (in input table) data type | ID (in input table) column name | "ID" |
| 2nd column | NVARCHAR (100) | CLASS | Predicted label for the sample |
| 3rd column | DOUBLE | PROBABILITY | Confidence for the prediction of the sample |

### General Guides for Prediction

This procedure is used to predict the associated probability for each possible class label value with provided data instance and a previously trained logistic model. The independent feature name and its type should precisely match corresponding independent feature in training data that is used to obtain current trained model, otherwise unexpected results will happen.

For input data table, the first column should always be ID column, followed by all independent features that was used in the training phase. The feature used in prediction procedure is matched with column name to corresponding feature used in data table during the training phase.

In principle, the prediction procedure is able to associate probability for each possible class label per prediction data instance. There is an option to output exactly one class label with maximum associated probability per prediction data instance. Whether to output all class labels or exactly one is controlled by parameter `VERBOSE_OUTPUT`. By default, only one class label and associated probability will be output.

During prediction, categorical feature for some prediction data instance may have values unseen during training, there are two ways to handle this situation:

- Report this abnormal issue
- Use an all zero vector for this categorical feature after one-hot encoding

Users can use parameter `IGNORE_UNKNOWN_CATEGORY` to choose one way above. Currently the default way is the second way above.

### Examples

#### Example: Prediction with Key-Value Format Model

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_MCLR_PREDICT_DATA_TBL;
CREATE COLUMN TABLE PAL_MCLR_PREDICT_DATA_TBL (
    "ID" INTEGER,
    "V1" VARCHAR (50),
    "V2" DOUBLE,
    "V3" INTEGER
);
INSERT INTO PAL_MCLR_PREDICT_DATA_TBL VALUES (0,'B',2.62,0);
INSERT INTO PAL_MCLR_PREDICT_DATA_TBL VALUES (1,'B',2.875,0);
INSERT INTO PAL_MCLR_PREDICT_DATA_TBL VALUES (2,'A',2.32,1);
INSERT INTO PAL_MCLR_PREDICT_DATA_TBL VALUES (3,'A',3.215,2);
INSERT INTO PAL_MCLR_PREDICT_DATA_TBL VALUES (4,'B',3.44,3);
INSERT INTO PAL_MCLR_PREDICT_DATA_TBL VALUES (5,'B',3.46,0);
INSERT INTO PAL_MCLR_PREDICT_DATA_TBL VALUES (6,'A',3.57,1);
INSERT INTO PAL_MCLR_PREDICT_DATA_TBL VALUES (7,'B',3.19,2);
INSERT INTO PAL_MCLR_PREDICT_DATA_TBL VALUES (8,'A',3.15,3);
INSERT INTO PAL_MCLR_PREDICT_DATA_TBL VALUES (9,'B',3.44,0);
INSERT INTO PAL_MCLR_PREDICT_DATA_TBL VALUES (10,'B',3.44,1);
INSERT INTO PAL_MCLR_PREDICT_DATA_TBL VALUES (11,'A',4.07,3);
INSERT INTO PAL_MCLR_PREDICT_DATA_TBL VALUES (12,'A',3.73,1);
INSERT INTO PAL_MCLR_PREDICT_DATA_TBL VALUES (13,'B',3.78,2);
INSERT INTO PAL_MCLR_PREDICT_DATA_TBL VALUES (14,'B',5.25,2);
INSERT INTO PAL_MCLR_PREDICT_DATA_TBL VALUES (15,'A',5.424,3);
INSERT INTO PAL_MCLR_PREDICT_DATA_TBL VALUES (16,'A',5.345,0);
INSERT INTO PAL_MCLR_PREDICT_DATA_TBL VALUES (17,'B',2.2,1);
INSERT INTO PAL_MCLR_PREDICT_DATA_TBL VALUES (18,'B',1.615,2);
INSERT INTO PAL_MCLR_PREDICT_DATA_TBL VALUES (19,'A',1.835,0);
INSERT INTO PAL_MCLR_PREDICT_DATA_TBL VALUES (20,'B',2.465,3);
INSERT INTO PAL_MCLR_PREDICT_DATA_TBL VALUES (21,'A',3.52,1);
INSERT INTO PAL_MCLR_PREDICT_DATA_TBL VALUES (22,'A',3.435,0);
INSERT INTO PAL_MCLR_PREDICT_DATA_TBL VALUES (23,'B',3.84,2);
INSERT INTO PAL_MCLR_PREDICT_DATA_TBL VALUES (24,'B',3.845,3);
INSERT INTO PAL_MCLR_PREDICT_DATA_TBL VALUES (25,'A',1.935,1);
INSERT INTO PAL_MCLR_PREDICT_DATA_TBL VALUES (26,'B',2.14,0);
INSERT INTO PAL_MCLR_PREDICT_DATA_TBL VALUES (27,'B',1.513,1);
INSERT INTO PAL_MCLR_PREDICT_DATA_TBL VALUES (28,'A',3.17,3);
INSERT INTO PAL_MCLR_PREDICT_DATA_TBL VALUES (29,'B',2.77,0);
INSERT INTO PAL_MCLR_PREDICT_DATA_TBL VALUES (30,'B',3.57,0);
INSERT INTO PAL_MCLR_PREDICT_DATA_TBL VALUES (31,'A',2.78,3);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" NVARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('VERBOSE_OUTPUT',0,NULL,NULL);
CALL _SYS_AFL.PAL_MULTICLASS_LOGISTIC_REGRESSION_PREDICT
(PAL_MCLR_PREDICT_DATA_TBL,PAL_MCLR_MODEL_TBL,#PAL_PARAMETER_TBL,?);
```

#### Expected Results

### Results

| | ID | CLASS | PROBABILITY |
|---|---|---|---|
| 1 | 0 | 1 | 0.932604753544292 |
| 2 | 1 | 1 | 0.8008365401061912 |
| 3 | 2 | 1 | 0.986400070044168 |
| 4 | 3 | 0 | 0.7233797981694314 |
| 5 | 4 | 0 | 0.9832026470279668 |
| 6 | 5 | 0 | 0.8090291345312913 |
| 7 | 6 | 0 | 0.8549814934145565 |
| 8 | 7 | 0 | 0.8754844725725006 |
| 9 | 8 | 0 | 0.8254713286705403 |
| 10 | 9 | 0 | 0.7936031242945665 |
| 11 | 10 | 0 | 0.9050282239668521 |
| 12 | 11 | 0 | 0.997558534498266 |
| 13 | 12 | 0 | 0.9275513475252768 |
| 14 | 13 | 0 | 0.9919165872526643 |
| 15 | 14 | 0 | 0.9999934385126622 |
| 16 | 15 | 0 | 0.9999965425360743 |
| 17 | 16 | 0 | 0.9999228190853654 |
| 18 | 17 | 1 | 0.9771427351767369 |
| 19 | 18 | 1 | 0.9966082270642568 |
| 20 | 19 | 1 | 0.9994700452864712 |
| 21 | 20 | 1 | 0.6583195292704851 |
| 22 | 21 | 0 | 0.8222826497969367 |
| 23 | 22 | 0 | 0.5528833129991736 |
| 24 | 23 | 0 | 0.9939439270126126 |
| 25 | 24 | 0 | 0.9976061506176541 |
| 26 | 25 | 1 | 0.997870907020761 |
| 27 | 26 | 1 | 0.9929926375530904 |
| 28 | 27 | 1 | 0.9991630920064429 |
| 29 | 28 | 0 | 0.8389987314485977 |
| 30 | 29 | 1 | 0.8699391776072998 |
| 31 | 30 | 0 | 0.8783435850755609 |
| 32 | 31 | 1 | 0.5595575004260337 |

## 5.2.8.1 Biased Model

The pure logit model has low bias but high variance, thus prediction accuracy will be severely impacted when we overfit the training data.

Current training procedure supports elastic-net regularization to improve the prediction accuracy.

### Elastic-Net Regularization

Elastic net regularization for multi-class logistic regression seeks to find $\mathbf{W}$ that minimizes:

$$\mathbf{W}^* = \arg\min_{\mathbf{W}} -\frac{1}{N}\sum_{i=1}^{N}\left[\sum_{k=1}^{K} 1_{y_i=k}\cdot \mathbf{X}_i\cdot \mathbf{W}_k - \log\left(\sum_{k=1}^{k}\exp(\mathbf{X}_i\cdot \mathbf{W}_k)\right)\right] + \lambda \sum_{k=1}^{k} P_\alpha(\mathbf{W}_{0^-}, k)$$

Where $P_\alpha(\mathbf{W}_{0^-}, k) = (1-\alpha)\cdot\frac{1}{2}\|\mathbf{W}_{0^-}, k\|_2^2 + \alpha\cdot\|\mathbf{W}_{0^-}, k\|_1$. Here $\mathbf{W}_{0^-}, k$ represents vector but excludes $\mathbf{W}_{0^-k}$. The definition for $\mathbf{X}_i$ is the same with definition in parent page.

#### Parameters to Control Elastic-Net Regularization

- ENET_LAMBDA

- ENET_ALPHA

These two parameters represent the λ and α in above math formula. From the math formula, it shows that elastic-net is a mixed form of Ridge regularization and Lasso regularization. The `ENET_ALPHA` represents the percentage of Lasso regularization in the elastic-net. When its value is 1, it represents a pure Lasso regularization. When it is 0, it represents pure Ridge regularization.

If users wish to obtain a model with some coefficient eliminated, they can prefer Lasso regularization in elastic-net form. The Lasso regularization can achieve a sparse model effect by setting some coefficient weight to 0.

Currently there are two methods to solve elastic-net Model: LBFGS and cyclical coordinate descent. However, only the latter method supports the full form of elastic-net model. LBFGS only supports Ridge form regularization, comparing runtime performance, LBFGS is much faster than the latter method. It's preferable to use LBFGS if possible. Two parameters can be used to control maximum iteration steps and model accuracy for both methods, for details, refer to the parent page.

**Examples for Regularization**

**Example: Fitting Multi-Class Logistic Regression Model with Elastic Net Penalty**

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_MCLR_DATA_TBL;
CREATE COLUMN TABLE PAL_MCLR_DATA_TBL (
    "V1" VARCHAR (50),
    "V2" DOUBLE,
    "V3" INTEGER,
    "CATEGORY" INTEGER
);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',2.62,0,1);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',2.875,0,1);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('A',2.32,1,1);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('A',3.215,2,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',3.44,3,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',3.46,0,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('A',3.57,1,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',3.19,2,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('A',3.15,3,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',3.44,0,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',3.44,1,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('A',4.07,3,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('A',3.73,1,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',3.78,2,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',5.25,2,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('A',5.424,3,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('A',5.345,0,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',2.2,1,1);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',1.615,2,1);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('A',1.835,0,1);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',2.465,3,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('A',3.52,1,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('A',3.435,0,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',3.84,2,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',3.845,3,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('A',1.935,1,1);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',2.14,0,1);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',1.513,1,1);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('A',3.17,3,1);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',2.77,0,1);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',3.57,0,1);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('A',2.78,3,1);
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL (
    "PARAM_NAME" NVARCHAR (256),
```

```
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR (1000)
);
INSERT INTO PAL_PARAMETER_TBL VALUES ('MAX_ITERATION',500,NULL,NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('HAS_ID',0,NULL,NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('STANDARDIZE',1,NULL,NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES('ENET_LAMBDA', NULL, 0.01, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES('ENET_ALPHA', NULL, 0.7, NULL);
DROP TABLE PAL_MCLR_MODEL_TBL;
CREATE COLUMN TABLE PAL_MCLR_MODEL_TBL (
    "VARIABLE_NAME" NVARCHAR(1000),
    "CLASS" NVARCHAR(100),
    "COEFFICIENT" DOUBLE,
    "Z_SCORE" DOUBLE,
    "P_VALUE" DOUBLE
);
DROP TABLE PAL_MCLR_STATISTIC_TBL;
CREATE COLUMN TABLE PAL_MCLR_STATISTIC_TBL ("STAT_NAME"
NVARCHAR(256),"STAT_VALUE" NVARCHAR(1000));
DO BEGIN
lt_data = SELECT * FROM PAL_MCLR_DATA_TBL;
lt_para = SELECT * FROM PAL_PARAMETER_TBL;
CALL _SYS_AFL.PAL_MULTICLASS_LOGISTIC_REGRESSION
(:lt_data,:lt_para,lt_result,lt_json_model,lt_stat,lt_opt_para);
INSERT INTO PAL_MCLR_MODEL_TBL
SELECT * FROM :lt_result;
INSERT INTO PAL_MCLR_STATISTIC_TBL
SELECT * FROM :lt_stat;
END;
SELECT * FROM PAL_MCLR_MODEL_TBL;
SELECT * FROM PAL_MCLR_STATISTIC_TBL;
```

## Expected Results

### Coefficient

| | VARIABLE_NAME | CLASS | COEFFICIENT | Z_SCORE | P_VALUE |
|---|---|---|---|---|---|
| 1 | __PAL_INTERCEPT__ | 1 | 5.959611369633791 | *NULL* | *NULL* |
| 2 | __PAL_INTERCEPT__ | 0 | -5.35264247857078 | *NULL* | *NULL* |
| 3 | V2 | 1 | -1.8055306192626162 | *NULL* | *NULL* |
| 4 | V2 | 0 | 1.7278313482719818 | *NULL* | *NULL* |
| 5 | V3 | 1 | -0.34017392352059933 | *NULL* | *NULL* |
| 6 | V3 | 0 | 0.24241356202856756 | *NULL* | *NULL* |
| 7 | V1__PAL_KEY_DELIMIT__A | 1 | 0.020094551224522084 | *NULL* | *NULL* |
| 8 | V1__PAL_KEY_DELIMIT__A | 0 | -0.11627533385659605 | *NULL* | *NULL* |
| 9 | V1__PAL_KEY_DELIMIT__B | 1 | -0.01760246374620425 | *NULL* | *NULL* |
| 10 | V1__PAL_KEY_DELIMIT__B | 0 | 0.11879614662284142 | *NULL* | *NULL* |

### Statistics

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | AIC | 37.30856034515221 |
| 2 | obj | 0.31083953713359036 |
| 3 | iter | 13 |
| 4 | method | cycle coordinate descent |
| 5 | solution status | Converged |

## 5.2.8.2 Model Evaluation and Parameter Selection

Parameter selection is an automated process to select best one among given hyper-parameters by training and evaluating model for each one.

For elastic-net model, it usually requires user's prior knowledge to set configuration parameter `ENET_LAMBDA` and `ENET_ALPHA`. However, users normally may not have such prior knowledge so they have to test various values to guess. To alleviate the continual try that is time-consuming and tedious, current implementation provides model evaluation and parameter selection function.

With parameter selection function, instead of providing a single value for `ENET_LAMBDA` and `ENET_ALPHA`, users can instead provide a set of candidates and stored procedure will automate the process to train and validate model with each candidate and select the best one, thus liberating users from previously time-consuming and tedious work.

The parameter selection process is time-consuming and linearly proportional to the candidate space size, in order to accelerate this process, current function supports early-stop technique to boost runtime performance. The early-stop technique is called successive-halving and hyperband algorithm, both of them are greedy algorithms and may fail to find best candidates.

### Successive-Halving and Hyperband

Successive-halving and hyperband are two algorithms to boost the parameter selection process.

Successive-halving algorithm is iteration based. With fixed total resource budget `B` and `n` candidates, initially algorithm will uniformly allocate resource `B/n` to each candidate. Then by evaluating all models trained with the given resource, the algorithm will reduce the candidates by preconfigured reduction rate depending on evaluation results, then in the next iteration, total resource budget `B` will be allocated uniformly to the survival candidates. This process will continue until only one candidate survives.

The above process is greedy and the final survival may not be optimal since the optimal candidate may falsely be eliminated in the intermediate iteration. In order to increase the chances of getting the optimal candidates the hyperband algorithm is introduced. With fixed total resource budget `B`, the algorithm will try different `n`. The larger value `n` is, the smaller value of `B/n`, therefore more aggressive for the early-stop. For each value of `n`, hyperband internally calls the successive-halving algorithm to get the optimal candidates. Finally, by comparing the evaluation result for different `n`, hyperband will get its final optimal candidates.

Note, both of them are approximate algorithms, so they can't guarantee the best candidate is obtained.

For details about the algorithm, refer to Successive Halving and Hyperband for Model Evaluation and Parameter Selection [page 2240].

**Parameters to Select Early-Stop Algorithm**

- RESAMPLING_METHOD

The early-stop algorithm is specified together with the sampling method. There are two supported sampling methods: cross validation and bootstrapping. Each of them can be combined with early-stop algorithm successive-halving or hyperband. Only limitation is that when using hyperband algorithm, the only supported candidate search method is random.

**Parameters to Control Early-Stop**

- RESOURCE
- MAX_RESOURCE
- MIN_RESOURCE_RATE
- REDUCTION_RATE
- AGGRESSIVE_ELIMINATION

Current implementation only supports iteration step number as resource. Both `MAX_RESOURCE` and `MIN_RESOURCE_RATE` are used to constrain the available resource for single candidate. The `MIN_RESOURCE_RATE` represents the minimum required resource divided by maximum allowed resource.

`REDUCTION_RATE` parameter and `AGGRESSIVE_ELIMINATION` parameter control the speed of candidate reductions and resource increase between iteration steps of whole algorithm. `AGGRESSIVE_ELIMINATION` is normally used when candidate size is too large comparing to relative small maximum allowed resource. In this case, after several iteration steps, each candidate will be allocated maximum allowed resource and never increases in following iterations if this parameter is turned off. To avoid this case, user can turn on this parameter so the algorithm behaviour will be more aggressive, so as to reduce the candidates in initial steps.

## Example for Successive-Halving and Hyperband

### Example: Parameter Selection with Successive-Halving

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_MCLR_DATA_TBL;
CREATE COLUMN TABLE PAL_MCLR_DATA_TBL (
    "V1" VARCHAR (50),
    "V2" DOUBLE,
    "V3" INTEGER,
    "CATEGORY" INTEGER
);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',2.62,0,1);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',2.875,0,1);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('A',2.32,1,1);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('A',3.215,2,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',3.44,3,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',3.46,0,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('A',3.57,1,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',3.19,2,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('A',3.15,3,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',3.44,0,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',3.44,1,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('A',4.07,3,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('A',3.73,1,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',3.78,2,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',5.25,2,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('A',5.424,3,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('A',5.345,0,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',2.2,1,1);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',1.615,2,1);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('A',1.835,0,1);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',2.465,3,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('A',3.52,1,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('A',3.435,0,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',3.84,2,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',3.845,3,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('A',1.935,1,1);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',2.14,0,1);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',1.513,1,1);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('A',3.17,3,1);
```

```
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',2.77,0,1);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',3.57,0,1);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('A',2.78,3,1);
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL (
    "PARAM_NAME" NVARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR (1000)
);
INSERT INTO PAL_PARAMETER_TBL VALUES ('MAX_ITERATION',500,NULL,NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('PMML_EXPORT',1,NULL,NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('HAS_ID',0,NULL,NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('STANDARDIZE',1,NULL,NULL);
--parameters to be selected
INSERT INTO PAL_PARAMETER_TBL VALUES ('ENET_LAMBDA_VALUES', NULL, NULL,
'{0.01,0.02,0.03167}');
INSERT INTO PAL_PARAMETER_TBL VALUES ('ENET_ALPHA_VALUES', NULL, NULL,
'{0.15,0.2,0.7}');
--Successive-Halving related parameters
INSERT INTO PAL_PARAMETER_TBL VALUES ('RESAMPLING_METHOD', NULL, NULL,
'cv_sha');
INSERT INTO PAL_PARAMETER_TBL VALUES ('RESOURCE', NULL, NULL, 'MAX_ITERATION');
INSERT INTO PAL_PARAMETER_TBL VALUES ('MAX_RESOURCE', 100, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('MIN_RESOURCE_RATE', NULL, 0.2, NULL);
--parameter selection parameters
INSERT INTO PAL_PARAMETER_TBL VALUES ('EVALUATION_METRIC', NULL, NULL,
'F1_SCORE');
INSERT INTO PAL_PARAMETER_TBL VALUES ('FOLD_NUM', 2, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('REPEAT_TIMES', 2, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('SEED', 2022, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('PARAM_SEARCH_STRATEGY', NULL, NULL,
'grid');
INSERT INTO PAL_PARAMETER_TBL VALUES ('PROGRESS_INDICATOR_ID', NULL, NULL,
'TEST');
DROP TABLE PAL_MCLR_STATISTIC_TBL;
CREATE COLUMN TABLE PAL_MCLR_STATISTIC_TBL ("STAT_NAME"
NVARCHAR(256),"STAT_VALUE" NVARCHAR(1000));
DROP TABLE PAL_MCLR_OPT_PARAM_TBL;
CREATE COLUMN TABLE PAL_MCLR_OPT_PARAM_TBL ("PARAM_NAME"
NVARCHAR(256),"INT_VALUE" INTEGER, "DOUBLE_VALUE" DOUBLE, "STRING_VALUE"
NVARCHAR(1000));
DO BEGIN
lt_data = SELECT * FROM PAL_MCLR_DATA_TBL;
lt_para = SELECT * FROM PAL_PARAMETER_TBL;
CALL _SYS_AFL.PAL_MULTICLASS_LOGISTIC_REGRESSION
(:lt_data,:lt_para,lt_result,lt_pmml_model,lt_stat,lt_opt_para);
INSERT INTO PAL_MCLR_STATISTIC_TBL
SELECT * FROM :lt_stat;
INSERT INTO PAL_MCLR_OPT_PARAM_TBL
SELECT * FROM :lt_opt_para;
END;
SELECT * FROM PAL_MCLR_STATISTIC_TBL;
SELECT * FROM PAL_MCLR_OPT_PARAM_TBL;
```

### Expected Results

### Statistics

| | STAT_NAME | |
|---|---|---|
| 1 | timeout | FALSE |
| 2 | TEST_1_F1_SCORE | 0.854545, 0.937254 |
| 3 | TEST_2_F1_SCORE | 0.875, 0.768115 |
| 4 | EVAL_RESULTS_1 | {"candidates":[{"TEST_F1_SCORE":[[[0.9372549019607843,0.7922077922077921],[0.8666666666666667,0.6761133603238865]],[[0.8545 |
| 5 | EVAL_RESULTS_2 | TEST_F1_SCORE.VAR":"n/a","parameters":"ENET_ALPHA=0.15;ENET_LAMBDA=0.02;"},{"TEST_F1_SCORE":[[[0.9372549019607843,0.854 |
| 6 | EVAL_RESULTS_3 | _F1_SCORE.MEAN":"n/a","TEST_F1_SCORE.VAR":"n/a","parameters":"ENET_ALPHA=0.7;ENET_LAMBDA=0.03167;"}],"version":"1.0"} |
| 7 | AIC | 37.30856034515221 |
| 8 | obj | 0.31083953713359036 |
| 9 | iter | 13 |
| 10 | method | cycle coordinate descent |
| 11 | solution status | Converged |

### Optimal Parameter

Rows (2) SQL

| | PARAM_NAME | INT_VALUE | DOUBLE_VALUE | STRING_VALUE |
|---|---|---|---|---|
| 1 | ENET_ALPHA | *NULL* | 0.7 | *NULL* |
| 2 | ENET_LAMBDA | *NULL* | 0.01 | *NULL* |

## Parameter Selection

Parameter selection will first generate a set of candidates. The candidates will be determined with a range or discrete set, plus a search method. Current supported search methods are grid search and random search. Both of them can be used over range or discrete set.

If users just provide one candidate by providing `ENET_LAMBDA` and `ENET_ALPHA` parameters, then parameter selection reduces to model evaluation.

For each candidate of hyper parameter, parameter selection needs sample the provided data to separate it as training data and validating data. With candidate and training data, it will start a training process to obtain one model, then use the validating data to evaluate the model with specified evaluation method. Above process may repeat multiple times and average the evaluation metrics to get a final metric. By comparing the final evaluation metric for each candidate, it will elect the best candidate.

### Parameters to Define Candidate Set

Current algorithm hyper-parameter is combination of `ENET_LAMBDA` and `ENET_ALPHA`. The search space can be specified as range or discrete set. The range is defined with a start value and an end value and an optional step value in the middle.

- ENET_LAMBDA_VALUES
- ENET_ALPHA_VALUES
- ENET_LAMBDA_RANGE
- ENET_ALPHA_RANGE

> **Note**
>
> Users can only select one way for both `ENET_LAMBDA` and `ENET_ALPHA`, either both use VALUES suffix parameter or use RANGE suffix.

Following parameter defines the way to decide the actual candidates within the given range or discrete set:

- PARAM_SEARCH_STRATEGY

- RANDOM_SEARCH_TIMES

There are two search ways supported within search space: grid search and random search. Both can be applied to range or discrete set. When grid search applies to discrete set, it equals to cartesian product with all the given set. When it applies to range, it starts from the start value and iterate for each step value specified in RANGES parameter until the end value. Random search can also apply to discrete set and range, it uses parameter `RANDOM_SEARCH_TIMES` to define the final candidate size. Note random search may generate duplicate candidates.

**Parameters to Control Sampling Behaviour**

- RESAMPLING_METHOD
- FOLD_NUM

Current implementation supports two methods: cross validation and bootstrap. Cross validation needs another parameter `FOLD_NUM` to define the fold number. Both methods also support stratified sampling when specifying value of parameter `RESAMPLING_METHOD` with prefix `stratified_`. Stratified sampling will take value distribution of dependent variable into consideration.

**Parameters to Define Validation Metrics**

- EVALUATION_METRIC

Currently there are four metrics supported in total:

- ACCURACY
- F1_SCORE
- AUC
- NLL

For each candidate and sampling data set for training and validation set, parameter `REPEAT_TIMES` controls how many times to repeat.

If you want to get deterministic results between different runs with exactly same result, parameter `SEED` can be used to reproduce the results.

For more details about this topic, refer to topic Model Evaluation and Parameter Selection [page 2227].

## Example for Model Evaluation and Parameter Selection

**Example: Model Evaluation**

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_MCLR_DATA_TBL;
CREATE COLUMN TABLE PAL_MCLR_DATA_TBL (
    "V1" VARCHAR (50),
    "V2" DOUBLE,
    "V3" INTEGER,
    "CATEGORY" INTEGER
);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',2.62,0,1);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',2.875,0,1);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('A',2.32,1,1);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('A',3.215,2,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',3.44,3,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',3.46,0,0);
```

```
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('A',3.57,1,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',3.19,2,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('A',3.15,3,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',3.44,0,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',3.44,1,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('A',4.07,3,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('A',3.73,1,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',3.78,2,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',5.25,2,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('A',5.424,3,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('A',5.345,0,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',2.2,1,1);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',1.615,2,1);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('A',1.835,0,1);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',2.465,3,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('A',3.52,1,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('A',3.435,0,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',3.84,2,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',3.845,3,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('A',1.935,1,1);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',2.14,0,1);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',1.513,1,1);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('A',3.17,3,1);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',2.77,0,1);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',3.57,0,1);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('A',2.78,3,1);
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL (
    "PARAM_NAME" NVARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR (1000)
);
INSERT INTO PAL_PARAMETER_TBL VALUES ('MAX_ITERATION',500,NULL,NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('PMML_EXPORT',1,NULL,NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('HAS_ID',0,NULL,NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('STANDARDIZE',1,NULL,NULL);
--model evaluation parameters
INSERT INTO PAL_PARAMETER_TBL VALUES ('RESAMPLING_METHOD', NULL, NULL, 'cv');
INSERT INTO PAL_PARAMETER_TBL VALUES ('EVALUATION_METRIC', NULL, NULL,
'F1_SCORE');
INSERT INTO PAL_PARAMETER_TBL VALUES ('FOLD_NUM', 2, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('REPEAT_TIMES', 2, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('SEED', 1, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('PROGRESS_INDICATOR_ID', NULL, NULL,
'TEST');
DROP TABLE PAL_MCLR_STATISTIC_TBL;
CREATE COLUMN TABLE PAL_MCLR_STATISTIC_TBL ("STAT_NAME"
NVARCHAR(256),"STAT_VALUE" NVARCHAR(1000));
DO BEGIN
lt_data = SELECT * FROM PAL_MCLR_DATA_TBL;
lt_para = SELECT * FROM PAL_PARAMETER_TBL;
CALL _SYS_AFL.PAL_MULTICLASS_LOGISTIC_REGRESSION
(:lt_data,:lt_para,lt_result,lt_pmml_model,lt_stat,lt_opt_para);
INSERT INTO PAL_MCLR_STATISTIC_TBL
SELECT * FROM :lt_stat;
END;
SELECT * FROM PAL_MCLR_STATISTIC_TBL;
```

### Expected Results

### Statistics

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | timeout | FALSE |
| 2 | TEST_1_F1_SCORE | 1, 0.805668 |
| 3 | TEST_2_F1_SCORE | 0.792207, 0.686274 |
| 4 | TEST_F1_SCORE.MEAN | 0.821037 |
| 5 | TEST_F1_SCORE.VAR | 0.0170853 |
| 6 | EVAL_RESULTS_1 | {"candidates":[{"TEST_F1_SCORE":[[1.0,0.8056680161943319],[0.7922077922077921,0.6862745098039216]],"TEST_F1_SCORE.MEA |
| 7 | AIC | 26.634071919175714 |
| 8 | obj | 0.25990737373712053 |
| 9 | iter | 10 |
| 10 | method | lbfgs |
| 11 | solution status | Converged |

## Example: Parameter Selection

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_MCLR_DATA_TBL;
CREATE COLUMN TABLE PAL_MCLR_DATA_TBL (
    "V1" VARCHAR (50),
    "V2" DOUBLE,
    "V3" INTEGER,
    "CATEGORY" INTEGER
);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',2.62,0,1);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',2.875,0,1);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('A',2.32,1,1);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('A',3.215,2,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',3.44,3,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',3.46,0,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('A',3.57,1,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',3.19,2,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('A',3.15,3,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',3.44,0,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',3.44,1,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('A',4.07,3,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('A',3.73,1,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',3.78,2,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',5.25,2,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('A',5.424,3,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('A',5.345,0,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',2.2,1,1);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',1.615,2,1);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('A',1.835,0,1);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',2.465,3,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('A',3.52,1,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('A',3.435,0,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',3.84,2,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',3.845,3,0);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('A',1.935,1,1);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',2.14,0,1);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',1.513,1,1);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('A',3.17,3,1);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',2.77,0,1);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('B',3.57,0,1);
INSERT INTO PAL_MCLR_DATA_TBL VALUES ('A',2.78,3,1);
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL (
    "PARAM_NAME" NVARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR (1000)
);
INSERT INTO PAL_PARAMETER_TBL VALUES ('MAX_ITERATION',500,NULL,NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('PMML_EXPORT',1,NULL,NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('HAS_ID',0,NULL,NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('STANDARDIZE',1,NULL,NULL);
--parameters to be selected
```

```
INSERT INTO PAL_PARAMETER_TBL VALUES ('ENET_LAMBDA_VALUES', NULL, NULL,
'{0.01,0.02,0.03167}');
INSERT INTO PAL_PARAMETER_TBL VALUES ('ENET_ALPHA_VALUES', NULL, NULL,
'{0.15,0.2,0.7}');
INSERT INTO PAL_PARAMETER_TBL VALUES ('RESAMPLING_METHOD', NULL, NULL, 'cv');
--parameter selection parameters
INSERT INTO PAL_PARAMETER_TBL VALUES ('EVALUATION_METRIC', NULL, NULL,
'F1_SCORE');
INSERT INTO PAL_PARAMETER_TBL VALUES ('FOLD_NUM', 2, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('REPEAT_TIMES', 2, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('SEED', 1, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('PARAM_SEARCH_STRATEGY', NULL, NULL,
'grid');
INSERT INTO PAL_PARAMETER_TBL VALUES ('PROGRESS_INDICATOR_ID', NULL, NULL,
'TEST');
DROP TABLE PAL_MCLR_STATISTIC_TBL;
CREATE COLUMN TABLE PAL_MCLR_STATISTIC_TBL ("STAT_NAME"
NVARCHAR(256),"STAT_VALUE" NVARCHAR(1000));
DROP TABLE PAL_MCLR_OPT_PARAM_TBL;
CREATE COLUMN TABLE PAL_MCLR_OPT_PARAM_TBL ("PARAM_NAME"
NVARCHAR(256),"INT_VALUE" INTEGER, "DOUBLE_VALUE" DOUBLE, "STRING_VALUE"
NVARCHAR(1000));
DO BEGIN
lt_data = SELECT * FROM PAL_MCLR_DATA_TBL;
lt_para = SELECT * FROM PAL_PARAMETER_TBL;
CALL _SYS_AFL.PAL_MULTICLASS_LOGISTIC_REGRESSION
(:lt_data,:lt_para,lt_result,lt_pmml_model,lt_stat,lt_opt_para);
INSERT INTO PAL_MCLR_STATISTIC_TBL
SELECT * FROM :lt_stat;
INSERT INTO PAL_MCLR_OPT_PARAM_TBL
SELECT * FROM :lt_opt_para;
END;
SELECT * FROM PAL_MCLR_STATISTIC_TBL;
SELECT * FROM PAL_MCLR_OPT_PARAM_TBL;
```

### Expected Results

### Statistics

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | timeout | FALSE |
| 2 | TEST_1_F1_SCORE | 0.930735, 0.805668 |
| 3 | TEST_2_F1_SCORE | 0.854545, 0.935222 |
| 4 | TEST_F1_SCORE.MEAN | 0.881543 |
| 5 | TEST_F1_SCORE.VAR | 0.00392911 |
| 6 | EVAL_RESULTS_1 | {"candidates":[{"TEST_F1_SCORE":[[0.9307359307359306,0.8056680161943319],[0.7922077922077921,0.8730158730158730]],"TES |
| 7 | EVAL_RESULTS_2 | 7922077921,0.9352226720647774]],"TEST_F1_SCORE.MEAN":0.8659586028007080,"TEST_F1_SCORE.VAR":0.00602258346729942 |
| 8 | EVAL_RESULTS_3 | _F1_SCORE.VAR":0.003929110751817133,"parameters":"ENET_ALPHA=0.7;ENET_LAMBDA=0.03167;"}],"version":"1.0"} |
| 9 | AIC | 38.14999184799123 |
| 10 | obj | 0.34534638048288435 |
| 11 | iter | 11 |
| 12 | method | cycle coordinate descent |
| 13 | solution status | Converged |

### Optimal Parameter

| | PARAM_NAME | INT_VALUE | DOUBLE_VALUE | STRING_VALUE |
|---|---|---|---|---|
| 1 | ENET_ALPHA | *NULL* | 0.7 | *NULL* |
| 2 | ENET_LAMBDA | *NULL* | 0.02 | *NULL* |

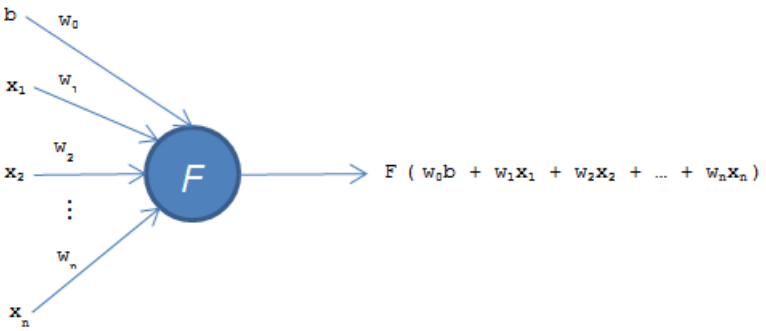
## 5.2.9 Multilayer Perceptron

A multilayer perceptron (MLP) is a class of feed forward artificial neural network.

Neural network is a calculation model inspired by biological nervous system. The functionality of neural network is determined by its network structure and connection weights between neurons. MLP has at least 3 layers with first layer and last layer called input layer and output layer accordingly. Each layer is fully connected with its preceding and succeeding layers if they exist. It is trained using a technique called back propagation.

### Single Neuron

A neuron takes signals from outside and transforms them to a single value as output:

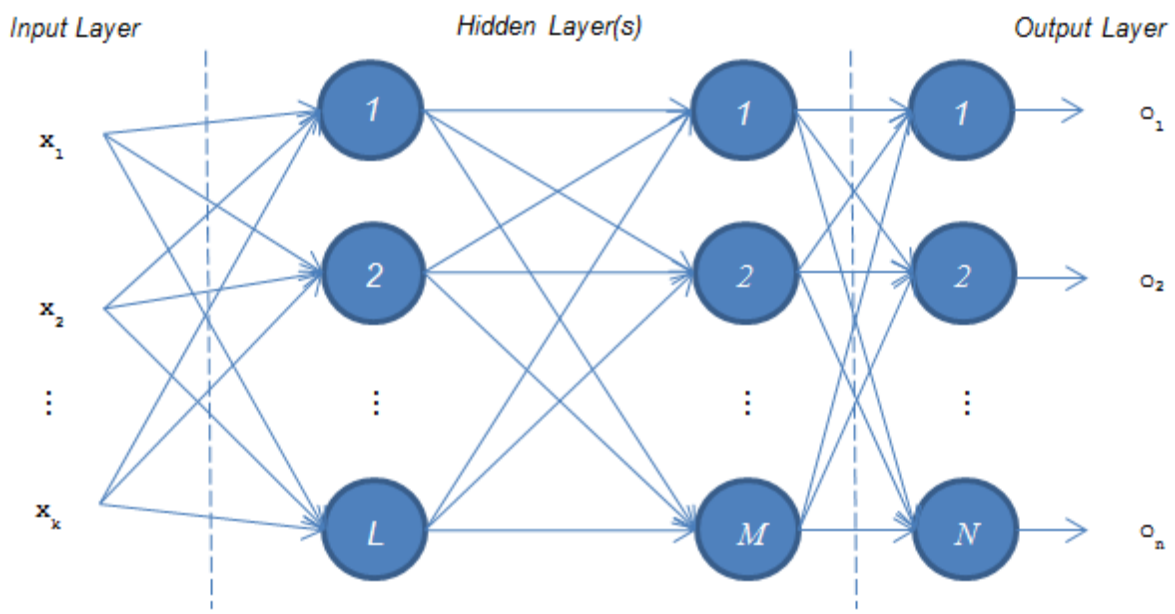



The function F is called activate function.

- $x_i$ ( i = 1, 2, …, n ) is the outside signal and b is a constant bias usually set to 1.
- $w_i$ ( i = 0, 1, …, n ) is the weight value on each connection.

### Neural Network Structure

A neural network consists of three parts: input layer, hidden layer(s), and output layer. Each layer owns several neurons and there are connections between layers. Signals are given in input layer and transmit through connections and layers. Finally output layer gives transformed signal.




### Training

The following training styles are available:

- Batch training: weights updating is based on the error of the entire package of training patterns. Thus, in one round the weights are updated once.
- Stochastic training: weights updating is based on the error of a single training pattern. Thus, in one round the weights are updated for each pattern.

Training log is outputted if the corresponding training log table is provided through the function call. Currently the training log contains the mean squared error between predicted values and target values for each iteration.

### Support for Categorical Attributes

If an attribute is of category type, it will be converted to a binary vector and then be used as numerical attributes. For example, in the below table, "Gender" is of category type.

| Customer ID | Age | Income | Gender |
|---|---|---|---|
| T1 | 31 | 10,000 | Female |
| T2 | 27 | 8,000 | Male |

Because "Gender" has two distinct values, it will be converted into a binary vector with two dimensions:

| Customer ID | Age | Income | Gender_1 | Gender_2 |
|---|---|---|---|---|
| T1 | 31 | 10,000 | 0 | 1 |
| T2 | 27 | 8,000 | 1 | 0 |

## Prerequisites

For training data:

- Data cannot contain null value. Otherwise the algorithm will issue errors.
- If it is for classification, there can only be 1 dependent variable. Dependent variable should be specified or the last column will be the one.
- If it is for regression, there can be multiple dependent variables. Dependent variables should be specified or the last column will be the only one.

For predicted data:

- Data cannot contain null value. Otherwise the algorithm will issue errors.
- The first column is ID column.
- The column order, column number, and column names of the predicted data should be the same as the order, number, and names used in model training.

## _SYS_AFL.PAL_MULTILAYER_PERCEPTRON

This function trains a MLP model using back propagation.

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<MODEL table>` | OUT |
| 4 | `<TRAINING_LOG table>` | OUT |
| 5 | `<STATISTICS table>` | OUT |
| 6 | `<OPTIMAL_PARAM table>` | OUT |

### Input Table

| Table | Column | Data Type | Description |
|---|---|---|---|
| DATA | 1st column | INTEGER, BIGINT, VARCHAR or NVARCHAR | Record ID. If this column is absent, the `HAS_ID` parameter must be set to 0. |

| Table | Column | Data Type | Description |
|---|---|---|---|
| | Other columns | Attribute data type should be INTEGER, DOUBLE, DECIMAL(p,s), VARCHAR, or NVARCHAR. For classification, dependency variable data type should be INTEGER, VARCHAR or NVARCHAR. For regression dependency variable(s) data type should be INTEGER, DOUBLE, or DECIMAL(p,s). | Attribute data, and dependency variable(s). |

### Parameter Table

Mandatory Parameters

The following parameters are mandatory and must be given a value.

| Name | Data Type | Description |
|---|---|---|
| HIDDEN_LAYER_ACTIVE_FUNC | INTEGER | Active function code for the hidden layer.<br>• TANH = 1<br>• LINEAR = 2<br>• SIGMOID_ASYMMETRIC = 3<br>• SIGMOID_SYMMETRIC = 4<br>• GAUSSIAN_ASYMMETRIC = 5<br>• GAUSSIAN_SYMMETRIC = 6<br>• ELLIOT_ASYMMETRIC = 7<br>• ELLIOT_SYMMETRIC = 8<br>• SIN_ASYMMETRIC = 9<br>• SIN_SYMMETRIC = 10<br>• COS_ASYMMETRIC = 11<br>• COS_SYMMETRIC = 12<br>• ReLU = 13 |
| OUTPUT_LAYER_ACTIVE_FUNC | INTEGER | Active function code for the output layer. The code is the same as `HIDDEN_LAYER_ACTIVE_FUNC`. |
| HIDDEN_LAYER_SIZE | NVARCHAR | Specifies the size of each hidden layer in the format of “2, 3, 4”. The value 0 will be ignored, for example, “2, 0, 3” is equal to “2, 3”. |

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| HAS_ID | INTEGER | 0 | Specifies whether the first column is ID column. | |
| MAX_ITERATION | INTEGER | 100 | Maximum number of iterations. | |
| FUNCTIONALITY | INTEGER | 0 | Specifies the prediction type:<br>• 0: Classification<br>• 1: Regression | |
| TRAINING_STYLE | INTEGER | 1 | Specifies the training style:<br>• 0: Batch<br>• 1: Stochastic | |
| LEARNING_RATE | DOUBLE | None | Specifies the learning rate. | Mandatory and valid only when TRAINING_STYLE = 1. |
| MOMENTUM_FACTOR | DOUBLE | None | Specifies the momentum factor. | Mandatory and valid only when TRAINING_STYLE = 1. |
| MINI_BATCH_SIZE | INTEGER | 1 | Specifies the size of mini batch. | Valid only when TRAINING_STYLE = 1. |
| NORMALIZATION | INTEGER | 0 | Specifies the normalization type:<br>• 0: None<br>• 1: Z-transform<br>• 2: Scalar | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| WEIGHT_INIT | INTEGER | 0 | Specifies the weight initial value:<br>• 0: All zeros<br>• 1: Normal distribution<br>• 2: Uniform distribution in range (0, 1)<br>• 3: Variance scale normal<br>• 4: Variance scale uniform | |
| DEPENDENT_VARIABLE | NVARCHAR | No default value | Specifies dependent variable by name. For classification, only one can be specified. For regression, multiple ones can be specified. If no column is specified, the last one is considered to be the only one for both classification and regression. | |
| CATEGORICAL_VARIABLE | NVARCHAR | No default value | Indicates whether or not a column data is actually corresponding to a category variable even the data type of this column is INTEGER.<br>By default, 'VARCHAR' or 'NVARCHAR' is category variable, and 'INTEGER' or 'DOUBLE' is continuous variable. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| THREAD_RATIO | DOUBLE | 0 | Specifies the ratio of total number of threads that can be used by this function. The range of this parameter is from 0 to 1, where 0 means only using 1 thread, and 1 means using at most all the currently available threads. Values outside the range are ignored and this function heuristically determines the number of threads to use. | |

### Model Evaluation and Parameter Selection

The following parameters are only valid when model evaluation or parameter selection is enabled.

For more information, see the *Model Evaluation and Parameter Selection* topic.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| RESAMPLING_METHOD | NVARCHAR | No default value | Specifies the resampling method for model evaluation or parameter selection.<br>• cv<br>• cv_sha<br>• cv_hyperband<br>• bootstrap<br>• bootstrap_sha<br>• bootstrap_hyperband<br>• stratified_bootstrap<br>• stratified_bootstrap_sha<br>• stratified_bootstrap_hyperban<br>• stratified_cv<br>• stratified_cv_sha<br>• stratified_cv_hyperband<br>stratified_* can only apply to classification algorithms. If no value is specified for this parameter, neither model evaluation nor parameter selection is activated. | stratified_cv and stratified_bootstrap only apply when FUNCTIONALITY = 0 |
| EVALUATION_METRIC | NVARCHAR | No default value | Specifies the evaluation metric for model evaluation or parameter selection.<br>• ACCURACY<br>• F1_SCORE<br>• AUC_1VsRest<br>• AUC_pairwise<br>• RMSE | ACCURACY, F1_SCORE, AUC_1VsRest, and AUC_pairwise can only apply when FUNCTIONALITY = 0; while RMSE only apply when FUNCTIONALITY = 1 |
| FOLD_NUM | INTEGER | No default value | Specifies the fold number for the cross validation method. | Mandatory and valid only when RESAMPLING_METHOD is set to cv or stratified_cv. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| REPEAT_TIMES | INTEGER | 1 | Specifies the number of repeat times for re-sampling. | |
| PARAM_SEARCH_STRATEGY | NVARCHAR | No default value | Specifies the method to activate parameter selection.<br>• grid<br>• random | When *_sha is specified in RESAMPLING_METHOD, PARAM_SEARCH_STRATEGY is mandatory. When *_hyperband is specified in RESAMPLING_METHOD, PARAM_SEARCH_STRATEGY is set to random whatever other value is provided. |
| RANDOM_SEARCH_TIMES | INTEGER | No default value | Specifies the number of times to randomly select candidate parameters for selection. | Mandatory and valid when PARAM_SEARCH_STRATEGY is set to random. |
| REDUCTION_RATE | DOUBLE | 3.0 | Specifies reduction rate in SHA or hyperband method. | Only valid when RESAMPLING_METHOD is specified as *_sha or *_hyberband. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| AGGRESSIVE_ELIMINATION | INTEGER | | Specifies whether to apply aggressive elimination while using SHA method. It happens when the size of data and that of parameters to be searched does not match. And there are still bunch of parameters to be searched while data size reaches its upper limits. If aggressive elimination is applied, lower bound of limit of data size will be used multiple times first to reduce number of parameters.<br>• 0: not apply<br>• 1: apply | Only valid when RESAMPLING_METHOD is specified as *_sha. |
| SEED | INTEGER | 0 | Specifies the seed for random generation. Use system time when 0 is specified. | |
| TIMEOUT | INTEGER | 0 | Specifies maximum running time for model evaluation or parameter selection, in seconds. No timeout when 0 is specified. | |
| PROGRESS_INDICATOR_ID | NVARCHAR | No default value | Sets an ID of progress indicator for model evaluation or parameter selection. No progress indicator is active if no value is provided. | |
| HIDDEN_LAYER_ACTIVE_FUNC_VALUES | NVARCHAR | No default value | Specifies values of HIDDEN_LAYER_ACTIVE_FUNC to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| OUTPUT_LAYER_ACTIVE_FUNC_VALUES | NVARCHAR | No default value | Specifies values of OUTPUT_LAYER_ACTIVE_FUNC to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| HIDDEN_LAYER_SIZE_VALUES | NVARCHAR | No default value | Specifies values of HIDDEN_LAYER_SIZE to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| LEARNING_RATE_VALUES | NVARCHAR | No default value | Specifies values of LEARNING_RATE to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified and TRAINING_STYLE = 1. |
| LEARNING_RATE_RANGE | NVARCHAR | No default value | Specifies the range of LEARNING_RATE to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified and TRAINING_STYLE = 1. |
| MOMENTUM_FACTOR_VALUES | NVARCHAR | No default value | Specifies values of MOMENTUM_FACTOR to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified and TRAINING_STYLE = 1. |
| MOMENTUM_FACTOR_RANGE | NVARCHAR | No default value | Specifies the range of MOMENTUM_FACTOR to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified and TRAINING_STYLE = 1. |
| MINI_BATCH_SIZE_VALUES | NVARCHAR | No default value | Specifies values of MINI_BATCH_SIZE to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified and TRAINING_STYLE = 1. |
| MINI_BATCH_SIZE_RANGE | NVARCHAR | No default value | Specifies the range of MINI_BATCH_SIZE to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified and TRAINING_STYLE = 1. |

### Output Tables

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| MODEL | 1st column | INTEGER | ROW_INDEX | Model row index |

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| | 2nd column | VARCHAR or NVARCHAR | MODEL_CONTENT | Saved model. The table must be a column table. The minimum length of every unit (row) is 5000. |
| TRAINING LOG | 1st column | INTEGER | ITERATION | Iteration number |
| | 2nd column | DOUBLE | ERROR | Mean squared error between predicted values and target values |
| STATISTICS | 1st column | VARCHAR or NVARCHAR | STAT_NAME | Statistic name |
| | 2nd column | VARCHAR or NVARCHAR | STAT_VALUE | Statistic value |
| OPTIMAL_PARAM | 1st column | NVARCHAR | PARAM_NAME | Optimal parameters selected |
| | 2nd column | INTEGER | INT_VALUE | |
| | 3rd column | DOUBLE | DOUBLE_VALUE | |
| | 4th column | NVARCHAR | STRING_VALUE | |

### Examples

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

**Classification example:**

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_TRAIN_MLP_CLS_DATA_TBL;
CREATE COLUMN TABLE PAL_TRAIN_MLP_CLS_DATA_TBL(
    "V000" INTEGER,
    "V001" DOUBLE,
    "V002" VARCHAR(10),
    "V003" INTEGER,
    "LABEL" VARCHAR(2)
);
INSERT INTO PAL_TRAIN_MLP_CLS_DATA_TBL VALUES (1, 1.71, 'AC', 0, 'AA');
INSERT INTO PAL_TRAIN_MLP_CLS_DATA_TBL VALUES (10, 1.78, 'CA', 5, 'AB');
INSERT INTO PAL_TRAIN_MLP_CLS_DATA_TBL VALUES (17, 2.36, 'AA', 6, 'AA');
INSERT INTO PAL_TRAIN_MLP_CLS_DATA_TBL VALUES (12, 3.15, 'AA', 2, 'C');
INSERT INTO PAL_TRAIN_MLP_CLS_DATA_TBL VALUES (7, 1.05, 'CA', 3, 'AB');
INSERT INTO PAL_TRAIN_MLP_CLS_DATA_TBL VALUES (6, 1.50, 'CA', 2, 'AB');
INSERT INTO PAL_TRAIN_MLP_CLS_DATA_TBL VALUES (9, 1.97, 'CA', 6, 'C');
INSERT INTO PAL_TRAIN_MLP_CLS_DATA_TBL VALUES (5, 1.26, 'AA', 1, 'AA');
INSERT INTO PAL_TRAIN_MLP_CLS_DATA_TBL VALUES (12, 2.13, 'AC', 4, 'C');
INSERT INTO PAL_TRAIN_MLP_CLS_DATA_TBL VALUES (18, 1.87, 'AC', 6, 'AA');
```

```
DROP TABLE PAL_MLP_CLS_MODEL_TBL;
CREATE COLUMN TABLE PAL_MLP_CLS_MODEL_TBL(
    "ROW_INDEX" INTEGER,
    "MODEL_CONTENT" NVARCHAR(5000)
);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL(
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR(1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('HIDDEN_LAYER_SIZE', NULL, NULL, '10,
10');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('HIDDEN_LAYER_ACTIVE_FUNC', 1, NULL,
NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('OUTPUT_LAYER_ACTIVE_FUNC', 1, NULL,
NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('LEARNING_RATE', NULL, 0.001, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MOMENTUM_FACTOR', NULL, 0.00001, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FUNCTIONALITY', 0, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('TRAINING_STYLE', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('CATEGORICAL_VARIABLE', NULL, NULL,
'V003');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('DEPENDENT_VARIABLE', NULL, NULL,
'LABEL');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MAX_ITERATION', 100, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('NORMALIZATION', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('WEIGHT_INIT', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 0.3, NULL);
-- call procedure
CALL _SYS_AFL.PAL_MULTILAYER_PERCEPTRON(PAL_TRAIN_MLP_CLS_DATA_TBL,
"#PAL_PARAMETER_TBL", PAL_MLP_CLS_MODEL_TBL, ?, ?, ?) WITH OVERVIEW;
```

### Expected Results

Your result may look different from the following results due to randomness.

MODEL:

| | ROW_INDEX | MODEL_CONTENT |
|---|---|---|
| 1 | 1 | {"CurrentVersion":"1.0","DataDictionary":[{"dataType":"intege... |
| 2 | 2 | 74291869},{"from":11,"weight":0.2091898792826473}],"bias... |
| 3 | 3 | 693502}],"bias":-0.6303833323329159,"id":27},{"Connection... |

LOG:

| | ITERATION | ERROR |
|---|---|---|
| 1 | 1 | 1.0748755402711112 |
| 2 | 2 | 1.0061572430118226 |
| 3 | 3 | 0.9452923206997197 |
| 4 | 4 | 0.8912162314729025 |
| 5 | 5 | 0.84303448460841 |
| 6 | 6 | 0.7999856724728426 |
| 7 | 7 | 0.7614172860853101 |
| 8 | 8 | 0.7267681972252736 |
| 9 | 9 | 0.6955549272766962 |
| 10 | 10 | 0.6673603401632373 |
| 11 | 11 | 0.6418240907680317 |
| 12 | 12 | 0.618634469252952 |
| 13 | 13 | 0.5975214165596662 |
| 14 | 14 | 0.5782505459479135 |
| 15 | 15 | 0.5606180343585689 |
| 16 | 16 | 0.5444462648177942 |
| 17 | 17 | 0.5295801145960346 |
| 18 | 18 | 0.5158837960041088 |

STATISTICS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | ERROR | 0.2741494239585351 |

**Regression example:**

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_TRAIN_MLP_REG_DATA_TBL;
CREATE COLUMN TABLE PAL_TRAIN_MLP_REG_DATA_TBL(
    "V000" INTEGER,
    "V001" DOUBLE,
    "V002" VARCHAR(10),
    "V003" INTEGER,
    "T001" DOUBLE,
    "T002" DOUBLE,
    "T003" DOUBLE
);
INSERT INTO PAL_TRAIN_MLP_REG_DATA_TBL VALUES(1, 1.71, 'AC', 0, 12.7, 2.8, 3.06);
INSERT INTO PAL_TRAIN_MLP_REG_DATA_TBL VALUES(10, 1.78, 'CA', 5, 12.1, 8.0,
2.65);
INSERT INTO PAL_TRAIN_MLP_REG_DATA_TBL VALUES(17, 2.36, 'AA', 6, 10.1, 2.8,
3.24);
```

```
INSERT INTO PAL_TRAIN_MLP_REG_DATA_TBL VALUES(12, 3.15, 'AA', 2, 28.1, 5.6,
2.24);
INSERT INTO PAL_TRAIN_MLP_REG_DATA_TBL VALUES(7, 1.05, 'CA', 3, 19.8, 7.1, 1.98);
INSERT INTO PAL_TRAIN_MLP_REG_DATA_TBL VALUES(6, 1.50, 'CA', 2, 23.2, 4.9, 2.12);
INSERT INTO PAL_TRAIN_MLP_REG_DATA_TBL VALUES(9, 1.97, 'CA', 6, 24.5, 4.2, 1.05);
INSERT INTO PAL_TRAIN_MLP_REG_DATA_TBL VALUES(5, 1.26, 'AA', 1, 13.6, 5.1, 2.78);
INSERT INTO PAL_TRAIN_MLP_REG_DATA_TBL VALUES(12, 2.13, 'AC', 4, 13.2, 1.9,
1.34);
INSERT INTO PAL_TRAIN_MLP_REG_DATA_TBL VALUES(18, 1.87, 'AC', 6, 25.5, 3.6,
2.14);
DROP TABLE PAL_MLP_REG_MODEL_TBL;
CREATE COLUMN TABLE PAL_MLP_REG_MODEL_TBL(
    "ROW_INDEX" INTEGER,
    "MODEL_CONTENT" NVARCHAR(5000)
);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL(
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR(1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 0.3, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('HIDDEN_LAYER_SIZE', NULL, NULL, '10, 5');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('HIDDEN_LAYER_ACTIVE_FUNC', 9, NULL,
NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('OUTPUT_LAYER_ACTIVE_FUNC', 9, NULL,
NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('LEARNING_RATE', NULL, 0.001, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MOMENTUM_FACTOR', NULL, 0.00001, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FUNCTIONALITY', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('TRAINING_STYLE', 0, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('DEPENDENT_VARIABLE', NULL, NULL, 'T001');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('DEPENDENT_VARIABLE', NULL, NULL, 'T002');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('DEPENDENT_VARIABLE', NULL, NULL, 'T003');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MAX_ITERATION', 10000, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('NORMALIZATION', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('WEIGHT_INIT', 1, NULL, NULL);
-- call procedure
CALL _SYS_AFL.PAL_MULTILAYER_PERCEPTRON(PAL_TRAIN_MLP_REG_DATA_TBL,
"#PAL_PARAMETER_TBL", PAL_MLP_REG_MODEL_TBL, ?, ?, ?) WITH OVERVIEW;
```

### Expected Results

Your result may look different from the following results due to randomness.

MODEL:

| | ROW_INDEX | MODEL_CONTENT |
|---|---|---|
| 1 | 1 | {"CurrentVersion":"1.0","DataDictionary":[{"dataType":"intege... |
| 2 | 2 | 9383361847306},{"from":10,"weight":-0.358472575000410... |

LOG:

| | ITERATION | ERROR |
|---|---|---|
| 1 | 1 | 0.2912169375465381 |
| 2 | 2 | 0.1991493481312584 |
| 3 | 3 | 0.18435120055903... |
| 4 | 4 | 0.14817513651537... |
| 5 | 5 | 0.147096756140456 |
| 6 | 6 | 0.14600990979843... |
| 7 | 7 | 0.14523675370943... |
| 8 | 8 | 0.14025989001197... |
| 9 | 9 | 0.13071302838447... |
| 10 | 10 | 0.12120908351425... |
| 11 | 11 | 0.10880923142128... |
| 12 | 12 | 0.1045563792297178 |
| 13 | 13 | 0.09329807518227... |
| 14 | 14 | 0.08772500291759... |
| 15 | 15 | 0.08142355867994... |
| 16 | 16 | 0.07792150753708... |

STATISTICS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | ERROR | 0.00000011868586453805247 |

**Model evaluation example:**

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_TRAIN_MLP_CLS_DATA_TBL;
CREATE COLUMN TABLE PAL_TRAIN_MLP_CLS_DATA_TBL(
    "V000" INTEGER,
    "V001" DOUBLE,
    "V002" VARCHAR(10),
    "V003" INTEGER,
    "LABEL" VARCHAR(2)
);
INSERT INTO PAL_TRAIN_MLP_CLS_DATA_TBL VALUES (1, 1.71, 'AC', 0, 'AA');
INSERT INTO PAL_TRAIN_MLP_CLS_DATA_TBL VALUES (10, 1.78, 'CA', 5, 'AB');
INSERT INTO PAL_TRAIN_MLP_CLS_DATA_TBL VALUES (17, 2.36, 'AA', 6, 'AA');
INSERT INTO PAL_TRAIN_MLP_CLS_DATA_TBL VALUES (12, 3.15, 'AA', 2, 'C');
INSERT INTO PAL_TRAIN_MLP_CLS_DATA_TBL VALUES (7, 1.05, 'CA', 3, 'AB');
INSERT INTO PAL_TRAIN_MLP_CLS_DATA_TBL VALUES (6, 1.50, 'CA', 2, 'AB');
INSERT INTO PAL_TRAIN_MLP_CLS_DATA_TBL VALUES (9, 1.97, 'CA', 6, 'C');
INSERT INTO PAL_TRAIN_MLP_CLS_DATA_TBL VALUES (5, 1.26, 'AA', 1, 'AA');
INSERT INTO PAL_TRAIN_MLP_CLS_DATA_TBL VALUES (12, 2.13, 'AC', 4, 'C');
```

```
INSERT INTO PAL_TRAIN_MLP_CLS_DATA_TBL VALUES (18, 1.87, 'AC', 6, 'AA');
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL(
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR(1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('HIDDEN_LAYER_SIZE', NULL, NULL, '10,
10');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('HIDDEN_LAYER_ACTIVE_FUNC', 1, NULL,
NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('OUTPUT_LAYER_ACTIVE_FUNC', 1, NULL,
NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('LEARNING_RATE', NULL, 0.001, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MOMENTUM_FACTOR', NULL, 0.00001, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FUNCTIONALITY', 0, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('TRAINING_STYLE', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('CATEGORICAL_VARIABLE', NULL, NULL,
'V003');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('DEPENDENT_VARIABLE', NULL, NULL,
'LABEL');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MAX_ITERATION', 100, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('NORMALIZATION', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('WEIGHT_INIT', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 0.3, NULL);
-- model evaluation parameters
INSERT INTO #PAL_PARAMETER_TBL VALUES ('RESAMPLING_METHOD', NULL, NULL, 'cv');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('EVALUATION_METRIC', NULL, NULL,
'F1_SCORE');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FOLD_NUM', 10, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('REPEAT_TIMES', 2, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SEED', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PROGRESS_INDICATOR_ID', NULL, NULL,
'TEST');
-- call procedure
CALL _SYS_AFL.PAL_MULTILAYER_PERCEPTRON(PAL_TRAIN_MLP_CLS_DATA_TBL,
"#PAL_PARAMETER_TBL", ?, ?, ?, ?);
```

**Expected Result**

Your result may look different from the following results due to randomness.

STATISTICS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | timeout | FALSE |
| 2 | TEST_1_F1_SCORE | 0, 0, 1, 0, 0, 1, 0, 0, 0, 0 |
| 3 | TEST_2_F1_SCORE | 0, 0, 1, 0, 0, 0, 0, 0, 1, 0 |
| 4 | TEST_F1_SCORE.MEAN | 0.2 |
| 5 | TEST_F1_SCORE.VAR | 0.168421 |
| 6 | EVAL_RESULTS_1 | {"candidates":[{"TEST_F1_SCORE":[[0.0,0.0,1.0,0.... |
| 7 | solution status | Convergence not reached after maximum num... |
| 8 | ERROR | 0.30593245718906725 |

**Parameter selection example:**

```
SET SCHEMA DM_PAL;
```

```
DROP TABLE PAL_TRAIN_MLP_CLS_DATA_TBL;
CREATE COLUMN TABLE PAL_TRAIN_MLP_CLS_DATA_TBL(
    "V000" INTEGER,
    "V001" DOUBLE,
    "V002" VARCHAR(10),
    "V003" INTEGER,
    "LABEL" VARCHAR(2)
);
INSERT INTO PAL_TRAIN_MLP_CLS_DATA_TBL VALUES (1, 1.71, 'AC', 0, 'AA');
INSERT INTO PAL_TRAIN_MLP_CLS_DATA_TBL VALUES (10, 1.78, 'CA', 5, 'AB');
INSERT INTO PAL_TRAIN_MLP_CLS_DATA_TBL VALUES (17, 2.36, 'AA', 6, 'AA');
INSERT INTO PAL_TRAIN_MLP_CLS_DATA_TBL VALUES (12, 3.15, 'AA', 2, 'C');
INSERT INTO PAL_TRAIN_MLP_CLS_DATA_TBL VALUES (7, 1.05, 'CA', 3, 'AB');
INSERT INTO PAL_TRAIN_MLP_CLS_DATA_TBL VALUES (6, 1.50, 'CA', 2, 'AB');
INSERT INTO PAL_TRAIN_MLP_CLS_DATA_TBL VALUES (9, 1.97, 'CA', 6, 'C');
INSERT INTO PAL_TRAIN_MLP_CLS_DATA_TBL VALUES (5, 1.26, 'AA', 1, 'AA');
INSERT INTO PAL_TRAIN_MLP_CLS_DATA_TBL VALUES (12, 2.13, 'AC', 4, 'C');
INSERT INTO PAL_TRAIN_MLP_CLS_DATA_TBL VALUES (18, 1.87, 'AC', 6, 'AA');
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL(
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR(1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('LEARNING_RATE', NULL, 0.001, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MOMENTUM_FACTOR', NULL, 0.00001, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FUNCTIONALITY', 0, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('TRAINING_STYLE', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('CATEGORICAL_VARIABLE', NULL, NULL,
'V003');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('DEPENDENT_VARIABLE', NULL, NULL,
'LABEL');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MAX_ITERATION', 100, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('NORMALIZATION', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('WEIGHT_INIT', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 0.3, NULL);
-- parameters to be selected
INSERT INTO #PAL_PARAMETER_TBL VALUES ('HIDDEN_LAYER_SIZE_VALUES', NULL, NULL,
'{"10, 10", "5, 5, 5"}');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('HIDDEN_LAYER_ACTIVE_FUNC_VALUES', NULL,
NULL, '{1, 2, 3}');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('OUTPUT_LAYER_ACTIVE_FUNC_VALUES', NULL,
NULL, '{4, 5, 6}');
-- parameter selection parameters
INSERT INTO #PAL_PARAMETER_TBL VALUES ('RESAMPLING_METHOD', NULL, NULL,
'stratified_bootstrap');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('EVALUATION_METRIC', NULL, NULL,
'ACCURACY');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('REPEAT_TIMES', 2, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PARAM_SEARCH_STRATEGY', NULL, NULL,
'grid');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SEED', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PROGRESS_INDICATOR_ID', NULL, NULL,
'TEST');
-- call procedure
CALL _SYS_AFL.PAL_MULTILAYER_PERCEPTRON(PAL_TRAIN_MLP_CLS_DATA_TBL,
"#PAL_PARAMETER_TBL", ?, ?, ?, ?);
```

### Expected Result

Your result may look different from the following results due to randomness.

STATISTICS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | timeout | FALSE |
| 2 | TEST_1_ACCURACY | 0.5 |
| 3 | TEST_2_ACCURACY | 0.333333 |
| 4 | TEST_ACCURACY.MEAN | 0.416666 |
| 5 | TEST_ACCURACY.VAR | 0.0138888 |
| 6 | EVAL_RESULTS_1 | {"candidates":[{"TEST_ACCURACY":[... |
| 7 | EVAL_RESULTS_2 | ,"TEST_ACCURACY.VAR":0.003472... |
| 8 | EVAL_RESULTS_3 | T_ACCURACY":[[0.250],[0.3333333... |
| 9 | EVAL_RESULTS_4 | :0.003472222222222221,"paramet... |
| 10 | solution status | Convergence not reached after ma... |
| 11 | ERROR | 0.3348325831840434 |

OPTIMAL PARAMETERS:

| | PARAM_NAME | INT_VALUE | DOUBLE_VALUE | STRING_VALUE |
|---|---|---|---|---|
| 1 | HIDDEN_LAYER_SIZE | ? | ? | 10, 10 |
| 2 | OUTPUT_LAYER_ACTIVE_FUNC | 4 | ? | ? |
| 3 | HIDDEN_LAYER_ACTIVE_FUNC | 1 | ? | ? |

## _SYS_AFL.PAL_MULTILAYER_PERCEPTRON_PREDICT

This function makes prediction with a trained MLP model.

**Overview of All Tables**

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<MODEL table>` | IN |
| 3 | `<PARAMETER table>` | IN |
| 4 | `<RESULT table>` | OUT |

| Position | Table Name | Parameter Type |
|---|---|---|
| 5 | `<SOFTMAX table>` | OUT |

### Input Tables

| Table | Column | Reference | Data Type | Description |
|---|---|---|---|---|
| DATA | 1st column | ID | INTEGER, BIGINT, VARCHAR or NVARCHAR | Record ID. |
| | Other columns | | INTEGER, DOUBLE, DECIMAL(p,s), VARCHAR, or NVARCHAR | Attribute data to be predicted. |
| MODEL | 1st column | | INTEGER | Row index. |
| | 2nd column | | VARCHAR or NVARCHAR | Saved model. |

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| THREAD_RATIO | DOUBLE | 0 | Specifies the ratio of total number of threads that can be used by this function. The range of this parameter is from 0 to 1, where 0 means only using 1 thread, and 1 means using at most all the currently available threads. Values outside this range are ignored and this function heuristically determines the number of threads to use. |

### Output Tables

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| RESULT | 1st column | ID data type | ID column name | Record ID. |
| | 2nd column | NVARCHAR | TARGET | For classification, it is the class name. For regression, it is the target name. |
| | 3rd column | DOUBLE | VALUE | For classification, it is the SoftMax value for that class. For regression, it is the regression value. |
| SOFTMAX (only used for classification) | 1st column | ID data type | ID column name | Record ID. |
| | 2nd column | NVARCHAR | CLASS | Class name. |
| | 3rd column | DOUBLE | SOFTMAX_VALUE | SoftMax value. |

### Examples

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.
- The `PAL_MLP_CLS_MODEL_TBL` and `PAL_MLP_REG_MODEL_TBL` tables store the trained MLP model under schema DM_PAL.

**Classification example:**

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_PREDICT_MLP_CLS_DATA_TBL;
CREATE COLUMN TABLE PAL_PREDICT_MLP_CLS_DATA_TBL(
    "ID" INTEGER,
    "V000" INTEGER,
    "V001" DOUBLE,
    "V002" VARCHAR(10),
    "V003" INTEGER
);
INSERT INTO PAL_PREDICT_MLP_CLS_DATA_TBL VALUES (1, 3, 1.91, 'AC', 3);
INSERT INTO PAL_PREDICT_MLP_CLS_DATA_TBL VALUES (2, 7, 2.18, 'CA', 6);
INSERT INTO PAL_PREDICT_MLP_CLS_DATA_TBL VALUES (3, 11, 1.96, 'AA', 8);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL(
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR(1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 0.2, NULL);
CALL _SYS_AFL.PAL_MULTILAYER_PERCEPTRON_PREDICT(PAL_PREDICT_MLP_CLS_DATA_TBL,
PAL_MLP_CLS_MODEL_TBL, "#PAL_PARAMETER_TBL", ?, ?);
```

### Expected Result

Your result may look different from the following results due to model randomness.

RESULT:

| | ID | TARGET | VALUE |
|---|---|---|---|
| 1 | 1 | AB | 0.506863978638649 |
| 2 | 2 | AA | 0.3843203815104004 |
| 3 | 3 | C | 0.35952881905282... |

SOFTMAX:

| | ID | CLASS | SOFTMAX_VALUE |
|---|---|---|---|
| 1 | 1 | AA | 0.16554720896831115 |
| 2 | 1 | AB | 0.506863978638649 |
| 3 | 1 | C | 0.32758881239303983 |
| 4 | 2 | AA | 0.3843203815104004 |
| 5 | 2 | AB | 0.31841078305914366 |
| 6 | 2 | C | 0.2972688354304559 |
| 7 | 3 | AA | 0.31946746996261194 |
| 8 | 3 | AB | 0.3210037109845597 |
| 9 | 3 | C | 0.35952881905282835 |

**Regression example:**

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_PREDICT_MLP_REG_DATA_TBL;
CREATE COLUMN TABLE PAL_PREDICT_MLP_REG_DATA_TBL(
    "ID" INTEGER,
    "V000" INTEGER,
    "V001" DOUBLE,
    "V002" VARCHAR(10),
    "V003" INTEGER
);
INSERT INTO PAL_PREDICT_MLP_REG_DATA_TBL VALUES (1, 1, 1.71, 'AC', 0);
INSERT INTO PAL_PREDICT_MLP_REG_DATA_TBL VALUES (2, 10, 1.78, 'CA', 5);
INSERT INTO PAL_PREDICT_MLP_REG_DATA_TBL VALUES (3, 17, 2.36, 'AA', 6);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL(
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR(1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 0.2, NULL);
CALL _SYS_AFL.PAL_MULTILAYER_PERCEPTRON_PREDICT(PAL_PREDICT_MLP_REG_DATA_TBL,
PAL_MLP_REG_MODEL_TBL, "#PAL_PARAMETER_TBL", ?, ?);
```

**Expected Result**

Your result may look different from the following results due to model randomness.

RESULT:

| | ID | TARGET | VALUE |
|---|---|---|---|
| 1 | 1 | T001 | 12.699643390458254 |
| 2 | 2 | T001 | 12.101069099278842 |
| 3 | 3 | T001 | 10.117332653886322 |
| 4 | 1 | T002 | 2.7999787461946273 |
| 5 | 2 | T002 | 7.9991874753104515 |
| 6 | 3 | T002 | 2.800134295637946 |
| 7 | 1 | T003 | 3.0599191969944495 |
| 8 | 2 | T003 | 2.650080246414321 |
| 9 | 3 | T003 | 3.237846357890039 |

### Related Information

Model Evaluation and Parameter Selection [page 2227]

## 5.2.10 Naive Bayes

Naive Bayes is a classification algorithm based on Bayes theorem. It estimates the class-conditional probability by assuming that the attributes are conditionally independent of one another.

Given the class label y and a dependent feature vector $x_1$ through $x_n$, the conditional independence assumption can be formally stated as follows:

$$P(y|x_1,...,x_n) = \frac{P(y)P(x_1,...,x_n)|y}{P(x_1,...,x_n)}$$

Using the naive independence assumption that

$$P(x_i|y, x_1, ..., x_{i-1}, x_{i+1}, ..., x_n) = P(x_i|y)$$

for all i, this relationship is simplified to

$$P(y|x_1,...,x_n)=\frac{P(y)\prod_{i=1}^{n}P(x_i|y)}{P(x_1,...,x_n)}$$

Since $P(x_1, ..., x_n)$ is constant given the input, we can use the following classification rule:

$$P(y|x_1,...,x_n) \propto P(y)\prod_{i=1}^{n}P(x_i|y) \Rightarrow \hat{y} = \arg\max_{y} P(y)\prod_{i=1}^{n}P(x_i|y)$$

We can use Maximum a Posteriori (MAP) estimation to estimate `P(y)` and `P(x_i|y)`. The former is then the relative frequency of class `y` in the training set.

The different Naive Bayes classifiers differ mainly by the assumptions they make regarding the distribution of `P(x_i|y)`.

For discrete attributes, the attribute data are fitted to a Gaussian distribution and get the `P(x_i|y)`.

For discrete attributes, the count number ratio is used as `P(x_i|y)`. However, if there are categories that did not occur in the training set, `P(x_i|y)` will become 0, while the actual probability is merely small instead of 0. This will bring errors to the prediction. To handle this issue, PAL introduces Laplace smoothing. The `P(x_i|y)` is then denoted as:

$$\hat{\theta}_i = \frac{x_i + \alpha}{N + \alpha d} \qquad (i = 1, \ldots, d),$$

This is a type of shrinkage estimator, as the resulting estimate is between the empirical estimate `xi / N`, and the uniform probability `1/d`. `α > 0` is the smoothing parameter, also called Laplace control value in the following discussion.

Despite its simplicity, Naive Bayes works well in areas like document classification and spam filtering, and it only requires a small amount of training data to estimate the parameters necessary for classification.

The Naive Bayes algorithm in PAL includes two procedures: PAL_NAIVE_BAYES for generating training model; and PAL_NAIVE_BAYES_PREDICT for making prediction based on the training model.

## Prerequisites

- The input data does not contain null value.
- The minimum training sample size is 10.

## _SYS_AFL.PAL_NAIVE_BAYES

This function reads input data and generates training model with the Naive Bayes algorithm.

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<MODEL table>` | OUT |
| 4 | `<STATISTICS table>` | OUT |
| 5 | `<OPTIMAL_PARA table>` | OUT |

### Input Table

| Table | Column | Data Type | Description |
|---|---|---|---|
| Data | 1st column | VARCHAR, NVARCHAR, or INTEGER | Record ID. If this column is absent, the HAS_ID parameter must be set to 0. |
| | Other columns | VARCHAR, NVARCAHR, DOUBLE, DECIMAL (p, s), or INTEGER<br>Dependent variable data type cannot be DOUBLE. | Attribute data and dependent variable. |

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| HAS_ID | INTEGER | 0 | Specifies whether the input data table has an ID column.<br>• 0: No<br>• 1: Yes |
| MODEL_FORMAT | INTEGER | 0 | Specifies model format.<br>• 0: Serializes the model to JSON format.<br>• Other values: Serializes the model to PMML format. |

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| LAPLACE | DOUBLE | 0 | Enables or disables Laplace smoothing.<br>• 0: Disables Laplace smoothing<br>• Positive value: Enables Laplace smoothing for discrete values<br>**Note**<br>The LAPLACE value is only stored by JSON format models. If the PMML format is chosen, you may need to set the LAPLACE value again in the predicting phase. |
| DISCRETIZATION | INTEGER | 0 | • 0: Disables discretization.<br>• Other values: Uses supervised discretization to all the continuous attributes. |
| DEPENDENT_VARIABLE | NVARCHAR | No default value | Specifies dependent variable by name. If this parameter is not specified, then the last column is used. |
| CATEGORICAL_VARIABLE | VARCHAR | No default value | Column name in the data table used as category variable.<br>By default, STRING is category variable, and INTEGER or DOUBLE is continuous variable. |
| THREAD_RATIO | DOUBLE | 0 | Specifies the ratio of total number of threads that can be used by this function. The value range is from 0 to 1, where 0 means only using 1 thread, and 1 means using at most all the currently available threads. Values outside the range will be ignored and this function heuristically determines the number of threads to use. |

### Parameters for Model Evaluation and Parameter Selection

_SYS_AFL.PAL_NAIVE_BAYES supports model evaluation and parameter selection. For more information, see the Model Evaluation and Parameter Selection topic.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| RESAMPLING_METHOD | NVARCHAR | No default value | Specifies resampling method for model evaluation or parameter selection.<br>• cv<br>• cv_sha<br>• cv_hyperband<br>• stratified_cv<br>• stratified_cv_sha<br>• stratified_cv_hyperband<br>• bootstrap<br>• bootstrap_sha<br>• bootstrap_hyperband<br>• stratified_bootstrap<br>• stratified_bootstrap_sha<br>• stratified_bootstrap_hyperband<br>If no value is specified for this parameter, neither model evaluation nor parameter selection is activated. | Options like stratified_* can only be applied to classification algorithms. Options like *_sha and *_hyperband can only be used in parameter selection, can not be used in model evaluation. |
| EVALUATION_METRIC | NVARCHAR | No default value | Naive Bayes supports the following evaluation metrics:<br>• ACCURACY<br>• F1_SCORE<br>• AUC | Mandatory. |
| FOLD_NUM | INTEGER | No default value | Specifies the fold number forthe cross validation method. | Mandatory and valid only when RESAMPLING_METHOD is set to cv or stratified_cv. Specifies the evaluation metric for model evaluation or parameter selection. |
| REPEAT_TIMES | INTEGER | 1 | Specifies the number of repeat times for resampling. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| PARAM_SEARCH_STRATEGY | NVARCHAR | No default value | Specifies the parameter search method:<br>• grid<br>• random | |
| RANDOM_SEARCH_TIMES | INTEGER | No default value | Specifies the number of times to randomly select candidate parameters for selection. | Mandatory and valid when PARAM_SEARCH_STRATEGY is set to random. |
| REDUCTION_RATE | DOUBLE | 3.0 | Specifies reduction rate in SHA or Hyperband method. | Only valid when RESAMPLING_METHOD is specified as *_sha or *_hyberband. |
| AGGRESSIVE_ELIMINATION | INTEGER | 0 | Specifies whether to apply aggressive elimination while using SHA method. Aggressive elimination happens when the data size and parameters size to be searched does not match and there are still bunch of parameters to be searched while data size reaches its upper limits. If aggressive elimination is applied, lower bound of limit of data size will be used multiple times first to reduce number of parameters.<br>• 0: does not apply aggressive elimination<br>• 1: applies aggressive elimination | Only valid when RESAMPLING_METHOD is specified as *_sha. |
| SEED | INTEGER | 0 | Specifies the seed for random generation. Uses system time when 0 is specified. | |
| TIMEOUT | INTEGER | 0 | Specifies maximum running time for model evaluation or parameter selection, in seconds. No timeout when 0 is specified. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| PROGRESS_INDICATOR_ID | NVARCHAR | No default value | Specifies the evaluation metric for model evaluation or parameterSets an ID of progress indicator for model evaluation or parameter selection. No progress indicator is active if no value is provided. | |
| LAPLACE_RANGE | NVARCHAR | No default value | Specifies candidate LAPLACE values for parameter selection. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| LAPLACE_VALUES | NVARCHAR | No default value | Specifies the range for candidate LAPLACE values for parameter selection. | Only valid when PARAM_SEARCH_STRATEGY is specified. |

### Output Tables

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| MODEL | 1st column | INTEGER | ROW_INDEX | Cluster model row index |
| | 2nd column | VARCHAR or NVARCHAR | MODEL_CONTENT | Deserialized cluster model.<br>The table must be a column table. The maximum length of every unit (row) is 5000. |
| STATISTICS | 1st column | VARCHAR or NVARCHAR | STAT_NAME | Statistic name |
| | 2nd column | VARCHAR or NVARCHAR | STAT_VALUE | Statistic value |
| OPTIMAL_PARAM | 1st column | NVARCHAR (256) | PARAM_NAME | Optimal parameters selected |
| | 2nd column | INTEGER | INT_VALUE | |
| | 3rd column | DOUBLE | DOUBLE_VALUE | |
| | 4th column | NVARCHAR (1000) | STRING_VALUE | |

### Example

Assume that:

- DM_PAL is a schema belonging to USER1;

- USER1 has been assigned the AFL__SYS_AFL_AFLPAL_EXECUTE or AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION role.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_NBCTRAIN_TRAININGSET_TBL;
CREATE COLUMN TABLE PAL_NBCTRAIN_TRAININGSET_TBL(
    "HomeOwner" VARCHAR (100),
    "MaritalStatus" VARCHAR (100),
    "AnnualIncome" DOUBLE,
    "DefaultedBorrower" VARCHAR (100)
);
INSERT INTO PAL_NBCTRAIN_TRAININGSET_TBL VALUES ('YES', 'Single', 125, 'NO');
INSERT INTO PAL_NBCTRAIN_TRAININGSET_TBL VALUES ('NO', 'Married', 100, 'NO');
INSERT INTO PAL_NBCTRAIN_TRAININGSET_TBL VALUES ('NO', 'Single', 70, 'NO');
INSERT INTO PAL_NBCTRAIN_TRAININGSET_TBL VALUES ('YES', 'Married', 120, 'NO');
INSERT INTO PAL_NBCTRAIN_TRAININGSET_TBL VALUES ('NO', 'Divorced', 95, 'YES');
INSERT INTO PAL_NBCTRAIN_TRAININGSET_TBL VALUES ('NO', 'Married', 60, 'NO');
INSERT INTO PAL_NBCTRAIN_TRAININGSET_TBL VALUES ('YES', 'Divorced', 220, 'NO');
INSERT INTO PAL_NBCTRAIN_TRAININGSET_TBL VALUES ('NO', 'Single', 85, 'YES');
INSERT INTO PAL_NBCTRAIN_TRAININGSET_TBL VALUES ('NO', 'Married', 75, 'NO');
INSERT INTO PAL_NBCTRAIN_TRAININGSET_TBL VALUES ('NO', 'Single', 90, 'YES');
DROP TABLE PAL_NBC_MODEL_TBL;
CREATE COLUMN TABLE PAL_NBC_MODEL_TBL(
    "ROW_INDEX" INTEGER,
    "MODEL_CONTENT" NVARCHAR(5000)
);
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL(
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR(1000)
);
INSERT INTO PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 0.2, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('LAPLACE', NULL, 1.0, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('MODEL_FORMAT', 1, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('DEPENDENT_VARIABLE', NULL, NULL,
'DefaultedBorrower');
DO BEGIN
    lt_data =SELECT * FROM PAL_NBCTRAIN_TRAININGSET_TBL;
    lt_param=SELECT * FROM PAL_PARAMETER_TBL;
    CALL _SYS_AFL.PAL_NAIVE_BAYES(:lt_data, :lt_param, lt_model, lt_1, lt_2);
    INSERT INTO PAL_NBC_MODEL_TBL SELECT * FROM :lt_model;
END;
SELECT * FROM PAL_NBC_MODEL_TBL;
```

### Expected Result

MODEL:

| | ROW_INDEX | MODEL_CONTENT |
|---|---|---|
| 1 | 0 | <PMML version="4.0" xmlns="http://www.dmg.org/PMML-4_0" xmlns:xsi="http://www.... |

### Example: Parameter Selection and Model Evaluation

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_NBCTRAIN_CV_TBL;
CREATE COLUMN TABLE PAL_NBCTRAIN_CV_TBL(
    "HomeOwner" VARCHAR (100),
    "MaritalStatus" VARCHAR (100),
    "AnnualIncome" DOUBLE,
    "DefaultedBorrower" VARCHAR (100)
);
INSERT INTO PAL_NBCTRAIN_CV_TBL VALUES ('YES', 'Single', 125, 'NO');
```

```
INSERT INTO PAL_NBCTRAIN_CV_TBL VALUES ('NO', 'Married', 100, 'NO');
INSERT INTO PAL_NBCTRAIN_CV_TBL VALUES ('NO', 'Single', 70, 'NO');
INSERT INTO PAL_NBCTRAIN_CV_TBL VALUES ('YES', 'Married', 120, 'NO');
INSERT INTO PAL_NBCTRAIN_CV_TBL VALUES ('NO', 'Divorced', 95, 'YES');
INSERT INTO PAL_NBCTRAIN_CV_TBL VALUES ('NO', 'Married', 60, 'NO');
INSERT INTO PAL_NBCTRAIN_CV_TBL VALUES ('YES', 'Divorced', 220, 'NO');
INSERT INTO PAL_NBCTRAIN_CV_TBL VALUES ('NO', 'Single', 85, 'YES');
INSERT INTO PAL_NBCTRAIN_CV_TBL VALUES ('NO', 'Married', 75, 'NO');
INSERT INTO PAL_NBCTRAIN_CV_TBL VALUES ('NO', 'Single', 90, 'YES');
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL(
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR(1000)
);
INSERT INTO PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 0.2, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('DEPENDENT_VARIABLE', NULL, NULL,
'DefaultedBorrower');
INSERT INTO PAL_PARAMETER_TBL VALUES ('MODEL_FORMAT', 1, NULL, NULL);
/* parameters for parameter selection and model evaluation */
INSERT INTO PAL_PARAMETER_TBL VALUES ('RESAMPLING_METHOD', NULL, NULL, 'cv');
INSERT INTO PAL_PARAMETER_TBL VALUES ('EVALUATION_METRIC', NULL, NULL,
'ACCURACY');
INSERT INTO PAL_PARAMETER_TBL VALUES ('FOLD_NUM', 5, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('PARAM_SEARCH_STRATEGY', NULL, NULL,
'grid');
INSERT INTO PAL_PARAMETER_TBL VALUES ('TIMEOUT', 600, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 0.5, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('LAPLACE_RANGE', NULL,
NULL,'[0.01,0.1,1.0]');
DROP TABLE PAL_NBC_MODEL_TBL;
CREATE COLUMN TABLE PAL_NBC_MODEL_TBL(
    "ROW_INDEX" INTEGER,
    "MODEL_CONTENT" NVARCHAR(5000)
);
DO BEGIN
    lt_data =SELECT * FROM PAL_NBCTRAIN_CV_TBL;
    lt_param=SELECT * FROM PAL_PARAMETER_TBL;
    CALL _SYS_AFL.PAL_NAIVE_BAYES(:lt_data, :lt_param, lt_model, lt_1, lt_2);
    INSERT INTO PAL_NBC_MODEL_TBL SELECT * FROM :lt_model;
END;
SELECT * FROM PAL_NBC_MODEL_TBL;
```

### Expected Result

MODEL:

| | ROW_INDEX | MODEL_CONTENT |
|---|---|---|
| 1 | 0 | <PMML version="4.0" xmlns="http://www.dmg.org/PMML-4_0" xmlns:xsi="http://www.... |

OPTIMAL_PARM

| | PARAM_NAME | INT_VALUE | DOUBLE_VALUE | STRING_VALUE |
|---|---|---|---|---|
| 1 | LAPLACE | ? | 0.01 | ? |
| 2 | DISCRETIZATION | 0 | ? | ? |

## _SYS_AFL.PAL_NAIVE_BAYES_PREDICT

This function uses the training model generated by PAL_NAIVE_BAYES to make predictive analysis.

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 3 | `<MODEL table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 4 | `<RESULT table>` | OUT |

### Input Tables

| Table | Column | Reference for Output Table | Data Type | Description |
|---|---|---|---|---|
| DATA | 1st column | ID | VARCHAR, NVARCHAR, or INTEGER | Record ID. |
| | Other columns | | VARCHAR, NVARCAHR, DOUBLE, DECIMAL (p, s), or INTEGER | Attribute data to be predicted |
| MODEL | 1st column | | INTEGER | Row Index |
| | 2nd column | | VARCHAR or NVARCHAR | Saved model |

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| LAPLACE | DOUBLE | 0 | Enables or disables Laplace smoothing.<br>• 0: Disables Laplace smoothing<br>• Positive value: Enables Laplace smoothing for discrete values<br>**Note**<br>The LAPLACE value is only stored by JSON format models. If the PMML format is chosen, you may need to set the LAPLACE value again in the predicting phase. |
| VERBOSE_OUTPUT | INTEGER | 0 | • 1: Outputs the probability of all label categories.<br>• 0: Outputs the category of the highest probability only. |
| THREAD_RATIO | DOUBLE | 0 | Specifies the ratio of total number of threads that can be used by this function. The value range is from 0 to 1, where 0 means only using 1 thread, and 1 means using at most all the currently available threads. Values outside the range will be ignored and this function heuristically determines the number of threads to use. |

### Output Table

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| OPTIMAL_PARAM | 1st column | ID (in input table) data type | ID (in input table) data type | Record ID |
| | 2nd column | VARCHAR or NVARCHAR | CLASS | Predictive result |
| | 3rd column (optional) | DOUBLE | CONFIDENCE | Confidence for the prediction of the sample, which is a logarithmic value of the a-posterior probabilities. |

### Example

Assume that:

- DM_PAL is a schema belonging to USER1;
- USER1 has been assigned the AFL__SYS_AFL_AFLPAL_EXECUTE or AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION role.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_NBCPREDICT_PREDICTION_TBL;
CREATE COLUMN TABLE PAL_NBCPREDICT_PREDICTION_TBL(
    "ID" INTEGER,
    "HomeOwner" VARCHAR(100),
    "MaritalStatus" VARCHAR(100),
    "AnnualIncome" DOUBLE
);
INSERT INTO PAL_NBCPREDICT_PREDICTION_TBL VALUES (0, 'NO', 'Married', 120);
INSERT INTO PAL_NBCPREDICT_PREDICTION_TBL VALUES (1, 'YES', 'Married', 180);
INSERT INTO PAL_NBCPREDICT_PREDICTION_TBL VALUES (2, 'NO', 'Single', 90);
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL(
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR(1000)
);
INSERT INTO PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 0.2, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('LAPLACE', NULL, 1.0, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('VERBOSE_OUTPUT', 1, NULL, NULL);
DROP TABLE PAL_NBCPREDICT_RESULT_TBL;
CREATE COLUMN TABLE PAL_NBCPREDICT_RESULT_TBL(
    "ID" INTEGER,
    "CLASS" VARCHAR(100),
    "CONFIDENCE" DOUBLE
);
DO BEGIN
    lt_data = SELECT * FROM PAL_NBCPREDICT_PREDICTION_TBL;
    lt_model = SELECT * FROM PAL_NBC_MODEL_TBL;
    lt_para  = SELECT * FROM PAL_PARAMETER_TBL;
    CALL _SYS_AFL.PAL_NAIVE_BAYES_PREDICT(:lt_data, :lt_model, :lt_para,
lt_result);
    INSERT INTO PAL_NBCPREDICT_RESULT_TBL SELECT * FROM :lt_result;
END;
SELECT * FROM PAL_NBCPREDICT_RESULT_TBL;
```

### Expected Result

RESULT:

When VERBOSE_OUTPUT is set to 0:

| | ID | CLASS | CONFIDENCE |
|---|---|---|---|
| 1 | 0 | NO | -6.572352631131612 |
| 2 | 1 | NO | -7.602220663331096 |
| 3 | 2 | YES | -4.64863998190415 |

When VERBOSE_OUTPUT is set to 1:

| | ID | CLASS | CONFIDENCE |
|---|---|---|---|
| 1 | 0 | NO | -6.572352631131612 |
| 2 | 0 | YES | -23.74725227057226 |
| 3 | 1 | NO | -7.602220663331096 |
| 4 | 1 | YES | -169.13354663169... |
| 5 | 2 | NO | -7.133598534952933 |
| 6 | 2 | YES | -4.64863998190415 |

## 5.2.11 Online Multi-Class Logistic Regression

This algorithm is the online version of Multi-Class Logistic Regression, while the Multi-Class Logistic Regression is offline/batch version.

The difference is that during training phase, for the offline/batch version algorithm it requires all training data to be fed into the algorithm in one batch, then it tries its best to output one model to best fit the training data. This infers that the computer must have enough memory to store all data, and can obtain all data in one batch. Online version algorithm applies in scenario that either or all these two assumptions are not right.

The online version treats the training data input as a streaming data. Streaming data could continuously arrive at any time. Thus this algorithm is designed to run over multiple rounds. In each round we can obtain arbitrary arrival streaming data, based on the model obtained from the last round, we iterate the streaming data one by one and improve the model accordingly. At the end of current round it will output an improved model comparing to the input model at the beginning of the current round.

The online version only implements the training phase. Since the format of output model is completely compatible with Multi-Class Logistic Regression, so the procedure `_SYS_AFL.PAL_MULTICLASS_LOGISTIC_REGRESSION_PREDICT` can be reused in testing phase also. Note currently only table-based format of model is supported.

**Algorithms**

Current algorithm uses multinomial logit model.

Let $W = \{w_{j,k}\}_{(P+1)\times C}$ be the model, where P is feature dimension, and C is the number of class labels. When $P = 0$ it represents the intercept term in model. Given one observations $z = \langle x,y\rangle$, where $y \in \{1,\ldots,C\}$, and $x = \{x_j\}_{1\times P}$, the probability with a given model and given observation will be:

$$p(y \mid x, W) = \frac{\exp\left(w_{0,y} + \sum_{j=1}^{P} w_{j,y} \times x_j\right)}{\sum_{k=1}^{C} \exp\left(w_{0,k} + \sum_{j=1}^{P} w_{j,k} \times x_j\right)}$$

Here we use its negative log likelihood form $NLL(z,W) = -\log p(y \mid x,W)$ to compute the optimized model.

In online training, we try to obtain a model that minimize the expectation of `NLL`:

$$W = \underset{W'}{\operatorname{argmin}}\, E_Z\left(NLL(z,W')\right)$$

In order to avoid overfit, current implementation also support Ridge, Lasso and their mixed form ElasticNet.

The form of the penalty over W is:

$$\Psi(W) = \lambda \times \left[\frac{1}{2} \times (1-\alpha) \times \sum_{k=1}^{C} \sum_{j=1}^{p} w_{j,k}^2 + \alpha \times \sum_{k=1}^{C} \sum_{j=1}^{p} \left|w_{j,k}\right|\right]$$

Here when $\alpha = 1$, it's pure Lasso Penalty; when $\alpha=0$, it's pure Ridge Penalty.

The penalized objective for the expected model becomes:

$$W = \underset{W'}{\operatorname{argmin}} E_z\left(NLL(z, W')\right) + \Psi(W')$$

To compute the model we use SGD(Stochastic Gradient Descent) algorithm.

For the step size of SGD, current implementation support Learning Rate Schedule and Learning Rate Adaptation.

**Learning Rate Schedule**

In Learning Rate Schedule, the step size for each step in SGD is purely determined by preset hyper parameters and current step count. So the chosen hyper parameters for each kind of Learning Rate Schedule is fairly important, they will affect the convergence rate of the algorithm. Except the piece-wise constant decay schedule, all others contains decay rate, drop rate hyper parameters. Among them, the decay controls the drop speed of step size, the bigger the faster; The drop rate controls the drop frequency of step size.

In all following schedule related formula, $\eta_s$represents current step size, $\eta_0$represents initial step size, $\kappa$represents decay rate, $\tau$ represents drop rate, $s$represents current step counter.

**Inverse Time Decay**

The formula for inverse time decay is:

$$\eta_s = \frac{\eta_0}{1+\kappa \times s/\tau}$$

This formula also contains a stair case parameter, which is a boolean. It controls whether step size drop in a stair way or in a smooth way. In the above formula, if stair case is true then $s/\tau$ should be replaced by $\lfloor s/\tau \rfloor$ (applied for floor operation).

In order to avoid underflow, there is another parameter $\epsilon$ , which defines the smallest possible step size. Once current step size reach this value, it will no longer change in the following steps.

**Exponential Decay**

The formula for exponential decay is: $\eta_s = \frac{\eta_0}{\kappa^{s/\tau}}$

This formula also contains a stair case parameter, which is a boolean. It controls whether step size drop in a stair way or in a smooth way. In above formula, if stair case is true then $s/\tau$ should be replaced by $\lfloor s/\tau \rfloor$ (apply floor operation).

In order to avoid underflow, there is another parameter $\varepsilon$, which defines the smallest possible step size. Once current step size reach this value, it will no longer change in the following steps.

**Polynomial Decay**

In Polynomial Decay formula, cycle is boolean parameter indicating whether need cycle from start once reaching end step size; $\eta_e$ represents end step size. If cycle is false, the formula for polynomial decay is:

$$\eta_s = (\eta_0 - \eta_e) \times (1 - s/\tau)^{\kappa} + \eta_e$$

Once reaching the end learning rate, it will keep this learning rate for all following steps.

If cycle is true, the formula for polynomial decay is:

$$\eta_s = (\eta_0 - \eta_e) \times \left(1 - \frac{s}{\tau \times \lceil s/\tau \rceil}\right)^{\kappa} + \eta_e$$

### Piece-wise Constant Decay

This schedule specifies different constant step size values for step counter falls in different regions. The parameters consist two vectors. One vector specifies step counter boundaries, the other specifies constant step sizes for different regions defined by the boundaries. Let B denote the vector containing boundary values, C denote the vector containing constant step size values. The C vector must always has one more element than B. Vector B can be empty. But if it contains values, the value must be stored in increasing order.

The formula for piece-wise constant decay is: When

$$\eta_s = \begin{cases} C_0, & s \in [0, B_0] \\ C_i, & s \in (B_{i-1}, B_i] \\ C_{len(B)}, & s \in (B_{len(B)-1}, \infty) \end{cases}$$

### Learning Rate Adaptation

These kind of learning rate is adaptive. Unlike learning rate schedule, which solely depends on the preset hyper parameter, adaptive learning rate also depends on the training data. Also unlike learning rate schedule, which always specify single step size for all model coefficients, adaptive learning rate specify step size per model coefficient. Current implementation support three kinds of adaptive learning rate: AdaGrad, AdaDelta, RMSProp.

In following formula, we use $\eta_{j,k}(S)$ representing the current step size for feature j and class label k, η representing the general learning rate, ϵ to represent a small value, which usually used to avoid divide by 0, s represents current step counter.

### AdaGrad

Let $G_{j,k}(S)$ be the accumulator of squares of gradient $g_{j,k}(s)$ for feature `j` and class label `k` from step `0` to step `s`:

$$G_{j,k}(s) = G_{j,k}(s-1) + g_{j,k}(s)^2$$

Then the step size for step `s` will be:

$$\eta_{j,k}(s) = \frac{\eta}{\sqrt{G_{j,k}(s) + \epsilon}}$$

### AdaDelta

Unlike AdaGrad, which take all historic gradients values into account, AdaDelta will take only recent gradients into account. To accomplish this function, instead manually store recent values, it uses exponential moving average to gradually forget too far away gradients. Here it uses a hyper parameter γ to define how many gradients to remember. Now the accumulator of gradients square define as:

$$G_{j,k}(s) = \gamma \times G_{j,k}(s-1) + (1-\gamma) \times g_{j,k}(s)^2$$

AdaDelta also take delta coefficients

$$\Delta W_{j,k}(s) = -\frac{\eta}{\sqrt{G_{j,k}(s) + \epsilon}} \times g_{j,k}(s)$$

into account. Let $\Delta W_{j,k}$ represents the delta accumulator, it is defined as:

$$\Delta W_{j,k}(s) = \gamma \times \Delta W_{j,k}(s-1) + (1-\gamma) \times \Delta w_{j,k}(s)^2$$

Then the step size for step s will be:

$$\eta_{j,k}(s) = \frac{\sqrt{\Delta W_{j,k}(s-1)+\epsilon}}{\sqrt{G_{j,k}(s)+\epsilon}}$$

**RMSProp**

RMSProp is similar to AdaDelta, except it only consider gradients.

The graduent square accumulation is the same with AdaDelta. The step size for step s will be:

$$\eta_{j,k}(s) = \frac{\eta}{\sqrt{G_{j,k}(s)+\epsilon}}$$

## _SYS_AFL.PAL_INIT_ONLINE_MULTICLASS_LOGISTIC_REGRESSION

This procedure is used to initialize the online logistic regression model.

The initialization mainly deal with the hyper parameter initialization of training model.

It includes regularization, learning rate, data shuffle, etc.

The hyper parameters can only be set in the initialization procedure, and once set, these parameters can't be altered any more in later online training phase.

**Overview of All Tables**

| Position | Table Name | Type |
|---|---|---|
| 1 | `<Parameter Table>` | IN |
| 2 | `<Online Result Table>` | OUT |

Parameter Table

Mandatory Parameters

| Name | Data Type | Description | Dependency |
|---|---|---|---|
| CLASS_LABEL | VARCHAR, NVARCHAR | This is a multi-value parameter for class label. Specify this parameter multiple times, one per label. There should be at least two class labels. | |
| INITIAL_LEARN_RATE | DOUBLE | The initial learning rate for learning rate schedule. Value should be larger than 0. | Only valid when LEARN_RATE_TYPE is 0, 1, 2 |

| Name | Data Type | Description | Dependency |
|---|---|---|---|
| DECAY | DOUBLE | Specify the learning rate decay speed for learning rate schedule. Larger value indicates faster decay. Value should be larger than 0. When LEARN_RATE_TYPE is 1, value should be larger than 1. | Only valid when LEARN_RATE_TYPE is 0, 1, 2 |
| DROP_RATE | INTEGER | Specify the decay frequency. There are apparent effect when STAIR_CASE is true. Value should be larger than 0. | Only valid when LEARN_RATE_TYPE is 0, 1, 2 |
| STEP_BOUNDARIES | VARCHAR, NVARCHAR | Specify the step boundaries for regions where step size remains constant. The format of this parameter is a comma separated unsigned integer value. The step value start from 0. The values should be in increasing order. Empty value for this parameter is allowed. | Only valid when LEARN_RATE_TYPE is 3 |
| CONSTANT_VALUES | VARCHAR, NVARCHAR | Specifies the constant step size for each region defined by STEP_BOUNDARIES. The format of this parameter is a comma separated double value. There should always be one more value than STEP_BOUNDARIES. | Only valid when LEARN_RATE_TYPE is 3 |

Optional Parameters

The following parameter is optional. If it is not specified. PAL will use its default value.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| ENET_LAMBDA | DOUBLE | 0.0 | Penalized constant. The value should be larger than or equal to 0.0. The higher the value, the stronger the regularization. When it equal to 0.0, there is no regularization. | |
| ENET_ALPHA | DOUBLE | 1.0 | Elastic-Net mixing parameter. The valid range is [0, 1]. When it is 0, this means Ridge penalty; When it is 1, it is Lasso penalty. | Only valid when ENET_LAMBDA is not 0.0 |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| NEED_SHUFFLE | INTEGER | 0 | Boolean value indicating whether need to shuffle the row order of observation data. The value can only be 0 or 1. 0 means keeping original order; 1 means performing shuffle operation. | |
| SHUFFLE_SEED | INTEGER | 0 | The seed is used to initialize the random generator to perform shuffle operation. The value of this parameter should be larger than or equal to 0. If need to reproduce the result when performing shuffle operation, please set this value to non-zero. | Only valid when NEED_SHUFFLE is 1. |
| NEED_WEIGHT_AVERAGE | INTEGER | 0 | Boolean value indicating whether need to perform average operator over output model. The value can only be 0 or 1. 0 means directly output model; 1 means perform average operator over output model. Currently only support Polyak Ruppert Average. | |
| WEIGHT_AVERAGE_BEGIN | INTEGER | 0 | Specify the beginning step counter to perform the average operator over model. The value should be larger than or equal to 0. When current step counter is less than this parameter, just directly output model. | Only valid when NEED_WEIGHT_AVERAGE is 1. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| LEARN_RATE_TYPE | INTEGER | 6 | Specify the learning rate type for SGD algorithm. The value of this parameter should be within [0, 6]. Meaning of each value is:<br>• 0：Inverse Time Decay<br>• 1：Exponential Decay<br>• 2：Polynomial Decay<br>• 3：Piecewise Constant Decay<br>• 4：AdaGrad<br>• 5：AdaDelta<br>• 6：RMSProp | |
| GENERAL_LEARN_RATE | DOUBLE | 0.001 | Specify the general learning rate used in AdaGrad and RMSProp. The value should be larger than 0. | Only valid when LEARN_RATE_TYPE is 4, 6. |
| STAIR_CASE | INTEGER | 0 | Boolean value indicate the drop way of step size. The value of this parameter can only be 0 or 1. 0 means drop step size smoothly; 1 means drop step size in a stair way. | Only valid when LEARN_RATE_TYPE is 0, 1. |
| CYCLE | INTEGER | 0 | Boolean value indicate whether need to cycle from the start when reaching specified end learning rate. The value of this parameter can only be 0 or 1. 0 means do not cycle from the start; 1 means cycle from the start. | Only valid when LEARN_RATE_TYPE is 2. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| EPSILON | DOUBLE | 1E-8 | This parameter has multiple purposes depending on the learn rate type. The value should be within (0, 1). When used in learn rate type 0 and 1, it represent the smallest allowable step size. When step size reach this value, it will no longer change. When used in learn rate type 2, it represent the end learn rate. When used in learn rate type 4, 5, 6, it is used to avoid dividing by 0. | Only valid when LEARN_RATE_TYPE is not 3. |
| WINDOW_SIZE | DOUBLE | 0.9 | This parameter controls the moving window size of recent steps. The value should be in range (0, 1). Larger value means more steps are kept in track. | Only valid when LEARN_RATE_TYPE is 5, 6. |

### Output Table

Table Online Result

| Column | Data Type | Column Name | Description |
|---|---|---|---|
| 1st Column | INTEGER | SEQUENCE | Sequence ID |
| 2nd Column | NVARCHAR(5000) | SERIALIZED_RESULT | Serialized UTF-8 escaped initial Online Model |

### Example: Create initial model for online multi-class logistic regression

Example 1: Initialize a Inverse Time Schedule Model without Regularization

```
SET SCHEMA DM_PAL;

DROP TABLE PAL_ONLINE_RESULT_TBL;
CREATE COLUMN TABLE PAL_ONLINE_RESULT_TBL(
    "SEQUENCE" INTEGER,
    "SERIALIZED_RESULT" NVARCHAR(5000)
);
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL(
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR(1000)
);
TRUNCATE TABLE PAL_PARAMETER_TBL;
```

```
-- Use Inverse Time Schedule
INSERT INTO PAL_PARAMETER_TBL("PARAM_NAME", "INT_VALUE", "DOUBLE_VALUE",
"STRING_VALUE") VALUES('LEARN_RATE_TYPE', 0, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL("PARAM_NAME", "INT_VALUE", "DOUBLE_VALUE",
"STRING_VALUE") VALUES('INITIAL_LEARN_RATE', NULL, 0.1, NULL);
INSERT INTO PAL_PARAMETER_TBL("PARAM_NAME", "INT_VALUE", "DOUBLE_VALUE",
"STRING_VALUE") VALUES('DECAY', NULL, 0.1, NULL);
INSERT INTO PAL_PARAMETER_TBL("PARAM_NAME", "INT_VALUE", "DOUBLE_VALUE",
"STRING_VALUE") VALUES('DROP_RATE', 4, NULL, NULL);
-- Drop step size in a stair way
INSERT INTO PAL_PARAMETER_TBL("PARAM_NAME", "INT_VALUE", "DOUBLE_VALUE",
"STRING_VALUE") VALUES('STAIR_CASE', 1, NULL, NULL);
-- Set the minimum allowed step size
INSERT INTO PAL_PARAMETER_TBL("PARAM_NAME", "INT_VALUE", "DOUBLE_VALUE",
"STRING_VALUE") VALUES('EPSILON', NULL, 1E-4, NULL);
-- Set all class label value, value should be unique, and at least two value
INSERT INTO PAL_PARAMETER_TBL("PARAM_NAME", "INT_VALUE", "DOUBLE_VALUE",
"STRING_VALUE") VALUES('CLASS_LABEL', NULL, NULL, '0');
INSERT INTO PAL_PARAMETER_TBL("PARAM_NAME", "INT_VALUE", "DOUBLE_VALUE",
"STRING_VALUE") VALUES('CLASS_LABEL', NULL, NULL, '1');
INSERT INTO PAL_PARAMETER_TBL("PARAM_NAME", "INT_VALUE", "DOUBLE_VALUE",
"STRING_VALUE") VALUES('CLASS_LABEL', NULL, NULL, '2');
DO BEGIN
    lt_para = SELECT *
        FROM PAL_PARAMETER_TBL;
    CALL _SYS_AFL.PAL_INIT_ONLINE_MULTICLASS_LOGISTIC_REGRESSION( :lt_para,
lt_online_result );
    TRUNCATE TABLE PAL_ONLINE_RESULT_TBL;
    INSERT INTO PAL_ONLINE_RESULT_TBL
        SELECT *
        FROM :lt_online_result;
END;
SELECT *
FROM PAL_ONLINE_RESULT_TBL;
```

Expected Result

| | SEQUENCE | |
|---|---|---|
| 1 | 0 | {"SGD":{"data":{"current_learn_rate":0.10000000000000001,"feature_coefficient":[],"intercept_coefficient":[],"step_counter":0},"hyp |

Example 2: Initialize a Piecewise Constant Schedule Model with Lasso Regularization, Shuffle Data Order

```
SET SCHEMA DM_PAL;

DROP TABLE PAL_ONLINE_RESULT_TBL;
CREATE COLUMN TABLE PAL_ONLINE_RESULT_TBL(
    "SEQUENCE" INTEGER,
    "SERIALIZED_RESULT" NVARCHAR(5000)
);
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL(
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR(1000)
);
TRUNCATE TABLE PAL_PARAMETER_TBL;
-- Use Lasso Regularization with LAMBDA as 0.01
INSERT INTO PAL_PARAMETER_TBL("PARAM_NAME", "INT_VALUE", "DOUBLE_VALUE",
"STRING_VALUE") VALUES('ENET_LAMBDA', NULL, 0.01, NULL);
INSERT INTO PAL_PARAMETER_TBL("PARAM_NAME", "INT_VALUE", "DOUBLE_VALUE",
"STRING_VALUE") VALUES('ENET_ALPHA', NULL, 1.0, NULL);
-- Shuffle Data Order before training, use Shuffle Seed as non-zero to reproduce
result
```

```
INSERT INTO PAL_PARAMETER_TBL("PARAM_NAME", "INT_VALUE", "DOUBLE_VALUE",
"STRING_VALUE") VALUES('NEED_SHUFFLE', 1, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL("PARAM_NAME", "INT_VALUE", "DOUBLE_VALUE",
"STRING_VALUE") VALUES('SHUFFLE_SEED', 2021, NULL, NULL);
-- Use Piece-wise Constant Schedule
INSERT INTO PAL_PARAMETER_TBL("PARAM_NAME", "INT_VALUE", "DOUBLE_VALUE",
"STRING_VALUE") VALUES('LEARN_RATE_TYPE', 3, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL("PARAM_NAME", "INT_VALUE", "DOUBLE_VALUE",
"STRING_VALUE") VALUES('STEP_BOUNDARIES', NULL, NULL, '8,16,24');
INSERT INTO PAL_PARAMETER_TBL("PARAM_NAME", "INT_VALUE", "DOUBLE_VALUE",
"STRING_VALUE") VALUES('CONSTANT_VALUES', NULL, NULL, '0.2,0.1,0.05,0.01');
-- Set all class label value, value should be unique, and at least two value
INSERT INTO PAL_PARAMETER_TBL("PARAM_NAME", "INT_VALUE", "DOUBLE_VALUE",
"STRING_VALUE") VALUES('CLASS_LABEL', NULL, NULL, '0');
INSERT INTO PAL_PARAMETER_TBL("PARAM_NAME", "INT_VALUE", "DOUBLE_VALUE",
"STRING_VALUE") VALUES('CLASS_LABEL', NULL, NULL, '1');
INSERT INTO PAL_PARAMETER_TBL("PARAM_NAME", "INT_VALUE", "DOUBLE_VALUE",
"STRING_VALUE") VALUES('CLASS_LABEL', NULL, NULL, '2');
DO BEGIN
    lt_para = SELECT *
        FROM PAL_PARAMETER_TBL;
    CALL _SYS_AFL.PAL_INIT_ONLINE_MULTICLASS_LOGISTIC_REGRESSION( :lt_para,
lt_online_result );
    TRUNCATE TABLE PAL_ONLINE_RESULT_TBL;
    INSERT INTO PAL_ONLINE_RESULT_TBL
        SELECT *
        FROM :lt_online_result;
END;
SELECT *
FROM PAL_ONLINE_RESULT_TBL;
```

Expected Result

| | SEQUENCE | |
|---|---|---|
| 1 | 0 | {"SGD":{"data":{"feature_coefficient":[],"feature_gradient_square_accumulation":[],"intercept_coefficient":[],"intercept_gradient_squ |

Example 3: Initialize a RMSProp Adaptive Learning Rate Model with ElasticNet Regularization, apply Polyak Ruppert Average on output result

```
SET SCHEMA DM_PAL;

DROP TABLE PAL_ONLINE_RESULT_TBL;
CREATE COLUMN TABLE PAL_ONLINE_RESULT_TBL(
    "SEQUENCE" INTEGER,
    "SERIALIZED_RESULT" NVARCHAR(5000)
);
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL(
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR(1000)
);
TRUNCATE TABLE PAL_PARAMETER_TBL;
-- Use ElasticNet Regularization with LAMBDA as 0.01 and ALPHA as 0.2
INSERT INTO PAL_PARAMETER_TBL("PARAM_NAME", "INT_VALUE", "DOUBLE_VALUE",
"STRING_VALUE") VALUES('ENET_LAMBDA', NULL, 0.01, NULL);
INSERT INTO PAL_PARAMETER_TBL("PARAM_NAME", "INT_VALUE", "DOUBLE_VALUE",
"STRING_VALUE") VALUES('ENET_ALPHA', NULL, 0.2, NULL);
-- Apply Polyak Ruppert Average on output model at step 8. Step start from 0.
INSERT INTO PAL_PARAMETER_TBL("PARAM_NAME", "INT_VALUE", "DOUBLE_VALUE",
"STRING_VALUE") VALUES('NEED_WEIGHT_AVERAGE', 1, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL("PARAM_NAME", "INT_VALUE", "DOUBLE_VALUE",
"STRING_VALUE") VALUES('WEIGHT_AVERAGE_BEGIN', 8, NULL, NULL);
```

```
-- Use RMSProp Adaptive Learning Rate
INSERT INTO PAL_PARAMETER_TBL("PARAM_NAME", "INT_VALUE", "DOUBLE_VALUE",
"STRING_VALUE") VALUES('LEARN_RATE_TYPE', 6, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL("PARAM_NAME", "INT_VALUE", "DOUBLE_VALUE",
"STRING_VALUE") VALUES('GENERAL_LEARN_RATE', NULL, 0.1, NULL);
INSERT INTO PAL_PARAMETER_TBL("PARAM_NAME", "INT_VALUE", "DOUBLE_VALUE",
"STRING_VALUE") VALUES('WINDOW_SIZE', NULL, 0.9, NULL);
INSERT INTO PAL_PARAMETER_TBL("PARAM_NAME", "INT_VALUE", "DOUBLE_VALUE",
"STRING_VALUE") VALUES('EPSILON', NULL, 1E-6, NULL);
-- Set all class label value, value should be unique, and at least two value
INSERT INTO PAL_PARAMETER_TBL("PARAM_NAME", "INT_VALUE", "DOUBLE_VALUE",
"STRING_VALUE") VALUES('CLASS_LABEL', NULL, NULL, '0');
INSERT INTO PAL_PARAMETER_TBL("PARAM_NAME", "INT_VALUE", "DOUBLE_VALUE",
"STRING_VALUE") VALUES('CLASS_LABEL', NULL, NULL, '1');
INSERT INTO PAL_PARAMETER_TBL("PARAM_NAME", "INT_VALUE", "DOUBLE_VALUE",
"STRING_VALUE") VALUES('CLASS_LABEL', NULL, NULL, '2');
DO BEGIN
    lt_para = SELECT *
        FROM PAL_PARAMETER_TBL;
    CALL _SYS_AFL.PAL_INIT_ONLINE_MULTICLASS_LOGISTIC_REGRESSION( :lt_para,
lt_online_result );
    TRUNCATE TABLE PAL_ONLINE_RESULT_TBL;
    INSERT INTO PAL_ONLINE_RESULT_TBL
        SELECT *
        FROM :lt_online_result;
END;
SELECT *
FROM PAL_ONLINE_RESULT_TBL;
```

Expected Result

| | SEQUENCE | |
|---|---|---|
| 1 | 0 | {"SGD":{"data":{"avg_feature_coefficient":[],"avg_intercept_coefficient":[],"feature_coefficient":[],"feature_gradient_square_accumu |

## _SYS_AFL.PAL_TRAIN_ONLINE_MULTICLASS_LOGISTIC_REGRESSION

This procedure is used to do online training per round. In each round, it is required to input json based format of online model obtained either from the output of last round or from the output of initialization procedure. At the end current round, it will output the json based format of online model improved by the training data fed in current round. Besides that, there is another table-based model output which is designed for prediction procedure.

This procedure support user cancellation and procedure will exit gracefully once receiving cancellation request. The performance of the procedure should be linear to the amount of data to be trained in each round.
For foreseeable long-running task, user can keep track of the progress of current execution by specifying parameter *PROGRESS_INDICATOR_ID*. During the execution of the procedure, user can open another session and query like this:

```
SELECT * FROM _SYS_AFL.FUNCTION_PROGRESS_IN_AFLPAL WHERE EXECUTION_ID =
<PROGRESS_INDICATOR_ID>;
```

Access this table requires the same permission as executing the PAL procedure. Current implementation will report a progress event when execution proceeds at least 1%. When reaching 100%, that means the procedure finishes execution. Note, when execution finished, all progress event reports in above table will be deleted. Thus monitoring can only be done during the execution.

**Overview**

| Position | Table Name | Type |
|---|---|---|
| 1 | `<DATA Table>` | IN |
| 2 | `<ONLINE_RESULT Table>` | IN |
| 3 | `<PARAMETER Table>` | IN |
| 4 | `<MODEL Table>` | OUT |
| 5 | `<UPDATED_ONLINE_RESULT Table>` | OUT |

### Input Table

#### Table Data

Note ID Column is optional, which depends on the value of parameter HAS_ID. If provided, ID Column must locate at the first column.

The dependent column(Class Label) can be in arbitrary position, it depends on the parameter DEPENDENT_VARIABLE, which specifies the column name. But if the parameter is not specified, the default position of this column should be right after the ID Column, if exists; Otherwise the default position of dependent column should at the first column.

All other columns belong to feature related columns. The set of all feature column names will be determined at the first found of online training. All following rounds of online training must provide exactly the same set of feature column names.

| Column | Data Type | Description |
|---|---|---|
| 1st column | INTEGER, VARCHAR, NVARCHAR | ID |
| 2nd column | INTEGER, VARCHAR, NVARCHAR | Class Label |
| Feature Columns | INTEGER, DOUBLE, DECIMAL(p, s) | Feature Xm |

#### Table Online_Result

| Column | Data Type | Description |
|---|---|---|
| 1st column | INTEGER | Sequence ID |
| 2nd column | VARCHAR, NVARCHAR | UTF-8 escaped Online Result from last training or initialization |

### Parameter Table

Mandatory Parameters

None

Optional Parameters

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| DEPENDENT_VARIABLE | VARCHAR, NVARCHAR | No default value | Column name in the data table used as dependent variable. If this parameter is not specified, the second column or first column will be processed as dependent variables dependending on the existence of ID column. |
| THREAD_RATIO | DOUBLE | 1.0 | Specifies the ratio of total number of threads that can be used by this function. The value range is from 0 to 1, where 0 means only using 1 thread, and 1 means using at most all the currently available threads. Values outside the range will be ignored and this function heuristically determines the number of threads to use. |
| HAS_ID | INTEGER | 0 | Specifies whether the input data table has an ID column. If it exists, the ID column must be the first column. 0: No ID column 1: Has an ID column |
| PROGRESS_INDICATOR_ID | VARCHAR, NVARCHAR | No default value | Specifies execution id for each execution. The length of this id should not exceed 64. |

### Output Table

Table Model

| Column | Data Type | Column Name | Description |
|---|---|---|---|
| 1st column | NVARCHAR(1000) | VARIABLE_NAME | Name of variable |
| 2nd column | NVARCHAR(1000) | CLASSLABEL | Class Label |
| 3rd column | DOUBLE | COEFFICIENT | Value of coefficients |

Table Updated_Online_Result

| Column | Data Type | Column Name | Description |
|---|---|---|---|
| 1st column | INTEGER | SEQUENCE | Sequence ID |
| 2nd column | NVARCHAR(5000) | UPDATED_SERIALIZED_RESULT | UTF-8 escaped Online Result after current training |

Example: online training multi-class logistic regression model

The initial online model used in this example is the output of Example 3 above.

```
SET SCHEMA DM_PAL;

DROP TABLE MCLRONLINE_DATA_TBL;
CREATE COLUMN TABLE MCLRONLINE_DATA_TBL(
    "X1" DOUBLE,
    "X2" DOUBLE,
    "Y" NVARCHAR(10)
);
DROP TABLE PAL_MCLRONLINE_MODEL_TBL;
CREATE COLUMN TABLE PAL_MCLRONLINE_MODEL_TBL(
    "VARIABLE_NAME" NVARCHAR(1000),
    "CLASSLABEL" NVARCHAR(1000),
    "COEFFICINET" DOUBLE
);
TRUNCATE TABLE PAL_PARAMETER_TBL;
INSERT INTO PAL_PARAMETER_TBL VALUES('DEPENDENT_VARIABLE', NULL, NULL, 'Y');
INSERT INTO PAL_PARAMETER_TBL VALUES('THREAD_RATIO', NULL, 0.0, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES('HAS_ID', 0, NULL, NULL);


-- first round
TRUNCATE TABLE MCLRONLINE_DATA_TBL;
INSERT INTO MCLRONLINE_DATA_TBL(X1, X2, Y) VALUES(1.1604561542577971,
-0.0795835638144109, '0');
INSERT INTO MCLRONLINE_DATA_TBL(X1, X2, Y) VALUES(1.2167223451480362,
-1.3153484494085286, '2');
INSERT INTO MCLRONLINE_DATA_TBL(X1, X2, Y) VALUES(1.0184740904714196,
-0.6006470066155096, '1');
INSERT INTO MCLRONLINE_DATA_TBL(X1, X2, Y) VALUES(0.88457993830269,
1.5461152417423643, '1');
INSERT INTO MCLRONLINE_DATA_TBL(X1, X2, Y) VALUES(2.432160469409038,
0.4258954796418304, '1');
INSERT INTO MCLRONLINE_DATA_TBL(X1, X2, Y) VALUES(1.5735057373655277,
-0.019851871338952165, '0');
INSERT INTO MCLRONLINE_DATA_TBL(X1, X2, Y) VALUES(1.2856113425408873,
-2.004879455089761, '1');
INSERT INTO MCLRONLINE_DATA_TBL(X1, X2, Y) VALUES(0.47836383784256586,
-1.79127949738491, '2');
DO BEGIN
    lt_data = SELECT *
        FROM MCLRONLINE_DATA_TBL;
    lt_para = SELECT *
        FROM PAL_PARAMETER_TBL;
    lt_online = SELECT *
        FROM PAL_ONLINE_RESULT_TBL;
    CALL
_SYS_AFL.PAL_TRAIN_ONLINE_MULTICLASS_LOGISTIC_REGRESSION( :lt_data, :lt_online, :
lt_para, lt_model, lt_updated_online );
    TRUNCATE TABLE PAL_MCLRONLINE_MODEL_TBL;
    INSERT INTO PAL_MCLRONLINE_MODEL_TBL
        SELECT *
        FROM :lt_model;
    TRUNCATE TABLE PAL_ONLINE_RESULT_TBL;
    INSERT INTO PAL_ONLINE_RESULT_TBL
        SELECT *
        FROM :lt_updated_online;
END;
SELECT *
FROM PAL_MCLRONLINE_MODEL_TBL;
SELECT *
FROM PAL_ONLINE_RESULT_TBL;

-- second round
TRUNCATE TABLE MCLRONLINE_DATA_TBL;
INSERT INTO MCLRONLINE_DATA_TBL(X1, X2, Y) VALUES(-1.7998028319226718,
1.2253125317895788, '1');
```

```
INSERT INTO MCLRONLINE_DATA_TBL(X1, X2, Y) VALUES(0.5529556362504024,
-2.1340070846576666, '2');
INSERT INTO MCLRONLINE_DATA_TBL(X1, X2, Y) VALUES(0.7501529269184743,
-1.3329598337335686, '2');
INSERT INTO MCLRONLINE_DATA_TBL(X1, X2, Y) VALUES(2.0242232303065966,
-1.406925430211221, '2');
INSERT INTO MCLRONLINE_DATA_TBL(X1, X2, Y) VALUES(1.204172807289227,
-1.3952837585144027, '1');
INSERT INTO MCLRONLINE_DATA_TBL(X1, X2, Y) VALUES(1.7451832716791604,
0.6478913217496891, '0');
INSERT INTO MCLRONLINE_DATA_TBL(X1, X2, Y) VALUES(1.4060530841137617,
0.1805296256197263, '0');
INSERT INTO MCLRONLINE_DATA_TBL(X1, X2, Y) VALUES(1.880983282352953,
-1.6278337197906632, '2');
DO BEGIN
    lt_data = SELECT *
        FROM MCLRONLINE_DATA_TBL;
    lt_para = SELECT *
        FROM PAL_PARAMETER_TBL;
    lt_online = SELECT *
        FROM PAL_ONLINE_RESULT_TBL;
    CALL
_SYS_AFL.PAL_TRAIN_ONLINE_MULTICLASS_LOGISTIC_REGRESSION( :lt_data, :lt_online, :
lt_para, lt_model, lt_updated_online );
    TRUNCATE TABLE PAL_MCLRONLINE_MODEL_TBL;
    INSERT INTO PAL_MCLRONLINE_MODEL_TBL
        SELECT *
        FROM :lt_model;
    TRUNCATE TABLE PAL_ONLINE_RESULT_TBL;
    INSERT INTO PAL_ONLINE_RESULT_TBL
        SELECT *
        FROM :lt_updated_online;
END;
SELECT *
FROM PAL_MCLRONLINE_MODEL_TBL;
SELECT *
FROM PAL_ONLINE_RESULT_TBL;

-- third round
TRUNCATE TABLE MCLRONLINE_DATA_TBL;
INSERT INTO MCLRONLINE_DATA_TBL(X1, X2, Y) VALUES(1.8606340079727632,
-2.4743131992623377, '2');
INSERT INTO MCLRONLINE_DATA_TBL(X1, X2, Y) VALUES(0.7106616333428581,
-3.317884774132736, '2');
INSERT INTO MCLRONLINE_DATA_TBL(X1, X2, Y) VALUES(1.1535878264713326,
0.5399487053986844, '0');
INSERT INTO MCLRONLINE_DATA_TBL(X1, X2, Y) VALUES(1.2974895165445346,
-1.8119331260004206, '2');
INSERT INTO MCLRONLINE_DATA_TBL(X1, X2, Y) VALUES(2.0717842695295436,
0.35178895585177394, '0');
INSERT INTO MCLRONLINE_DATA_TBL(X1, X2, Y) VALUES(1.5524558261702572,
0.5507873829238434, '0');
INSERT INTO MCLRONLINE_DATA_TBL(X1, X2, Y) VALUES(1.2026152319042822,
-1.2565700255623353, '2');
INSERT INTO MCLRONLINE_DATA_TBL(X1, X2, Y) VALUES(-2.348316308231962,
1.3849347848332725, '1');
DO BEGIN
    lt_data = SELECT *
        FROM MCLRONLINE_DATA_TBL;
    lt_para = SELECT *
        FROM PAL_PARAMETER_TBL;
    lt_online = SELECT *
        FROM PAL_ONLINE_RESULT_TBL;
    CALL
_SYS_AFL.PAL_TRAIN_ONLINE_MULTICLASS_LOGISTIC_REGRESSION( :lt_data, :lt_online, :
lt_para, lt_model, lt_updated_online );
    TRUNCATE TABLE PAL_MCLRONLINE_MODEL_TBL;
    INSERT INTO PAL_MCLRONLINE_MODEL_TBL
```

```
        SELECT *
        FROM :lt_model;
    TRUNCATE TABLE PAL_ONLINE_RESULT_TBL;
    INSERT INTO PAL_ONLINE_RESULT_TBL
        SELECT *
        FROM :lt_updated_online;
END;
SELECT *
FROM PAL_MCLRONLINE_MODEL_TBL;
SELECT *
FROM PAL_ONLINE_RESULT_TBL;

-- fourth round
TRUNCATE TABLE MCLRONLINE_DATA_TBL;
INSERT INTO MCLRONLINE_DATA_TBL(X1, X2, Y) VALUES(-2.132379704553742,
1.457749051332808, '1');
INSERT INTO MCLRONLINE_DATA_TBL(X1, X2, Y) VALUES(0.549665491921197,
0.17407759154942526, '1');
INSERT INTO MCLRONLINE_DATA_TBL(X1, X2, Y) VALUES(1.4226287792807097,
0.8153580708923158, '0');
INSERT INTO MCLRONLINE_DATA_TBL(X1, X2, Y) VALUES(1.3185436494277254,
0.06247238225625318, '0');
INSERT INTO MCLRONLINE_DATA_TBL(X1, X2, Y) VALUES(0.501686322091475,
-1.286536789178292, '1');
INSERT INTO MCLRONLINE_DATA_TBL(X1, X2, Y) VALUES(1.5417106846367898,
0.7375165740164671, '1');
INSERT INTO MCLRONLINE_DATA_TBL(X1, X2, Y) VALUES(1.7094864353182662,
-0.036971061157854423, '0');
INSERT INTO MCLRONLINE_DATA_TBL(X1, X2, Y) VALUES(1.7083667663730027,
0.7615717369281185, '0');
DO BEGIN
    lt_data = SELECT *
        FROM MCLRONLINE_DATA_TBL;
    lt_para = SELECT *
        FROM PAL_PARAMETER_TBL;
    lt_online = SELECT *
        FROM PAL_ONLINE_RESULT_TBL;
    CALL
_SYS_AFL.PAL_TRAIN_ONLINE_MULTICLASS_LOGISTIC_REGRESSION( :lt_data, :lt_online, :
lt_para, lt_model, lt_updated_online );
    TRUNCATE TABLE PAL_MCLRONLINE_MODEL_TBL;
    INSERT INTO PAL_MCLRONLINE_MODEL_TBL
        SELECT *
        FROM :lt_model;
    TRUNCATE TABLE PAL_ONLINE_RESULT_TBL;
    INSERT INTO PAL_ONLINE_RESULT_TBL
        SELECT *
        FROM :lt_updated_online;
END;
SELECT *
FROM PAL_MCLRONLINE_MODEL_TBL;
SELECT *
FROM PAL_ONLINE_RESULT_TBL;
```

### Expected Result

Round 1 Model Result

| | VARIABLE_NAME | CLASSLABEL | COEFFICINET |
|---|---|---|---|
| 1 | __PAL_INTERCEPT__ | 0 | -0.24513706167750132 |
| 2 | __PAL_INTERCEPT__ | 1 | 0.11239582521792671 |
| 3 | __PAL_INTERCEPT__ | 2 | -0.23628436997169885 |
| 4 | X1 | 0 | -0.1899296044276244 |
| 5 | X1 | 1 | 0.21892032052514104 |
| 6 | X1 | 2 | -0.3725004770189846 |
| 7 | X2 | 0 | 0.2795467967708799 |
| 8 | X2 | 1 | 0.45821406233611545 |
| 9 | X2 | 2 | -0.18537810245065753 |

Round 1 Online Json Model Result

| | SEQUENCE | |
|---|---|---|
| 1 | 0 | {"SGD":{"data":{"avg_feature_coefficient":[-0.18992960442762441,0.2795467967708799,0.21892032052514104,0.45821406233 |

Round 2 Model Result

| | VARIABLE_NAME | CLASSLABEL | COEFFICINET |
|---|---|---|---|
| 1 | __PAL_INTERCEPT__ | 0 | -0.34661126992342417 |
| 2 | __PAL_INTERCEPT__ | 1 | 0.15757068774682068 |
| 3 | __PAL_INTERCEPT__ | 2 | -0.1884077721069244 |
| 4 | X1 | 0 | -0.061235895282289184 |
| 5 | X1 | 1 | -0.017015730990268197 |
| 6 | X1 | 2 | -0.15044652312330942 |
| 7 | X2 | 0 | 0.4047719238612003 |
| 8 | X2 | 1 | 0.5797602891750289 |
| 9 | X2 | 2 | -0.42153105011607134 |

Round 2 Online Json Model Result

| | SEQUENCE | |
|---|---|---|
| 1 | 0 | {"SGD":{"data":{"avg_feature_coefficient":[-0.061235895282289184,0.40477192386120031,-0.017015730990268197,0.5797602 |

Round 3 Model Result

| | VARIABLE_NAME | CLASSLABEL | COEFFICINET |
|---|---|---|---|
| 1 | __PAL_INTERCEPT__ | 0 | -0.22683088678330102 |
| 2 | __PAL_INTERCEPT__ | 1 | 0.03621479736237377 |
| 3 | __PAL_INTERCEPT__ | 2 | -0.17761149336119053 |
| 4 | X1 | 0 | 0.08349116397590711 |
| 5 | X1 | 1 | -0.1831290388527893 |
| 6 | X1 | 2 | -0.1135774972620241 |
| 7 | X2 | 0 | 0.6084207887206752 |
| 8 | X2 | 1 | 0.5695829471297338 |
| 9 | X2 | 2 | -0.5595682829519744 |

Round 3 Online Json Model Result

| | SEQUENCE | |
|---|---|---|
| 1 | 0 | {"SGD":{"data":{"avg_feature_coefficient":[0.083491163975907112,0.60842078872067518,-0.18312903885278931,0.569582947 |

Round 4 Model Result

| | VARIABLE_NAME | CLASSLABEL | COEFFICINET |
|---|---|---|---|
| 1 | __PAL_INTERCEPT__ | 0 | -0.20575855577250937 |
| 2 | __PAL_INTERCEPT__ | 1 | 0.07358178327121073 |
| 3 | __PAL_INTERCEPT__ | 2 | -0.26260154525832363 |
| 4 | X1 | 0 | 0.22255303340468582 |
| 5 | X1 | 1 | -0.30448138812850206 |
| 6 | X1 | 2 | -0.1491645862713716 |
| 7 | X2 | 0 | 0.6797150466981607 |
| 8 | X2 | 1 | 0.5847415392750938 |
| 9 | X2 | 2 | -0.6254168679423224 |

Round 4 Online Json Model Result

| | SEQUENCE | |
|---|---|---|
| 1 | 0 | {"SGD":{"data":{"avg_feature_coefficient":[0.22255303340468582,0.67971504669816074,-0.30448138812850206,0.5847415392 |

## 5.2.12 Random Decision Trees

The random decision trees algorithm is an ensemble learning method for classification and regression. It grows many classification and regression trees, and outputs the class (classification) that is voted by majority or mean prediction (regression) of the individual trees.

The algorithm uses both bagging and random feature selection techniques. Each new training set is drawn with replacement from the original training set, and then a tree is grown on the new training set using random feature selection. Considering that the number of rows of the training data is n originally, two sampling methods for classification are available:

1. Bagging: The sampling size is n, and each one is drawn from the original data set with replacement.
2. Stratified sampling: For class `j`, $n_j$ data is drawn from it with replacement. And $n_1+n_2+...$ might not be exactly equal to n, but in PAL, the summation should not be larger than n, for the sake of out-of-bag error estimation. This method is used usually when imbalanced data presents.

The random decision trees algorithm generates an internal unbiased estimate (out-of-bag error) of the generalization error as the trees building processes, which avoids cross-validation. It gives estimates of what variables are important from nodes' splitting process. It also has an effective method for estimating missing data:

1. Training data: If the $m_{th}$ variable is numerical, the method computes the median of all values of this variable in class `j` or computes the most frequent non-missing value in class `j`, and then it uses this value to replace all missing values of the $m_{th}$ variable in class `j`.
2. Test data: The class label is absent, therefore one missing value is replicated n times, each filled with the corresponding class' most frequent item or median.

### Prerequisites

- The table used to store the tree model is a column table.
- The number of distinct class labels should be at least 2 for classification task.
- The scoring data (predict) must have an ID column.

- The independent variables of the predict data should be the same (name, order, data type) as those used in tree model training.

## _SYS_AFL.PAL_RANDOM_DECISION_TREES

This function is used for building a classification or regression model consisting of a number of decision trees.

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<MODEL table>` | OUT |
| 4 | `<VARIABLE_IMPORTANCE table>` | OUT |
| 5 | `<OUT_OF_BAG_ERROR table>` | OUT |
| 6 | `<CONFUSION_MATRIX table>` | OUT |

### Signature

### Input Table

| Table | Column | Data Type | Description | Constraint |
|---|---|---|---|---|
| DATA | Columns | INTEGER, DOUBLE, DECIMAL(p, s), VARCHAR, or NVARCHAR | Independent variables.<br>Discrete value: INTEGER, VARCHAR, or NVARCHAR<br>Continuous value: INTEGER, DOUBLE, or DECIMAL(p, s) | If the HAS_ID parameter is 1, the first column is an ID column, and can be of BIGINT type as well. |

| Table | Column | Data Type | Description | Constraint |
|---|---|---|---|---|
| | Last column | INTEGER, DOUBLE, DECIMAL(p, s), VARCHAR, or NVARCHAR | Dependent variable.<br>The varchar type is used for classification, and double type is for regression.<br>The integer type is by default used for regression, but if intentionally treated as categorical variable, which means using parameter CATEGORICAL_VARIABLE to specify, then it is solved as classification. | If DEPENDENT_VARIABLE is set to other columns, the last column is used as independent variable. |

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| THREAD_RATIO | DOUBLE | -1 | The ratio of available threads.<br>• 0: single thread<br>• 0~1: percentage<br>• Others: heuristically determined | |
| TREES_NUM | INTEGER | 100 | Specifies the number of trees to grow. | |
| TRY_NUM | INTEGER | sqrt(p) (for classification) or p/3(for regression), where p is the number of valid independent variables | Specifies the number of randomly selected variables for splitting.<br>Use a larger number for sparse datasets. | Should not be larger than the valid number of features. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| ALLOW_MISSING_DEPENDENT | INTEGER | 1 | Specifies if missing target value is allowed.<br>• 0: Not allowed. An error occurs if missing dependent presents.<br>• 1: Allowed. The datum with missing dependent is removed. | |
| CATEGORICAL_VARIABLE | VARCHAR or NVARCHAR | Detected from input data | Indicates which variables are treated as categorical. The default behavior is:<br>• STRING: categorical<br>• INTEGER and DOUBLE: continuous | VALID only for INTEGER variables; omitted otherwise. |
| SEED | INTEGER | 0 | Specifies the seed for random number generator.<br>• 0: Uses the current time (in second) as seed<br>• Others: Uses the specified value as seed | |
| NODE_SIZE | INTEGER | 1 for classification<br>5 for regression | Specifies the minimum number of records in a leaf. | |
| SPLIT_THRESHOLD | DOUBLE | 1e-5 | Specifies the stop condition: if the improvement value of the best split is less than this value, the tree stops growing. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| CALCULATE_OOB | INTEGER | 1 | Indicates whether to calculate out-of-bag error.<br>• 0: No<br>• 1: Yes | |
| MAX_DEPTH | INTEGER | 56 | The maximum depth of a tree. | If MAX_DEPTH is set to -1, there are no restrictions on the depth of the tree. Otherwise, MAX_DEPTH cannot be larger than 56. |
| SAMPLE_FRACTION | DOUBLE | 1 | The fraction of data used for training.<br>Assume there are `n` pieces of data, SAMPLE_FRACTION is `r`, then `n*r` data is selected for training. | In the range of [0,1].<br>If `n*r` is less than the number of classes, the sample size is restored to the number of classes.. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| STRATA | (INTEGER, DOUBLE), or (VARCHAR/NVARCHAR, DOUBLE) | No default value | Specifies the stratified sampling for classification.<br><br>This is a pair parameter, which means a parameter name corresponds to two values. The first value is the class label (INTEGER or VARCHAR/NVARCHAR), and the second one is the proportion of this class occupies in the sampling data. The class with a proportion less than 0 or larger than 1 will be ignored.<br><br>If some classes are absent, they share the remaining proportion equally.<br><br>If the summation of strata is less than 1, the remaining is shared by all classes equally. | Valid only for classification.<br><br>The summation of the proportion of strata should not be larger than 1.<br><br>If absent, bagging is used. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| PRIORS | (INTEGER, DOUBLE), (VARCHAR/NVARCHAR, DOUBLE) | Determined by the proportion of every class in the training data | Specifies the prior probabilities for classification.<br>This is also a pair parameter. The first value is the class label (INTEGER or VARCHAR/NVARCHAR), and the second one is the prior probability of this class. The class with a prior probability less than 0 or larger than 1 will be ignored.<br>If some classes are absent, they share the remaining probability equally.<br>If the summation of priors is less than 1, the remaining is shared by all classes equally. | Valid only for classification.<br>The summation of the priors should not be larger than 1. |
| HAS_ID | INTEGER | 0 | Specifies if the first column of DATA table is used as ID column<br>• 0: No<br>• 1: Yes | |
| DEPENDENT_VARIABLE | VARCHAR or NVARCHAR | The name of the last column name of DATA table | Specifies the variable used as dependent variable | Cannot be the first column name if HAS_ID is 1. |
| COMPRESSION | INTEGER | 0 | Specifies if the model is stored in compressed format<br>• 0: No<br>• 1: Yes | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| MAX_BITS | INTEGER | 12 | The maximum number of bits to quantize continuous features. Equivalent to use $2^{MAX_BITS}$ bins to quantize the values of these continuous features. Reduce the size of bins may affect the precision of split values and the accuracy in prediction. Must be less than 31. | Valid only when COMPRESSION is equal to 1. |
| QUANTIZE_RATE | DOUBLE | 0.005 | This value decides whether to do quantization for split values of a certain continuous feature. If the largest frequency of these continuous split values is less than QUANTIZE_RATE, quantization method will be used to quantize the split values of the continuous feature. | Valid only when COMPRESSION is equal to 1. |
| FITTINGS_QUANTIZATION | INTEGER | 0 | This parameter indicates whether to quantize fitting values (the values of leaves) in regression problem.<br>• 0: No<br>• 1: Yes<br>It is recommended to use this technique for large dataset in regression problem. | Valid only for regression when COMPRESSION is equal to 1. |
| MODEL_FORMAT | INTEGER | 1 | Specifies the tree model format for store:<br>• 1: JSON<br>• 2: PMML | If both MODEL_FORMAT and COMPRESSION are specified, COMPRESSION takes precedence. |

### Output Tables

| Table | Column | Data Type | Column Name | Description | Constraint |
|---|---|---|---|---|---|
| MODEL | 1st column | INTEGER | ROW_INDEX | Indicates the row. | Must be the first column. |
| | 2nd column | INTEGER | TREE_INDEX | Indicates the tree number. | |
| | 3rd column | NVARCHAR(5000) | MODEL_CONTENT | Model content. | PMML format |
| IMP | 1st column | NVARCHAR(256) | VARIABLE_NAME | Independent variable name | |
| | 2nd column | DOUBLE | IMPORTANCE | Variable importance | |
| OUT_OF_BAG | 1st column | INTEGER | TREE_INDEX | Indicates the tree number. | Must be the first column |
| | 2nd column | DOUBLE | ERROR | Out-of-bag error rate or mean squared error for random forest up to indexed tree | |
| CONFUSION (for classification only) | 1st column | NVARCHAR(1000) | ACTUAL_CLASS | The actual class name | |
| | 2nd column | NVARCHAR(1000) | PREDICTED_CLASS | The predicted class name | |
| | 3rd column | INTEGER | COUNT | Number of records | |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

Example 1: PMML format model.

```
SET SCHEMA DM_PAL;
DROP TABLE  PAL_RDT_DATA_TBL;
CREATE COLUMN TABLE PAL_RDT_DATA_TBL (
    "OUTLOOK" VARCHAR(20),
    "TEMP" INTEGER,
    "HUMIDITY" DOUBLE,
    "WINDY" VARCHAR(10),
    "CLASS" VARCHAR(20)
);
```

```
INSERT INTO PAL_RDT_DATA_TBL VALUES ('Sunny', 75, 70.0, 'Yes', 'Play');
INSERT INTO PAL_RDT_DATA_TBL VALUES ('Sunny', NULL, 90.0, 'Yes', 'Do not Play');
INSERT INTO PAL_RDT_DATA_TBL VALUES ('Sunny', 85, NULL, 'No', 'Do not Play');
INSERT INTO PAL_RDT_DATA_TBL VALUES ('Sunny', 72, 95.0, 'No', 'Do not Play');
INSERT INTO PAL_RDT_DATA_TBL VALUES (NULL, NULL, 70.0, NULL, 'Play');
INSERT INTO PAL_RDT_DATA_TBL VALUES ('Overcast', 72.0, 90, 'Yes', 'Play');
INSERT INTO PAL_RDT_DATA_TBL VALUES ('Overcast', 83.0, 78, 'No', 'Play');
INSERT INTO PAL_RDT_DATA_TBL VALUES ('Overcast', 64.0, 65, 'Yes', 'Play');
INSERT INTO PAL_RDT_DATA_TBL VALUES ('Overcast', 81.0, 75, 'No', 'Play');
INSERT INTO PAL_RDT_DATA_TBL VALUES (NULL, 71, 80.0, 'Yes', 'Do not Play');
INSERT INTO PAL_RDT_DATA_TBL VALUES ('Rain', 65, 70.0, 'Yes', 'Do not Play');
INSERT INTO PAL_RDT_DATA_TBL VALUES ('Rain', 75, 80.0, 'No', 'Play');
INSERT INTO PAL_RDT_DATA_TBL VALUES ('Rain', 68, 80.0, 'No', 'Play');
INSERT INTO PAL_RDT_DATA_TBL VALUES ('Rain', 70, 96.0, 'No', 'Play');
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (100)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('TREES_NUM', 300, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('TRY_NUM', 3, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SEED', 2, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SPLIT_THRESHOLD', NULL, 1e-5, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('CALCULATE_OOB', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('NODE_SIZE', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 1.0, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('MODEL_FORMAT', 2, NULL, NULL);
--A possible setting for stratified sampling
--INSERT INTO #PAL_PARAMETER_TBL VALUES ('SAMPLE_FRACTION', NULL, 1, NULL);
--INSERT INTO #PAL_PARAMETER_TBL VALUES ('STRATA', NULL, 0.5, 'Do not Play');
-- for class ‘Do not Play’
--INSERT INTO #PAL_PARAMETER_TBL VALUES ('STRATA', NULL, 0.5, 'Play');  -- for
class ‘Play’
-- A possible setting for prior probability
--INSERT INTO #PAL_PARAMETER_TBL VALUES ('PRIORS', NULL, 0.45, 'Do not Play');
-- for class ‘Do not Play’
--INSERT INTO #PAL_PARAMETER_TBL VALUES ('PRIORS', NULL, 0.55, 'Play');  -- for
class ‘Play’
DROP TABLE PAL_RDT_MODEL_TBL;  -- for predict followed
CREATE COLUMN TABLE PAL_RDT_MODEL_TBL (
    "ROW_INDEX" INTEGER,
    "TREE_INDEX" INTEGER,
    "MODEL_CONTENT" NVARCHAR(5000)
);
CALL _SYS_AFL.PAL_RANDOM_DECISION_TREES (PAL_RDT_DATA_TBL, #PAL_PARAMETER_TBL,
PAL_RDT_MODEL_TBL, ?, ?, ?) WITH OVERVIEW;
```

**Expected Results**

MODEL:

| | ROW_INDEX | TREE_INDEX | MODEL_CONTENT |
|---|---|---|---|
| 1 | 0 | -1 | <PMML version="4.0" xmlns="http://www.dmg.org/PMML-4_0" xmlns:xsi="http://www.... |
| 2 | 1 | 0 | <PMML version="4.0" xmlns="http://www.dmg.org/PMML-4_0" xmlns:xsi="http://www.... |
| 3 | 2 | 1 | <PMML version="4.0" xmlns="http://www.dmg.org/PMML-4_0" xmlns:xsi="http://www.... |
| 4 | 3 | 2 | <PMML version="4.0" xmlns="http://www.dmg.org/PMML-4_0" xmlns:xsi="http://www.... |
| 5 | 4 | 3 | <PMML version="4.0" xmlns="http://www.dmg.org/PMML-4_0" xmlns:xsi="http://www.... |
| 6 | 5 | 4 | <PMML version="4.0" xmlns="http://www.dmg.org/PMML-4_0" xmlns:xsi="http://www.... |
| 7 | 6 | 5 | <PMML version="4.0" xmlns="http://www.dmg.org/PMML-4_0" xmlns:xsi="http://www.... |
| 8 | 7 | 6 | <PMML version="4.0" xmlns="http://www.dmg.org/PMML-4_0" xmlns:xsi="http://www.... |
| 9 | 8 | 7 | <PMML version="4.0" xmlns="http://www.dmg.org/PMML-4_0" xmlns:xsi="http://www.... |
| 10 | 9 | 8 | <PMML version="4.0" xmlns="http://www.dmg.org/PMML-4_0" xmlns:xsi="http://www.... |
| 11 | 10 | 9 | <PMML version="4.0" xmlns="http://www.dmg.org/PMML-4_0" xmlns:xsi="http://www |

IMP:

| | VARIABLE_NAME | IMPORTANCE |
|---|---|---|
| 1 | OUTLOOK | 0.5084231490159329 |
| 2 | TEMP | 0.2404791120963799 |
| 3 | HUMIDITY | 0.2045606053096388 |
| 4 | WINDY | 0.04653713357804... |

OUT_OF_BAG:

| | TREE_INDEX | ERROR |
|---|---|---|
| 1 | 0 | 0.4 |
| 2 | 1 | 0.4 |
| 3 | 2 | 0.5 |
| 4 | 3 | 0.625 |
| 5 | 4 | 0.555... |
| 6 | 5 | 0.555... |
| 7 | 6 | 0.5 |
| 8 | 7 | 0.5 |
| 9 | 8 | 0.461... |
| 10 | 9 | 0.461... |
| 11 | 10 | 0.428... |
| 12 | 11 | 0.428... |
| 13 | 12 | 0.428... |
| 14 | 13 | 0.357... |
| 15 | 14 | 0.357... |
| 16 | 15 | 0.357... |
| 17 | 16 | 0.285... |
| 18 | 17 | 0.285... |
| 19 | 18 | 0.285... |
| 20 | 19 | 0.285... |
| 21 | 20 | 0.285... |
| 22 | 21 | 0.285... |
| 23 | 22 | 0.285... |
| 24 | 23 | 0.285... |
| 25 | 24 | 0.285... |
| 26 | 25 | 0.285... |
| 27 | 26 | 0.285... |

CONFUSION:

| | ACTUAL_CLASS | PREDICTED_CLASS | COUNT |
|---|---|---|---|
| 1 | Play | Play | 7 |
| 2 | Play | Do not Play | 2 |
| 3 | Do not Play | Play | 2 |
| 4 | Do not Play | Do not Play | 3 |

Example 2: JSON format model.

```
SET SCHEMA DM_PAL;
DROP TABLE  PAL_RDT_DATA_TBL;
CREATE COLUMN TABLE PAL_RDT_DATA_TBL (
    "OUTLOOK" VARCHAR(20),
    "TEMP" INTEGER,
    "HUMIDITY" DOUBLE,
    "WINDY" VARCHAR(10),
    "CLASS" VARCHAR(20)
);
INSERT INTO PAL_RDT_DATA_TBL VALUES ('Sunny', 75, 70.0, 'Yes', 'Play');
INSERT INTO PAL_RDT_DATA_TBL VALUES ('Sunny', NULL, 90.0, 'Yes', 'Do not Play');
INSERT INTO PAL_RDT_DATA_TBL VALUES ('Sunny', 85, NULL, 'No', 'Do not Play');
INSERT INTO PAL_RDT_DATA_TBL VALUES ('Sunny', 72, 95.0, 'No', 'Do not Play');
INSERT INTO PAL_RDT_DATA_TBL VALUES (NULL, NULL, 70.0, NULL, 'Play');
INSERT INTO PAL_RDT_DATA_TBL VALUES ('Overcast', 72.0, 90, 'Yes', 'Play');
INSERT INTO PAL_RDT_DATA_TBL VALUES ('Overcast', 83.0, 78, 'No', 'Play');
INSERT INTO PAL_RDT_DATA_TBL VALUES ('Overcast', 64.0, 65, 'Yes', 'Play');
INSERT INTO PAL_RDT_DATA_TBL VALUES ('Overcast', 81.0, 75, 'No', 'Play');
INSERT INTO PAL_RDT_DATA_TBL VALUES (NULL, 71, 80.0, 'Yes', 'Do not Play');
INSERT INTO PAL_RDT_DATA_TBL VALUES ('Rain', 65, 70.0, 'Yes', 'Do not Play');
INSERT INTO PAL_RDT_DATA_TBL VALUES ('Rain', 75, 80.0, 'No', 'Play');
INSERT INTO PAL_RDT_DATA_TBL VALUES ('Rain', 68, 80.0, 'No', 'Play');
INSERT INTO PAL_RDT_DATA_TBL VALUES ('Rain', 70, 96.0, 'No', 'Play');
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (100)
);
INSERT INTO PAL_PARAMETER_TBL VALUES ('TREES_NUM', 300, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('TRY_NUM', 3, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('SEED', 2, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('SPLIT_THRESHOLD', NULL, 1e-5, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('CALCULATE_OOB', 1, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('NODE_SIZE', 1, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 1.0, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('MODEL_FORMAT', 1, NULL, NULL);
DROP TABLE PAL_RDT_MODEL_TBL;  -- for predict followed
CREATE COLUMN TABLE PAL_RDT_MODEL_TBL (
    "ROW_INDEX" INTEGER,
    "TREE_INDEX" INTEGER,
    "MODEL_CONTENT" NVARCHAR(5000)
);
CALL _SYS_AFL.PAL_RANDOM_DECISION_TREES (PAL_RDT_DATA_TBL, PAL_PARAMETER_TBL,
PAL_RDT_MODEL_TBL, ?, ?, ?) WITH OVERVIEW;
```

Expected Results:

MODEL:

| | ROW_INDEX | TREE_INDEX | MODEL_CONTENT |
|---|---|---|---|
| 1 | 0 | -1 | {"atFields":[{"dataType":"string","name":"OUTLOOK","opType":"cat","valu... |
| 2 | 1 | 0 | {"h":3,"nnodes":5,"ts":"10100","nodes":[{"node_id":0,"record":14,"sc":"Play... |
| 3 | 2 | 1 | {"h":3,"nnodes":5,"ts":"10100","nodes":[{"node_id":0,"record":14,"sc":"Play... |
| 4 | 3 | 2 | {"h":2,"nnodes":3,"ts":"100","nodes":[{"node_id":0,"record":14,"sc":"Play","... |
| 5 | 4 | 3 | {"h":2,"nnodes":3,"ts":"100","nodes":[{"node_id":0,"record":14,"sc":"Play","... |
| 6 | 5 | 4 | {"h":5,"nnodes":9,"ts":"111100000","nodes":[{"node_id":0,"record":14,"sc":... |
| 7 | 6 | 5 | {"h":5,"nnodes":9,"ts":"111100000","nodes":[{"node_id":0,"record":14,"sc":... |
| 8 | 7 | 6 | {"h":2,"nnodes":3,"ts":"100","nodes":[{"node_id":0,"record":14,"sc":"Do no... |
| 9 | 8 | 7 | {"h":2,"nnodes":3,"ts":"100","nodes":[{"node_id":0,"record":14,"sc":"Do no... |
| 10 | 9 | 8 | {"h":4,"nnodes":7,"ts":"1010100","nodes":[{"node_id":0,"record":14,"sc":"D... |
| 11 | 10 | 9 | {"h":4,"nnodes":7,"ts":"1010100","nodes":[{"node_id":0,"record":14,"sc":"D... |
| 12 | 11 | 10 | {"h":3,"nnodes":5,"ts":"11000","nodes":[{"node_id":0,"record":14,"sc":"Play... |
| 13 | 12 | 11 | {"h":3,"nnodes":5,"ts":"11000","nodes":[{"node_id":0,"record":14,"sc":"Play... |
| 14 | 13 | 12 | {"h":4,"nnodes":7,"ts":"1011000","nodes":[{"node_id":0,"record":14,"sc":"Pl... |
| 15 | 14 | 13 | {"h":4,"nnodes":7,"ts":"1011000","nodes":[{"node_id":0,"record":14,"sc":"Pl... |
| 16 | 15 | 14 | {"h":4,"nnodes":11,"ts":"11100010100","nodes":[{"node_id":0,"record":14,"... |
| 17 | 16 | 15 | {"h":4,"nnodes":11,"ts":"11100010100","nodes":[{"node_id":0,"record":14,"... |
| 18 | 17 | 16 | {"h":3,"nnodes":5,"ts":"10100","nodes":[{"node_id":0,"record":14,"sc":"Play... |
| 19 | 18 | 17 | {"h":3,"nnodes":5,"ts":"10100","nodes":[{"node_id":0,"record":14,"sc":"Play... |
| 20 | 19 | 18 | {"h":6,"nnodes":13,"ts":"1111010010000","nodes":[{"node_id":0,"record":1... |
| 21 | 20 | 19 | {"h":6,"nnodes":13,"ts":"1111010010000","nodes":[{"node_id":0,"record":1... |
| 22 | 21 | 20 | {"h":3,"nnodes":7,"ts":"1100100","nodes":[{"node_id":0,"record":14,"sc":"Pl... |
| 23 | 22 | 21 | {"h":3,"nnodes":7,"ts":"1100100","nodes":[{"node_id":0,"record":14,"sc":"Pl... |

## _SYS_AFL.PAL_RANDOM_DECISION_TREES_PREDICT

This function is used for classification or regression scoring based on random decision trees model.

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<MODEL table>` | IN |
| 3 | `<PARAMETER table>` | IN |
| 4 | `<RESULT table>` | OUT |

### Signature

### Input Table

| Table | Column | Reference | Data Type | Description | Constraint |
|---|---|---|---|---|---|
| DATA | 1st column | ID | INTEGER, BIGINT, VARCHAR, or NVARCHAR | Data ID | This must be the first column. |
| | Other columns | | INTEGER, DOUBLE, DECIMAL(p, s), VARCHAR, or NVARCHAR | Independent variable | |
| MODEL | 1st column | | INTEGER | Row index | This must be the first column. |
| | 2nd column | | INTEGER | Tree index | |
| | 3rd column | | VARCHAR or NVARCHAR | Model content | PMML format |

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| THREAD_RATIO | DOUBLE | -1 | The ratio of available threads.<br>• 0: single thread<br>• 0~1: percentage<br>• Others: heuristically determined | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| BLOCK_SIZE | INTEGER | 0 | Specifies the number of data loaded per time during scoring.<br>• 0: load all data once<br>• Others: the specified number | This parameter is for reducing memory consumption, especially as the predict data is huge, or it consists of a large number of missing independent variables. However, you might lose some efficiency. |
| MISSING_REPLACEMENT | INTEGER | 1 | Specify the missing replacement strategy:<br>• 1: marginalise each missing feature out independently<br>• 2: marginalise all missing features in an instance as a whole corresponding to each category | Valid only for classification. Strategy 2 is a much coarser approximation, it should only be applied when there are a great number of missing features and a lot of classed, which might cause memory insufficiency. |
| VERBOSE | INTEGER | 0 | Controls whether to output all classes and the corresponding confidences for each data.<br>• 0: No<br>• 1: Yes | This parameter is valid only for classification. |

### Output Table

| Table | Column | Data Type | Column Name | Description | Constraint |
|---|---|---|---|---|---|
| RESULT | 1st column | ID column type | ID column name | ID. | Must be the first column. |
| | 2nd column | NVARCHAR(100) | SCORE | Scoring result. | |

| Table | Column | Data Type | Column Name | Description | Constraint |
|---|---|---|---|---|---|
| | 3rd column | DOUBLE | CONFIDENCE | Probability for classification, or sample standard error for regression. | |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_RDT_SCORING_DATA_TBL;
CREATE COLUMN TABLE PAL_RDT_SCORING_DATA_TBL (
    "ID" INTEGER,
    "OUTLOOK" VARCHAR(20),
    "TEMP" INTEGER,
    "HUMIDITY" DOUBLE,
    "WINDY" VARCHAR(10)
);
INSERT INTO PAL_RDT_SCORING_DATA_TBL VALUES (0, 'Overcast', 75, -10000.0, 'Yes');
INSERT INTO PAL_RDT_SCORING_DATA_TBL VALUES (1, 'Rain', 78, 70.0, 'Yes');
INSERT INTO PAL_RDT_SCORING_DATA_TBL VALUES (2, 'Sunny', -10000.0, NULL, 'Yes');
INSERT INTO PAL_RDT_SCORING_DATA_TBL VALUES (3, 'Sunny', 69, 70.0, 'Yes');
INSERT INTO PAL_RDT_SCORING_DATA_TBL VALUES (4, 'Rain', NULL, 70.0, 'Yes');
INSERT INTO PAL_RDT_SCORING_DATA_TBL VALUES (5, NULL, 70, 70.0, 'Yes');
INSERT INTO PAL_RDT_SCORING_DATA_TBL VALUES (6, '***', 70, 70.0, 'Yes');
DROP TABLE  #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR(100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR(100)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 0.5, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('VERBOSE', 0, NULL, NULL);
CALL _SYS_AFL.PAL_RANDOM_DECISION_TREES_PREDICT (PAL_RDT_SCORING_DATA_TBL,
PAL_RDT_MODEL_TBL, #PAL_PARAMETER_TBL, ?);
```

### Expected Results

RESULT:

| | ID | SCORE | CONFIDENCE |
|---|---|---|---|
| 1 | 0 | Play | 0.9533333333333334 |
| 2 | 1 | Play | 0.8533333333333334 |
| 3 | 2 | Do n... | 0.9 |
| 4 | 3 | Do n... | 0.58 |
| 5 | 4 | Play | 0.6866666666666666 |
| 6 | 5 | Do n... | 0.5933333333333334 |
| 7 | 6 | Play | 0.82 |

## Related Information

Decision Trees [page 187]

# 5.2.12.1 Model Compression

In order to attain better predictive performance, random decision trees method usually requires a large number of sub-learners, which tends to grow with the size of the problem.

An increasing dataset size also results in deeper and more complex models with large size.

This model compression technique is introduced to reduce the size of model with minimum loss of the accuracy.

## Relevant Parameters

- COMPRESSION
  Specifies if the model is stored in compressed format.
- QUANTIZE_RATE
  This value decides whether to do quantization for split values of a certain continuous feature. If the largest frequency of these continuous split values is less than QUANTIZE_RATE, quantization method will be used to quantize the split values of the continuous feature.
- MAX_BITS
  The maximum number of bits to quantize continuous features. Equivalent to use $2^{MAX_BITS}$ bins to quantize the values of these continuous features. Reduce the size of bins may affect the precision of split values and the accuracy in prediction. Must be less than 31.
- FITTINGS_QUANTIZATION
  This parameter indicates whether to quantize fitting values (the values of leaves) in regression problem. It is recommended to use this technique for large dataset in regression problem.

## Example

```
SET SCHEMA DM_PAL;
DROP TABLE  PAL_RDT_DATA_TBL;
CREATE COLUMN TABLE PAL_RDT_DATA_TBL (
    "OUTLOOK" VARCHAR(20),
    "TEMP" INTEGER,
    "HUMIDITY" DOUBLE,
    "WINDY" VARCHAR(10),
    "CLASS" VARCHAR(20)
);
INSERT INTO PAL_RDT_DATA_TBL VALUES ('Sunny', 75, 70.0, 'Yes', 'Play');
INSERT INTO PAL_RDT_DATA_TBL VALUES ('Sunny', NULL, 90.0, 'Yes', 'Do not Play');
INSERT INTO PAL_RDT_DATA_TBL VALUES ('Sunny', 85, NULL, 'No', 'Do not Play');
INSERT INTO PAL_RDT_DATA_TBL VALUES ('Sunny', 72, 95.0, 'No', 'Do not Play');
```

```
INSERT INTO PAL_RDT_DATA_TBL VALUES (NULL, NULL, 70.0, NULL, 'Play');
INSERT INTO PAL_RDT_DATA_TBL VALUES ('Overcast', 72.0, 90, 'Yes', 'Play');
INSERT INTO PAL_RDT_DATA_TBL VALUES ('Overcast', 83.0, 78, 'No', 'Play');
INSERT INTO PAL_RDT_DATA_TBL VALUES ('Overcast', 64.0, 65, 'Yes', 'Play');
INSERT INTO PAL_RDT_DATA_TBL VALUES ('Overcast', 81.0, 75, 'No', 'Play');
INSERT INTO PAL_RDT_DATA_TBL VALUES (NULL, 71, 80.0, 'Yes', 'Do not Play');
INSERT INTO PAL_RDT_DATA_TBL VALUES ('Rain', 65, 70.0, 'Yes', 'Do not Play');
INSERT INTO PAL_RDT_DATA_TBL VALUES ('Rain', 75, 80.0, 'No', 'Play');
INSERT INTO PAL_RDT_DATA_TBL VALUES ('Rain', 68, 80.0, 'No', 'Play');
INSERT INTO PAL_RDT_DATA_TBL VALUES ('Rain', 70, 96.0, 'No', 'Play');
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (100)
);
INSERT INTO PAL_PARAMETER_TBL VALUES ('TREES_NUM', 300, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('TRY_NUM', 3, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('SEED', 2, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('SPLIT_THRESHOLD', NULL, 1e-5, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('CALCULATE_OOB', 1, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('NODE_SIZE', 1, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 1.0, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('COMPRESSION', 1, NULL, NULL);
--INSERT INTO PAL_PARAMETER_TBL VALUES ('MAX_BITS', 12, NULL, NULL);
--INSERT INTO PAL_PARAMETER_TBL VALUES ('QUANTIZE_RATE', NULL, 0.005, NULL);
--INSERT INTO PAL_PARAMETER_TBL VALUES ('FITTINGS_QUANTIZATION', 12, NULL, NULL);
DROP TABLE PAL_RDT_MODEL_TBL;  -- for predict followed
CREATE COLUMN TABLE PAL_RDT_MODEL_TBL (
    "ROW_INDEX" INTEGER,
    "TREE_INDEX" INTEGER,
    "MODEL_CONTENT" NVARCHAR(5000)
);
do begin
    lt_data_train = select * from PAL_RDT_DATA_TBL;
    param_table =   select * from PAL_PARAMETER_TBL;

    CALL _SYS_AFL.PAL_RANDOM_DECISION_TREES (:lt_data_train, :param_table,
model_table, var_imp, out_of_bag, confusion);
    TRUNCATE TABLE PAL_RDT_MODEL_TBL;
    insert into PAL_RDT_MODEL_TBL select * from :model_table;
    select * from PAL_RDT_MODEL_TBL;
end;
```

### Expected Result

| | ROW_INDEX | TREE_INDEX | MODEL_CONTENT |
|---|---|---|---|
| 1 | 0 | -1 | {"atFields":[{"dataType":"string","name":"OUTLOOK","opType":"cat","valu... |
| 2 | 1 | 0 | {"cw":"RI\u001e\u0007A\u0012RS\f,,8SE\u0014Si\u0007R\fLL1))\u0012J... |
| 3 | 2 | 1 | {"cw":"P]9!:pf\u0018\u0019F\u0005%3x\nJgpOOSss[t ;>D\u0002p^\u00... |
| 4 | 3 | 2 | {"cw":")1\u0005\u0016\u0010;f;fknaW]F\u000biBz6\u0002d6\u0002d\f_\... |
| 5 | 4 | 3 | {"at":0,"cvs":["\u0002\u0001","\u0000","\u0001","\u0002","\u0002\u0000... |
| 6 | 5 | 3 | 4\u0001\u0018,\u0011IF&36\u0019\fLf0\b\u00026\u0005\u001b","idict"... |
| 7 | 6 | 4 | {"at":1,"cvs":["73.5","84","72","80","73","66.5","64.5","70.5","67.5","69","71.... |
| 8 | 7 | 4 | \u0002\u0002\u0002\u0002\u0002\u0002\u0002\u0002\u0002\u0002\u0002\u... |
| 9 | 8 | 5 | {"at":2,"cvs":["83","75","72.5","79","92.5","77.5","82.5","80","95.5","74","87.... |
| 10 | 9 | 5 | u012b"}]} |
| 11 | 10 | 6 | {"at":3,"cvs":["\u0001","\u0000"],"dicts":[[{"p":0.5,"v":0},{"p":0.5,"v":1}],[{"p... |

## 5.2.13 Support Vector Machine

Support Vector Machines (SVMs) refer to a family of supervised learning models using the concept of support vector.

Compared with many other supervised learning models, SVMs have the advantages in that the models produced by SVMs can be either linear or non-linear, where the latter is realized by a technique called Kernel Trick.

Like most supervised models, there are training phase and testing phase for SVMs. In the training phase, a function `f(x):->y` where `f(·)` is a function (can be non-linear) mapping a sample onto a TARGET, is learnt. The training set consists of pairs denoted by $\{x_i, y_i\}$, where `x` denotes a sample represented by several attributes, and `y` denotes a TARGET (supervised information). In the testing phase, the learnt `f(·)` is further used to map a sample with unknown TARGET onto its predicted TARGET.

In the current implementation in PAL, SVMs can be used for the following four tasks:

- **Support Vector Classification (SVC)**
  Classification is one of the most frequent tasks in many fields including machine learning, data mining, computer vision, and business data analysis. Compared with linear classifiers like logistic regression, SVC is able to produce non-linear decision boundary, which leads to better accuracy on some real world dataset. In classification scenario, `f(·)` refers to decision function, and a TARGET refers to a "label" represented by a real number.
- **Support Vector Regression (SVR)**
  SVR is another method for regression analysis. Compared with classical linear regression methods like least square regression, the regression function in SVR can be non-linear. In regression scenario, `f(·)` refers to regression function, and TARGET refers to "response" represented by a real number.
- **Support Vector Ranking**
  This implements a pairwise "learning to rank" algorithm which learns a ranking function from several sets (distinguished by Query ID) of ranked samples. In the scenario of ranking, `f(·)` refers to ranking function, and TARGET refers to score, according to which the final ranking is made. For pairwise ranking, `f(·)` is learnt so that the pairwise relationship expressing the rank of the samples within each set is considered.
- **One Class SVM**
  One class SVM is an unsupervised algorithm that learns a decision function for outlier detection: classifying new data as similar or different to the training set. In One class SVM scenario, `f(·)` refers to decision function, and there is no TARGET needed since it is unsupervised.

Because non-linearity is realized by Kernel Trick, besides the datasets, the kernel type and parameters should be specified as well.

### Prerequisite

No missing or null data in the inputs.

### _SYS_AFL.PAL_SVM

This function reads the input data and generates a training model.

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<MODEL table>` | OUT |
| 4 | `<STATISTIC table>` | OUT |
| 5 | `<OPTIMAL PARAM table>` | OUT |

### Input Table

| Table | Column | Column Data Type | Description | Constraint |
|---|---|---|---|---|
| DATA | 1st column | INTEGER, VARCHAR, or NVARCHAR | ID | This must be the first column. |
| | Feature columns | INTEGER, DOUBLE, DECIMAL(p,s), VARCHAR, or NVARCHAR | Features data | |
| | Query ID column | VARCHAR or NVARCHAR | Query ID | Only needed for SVM ranking. |
| | Dependent variable columns | INTEGER, DOUBLE, DECIMAL(p,s), or NVARCHAR | Dependent variable | Not needed for one class SVM. |

### Parameter Table

Mandatory Parameter

The following parameter is mandatory and must be given a value.

| Name | Data Type | Description |
|---|---|---|
| TYPE | INTEGER | SVM type:<br>• 1: SVC (for classification)<br>• 2: SVR (for regression)<br>• 3: Support Vector Ranking (for ranking)<br>• 4: One Class SVM (for outlier detection) |

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| KERNEL_TYPE | INTEGER | 2 | Kernel type:<br>• 0: LINEAR KERNEL<br>• 1: POLY KERNEL<br>• 2: RBF KERNEL<br>• 3: SIGMOID KERNEL | |
| THREAD_RATIO | DOUBLE | 0.0 | Specifies the ratio of total number of threads that can be used by this function. The value range is from 0 to 1, where 0 means only using 1 thread, and 1 means using at most all the currently available threads. Values outside the range will be ignored and this function heuristically determines the number of threads to use. | Only valid for parameter selection task. |
| POLY_DEGREE | INTEGER | 3 | Coefficient for the PLOY KERNEL type.<br>Value range: ≥ 1 | Only valid when KERNEL_TYPE is 1. |
| RBF_GAMMA | DOUBLE | 1.0/number of features in the dataset | Coefficient for the RBF KERNEL type.<br>Value range: > 0 | Only valid when KERNEL_TYPE is 2. |
| NU | DOUBLE | 0.5 | The value for both the upper bound of the fraction of training errors and the lower bound of the fraction of support vectors. | Only valid when TYPE is 4. |
| COEF_LIN | DOUBLE | 0.0 | Coefficient for the POLY/SIGMOID KERNEL type. | Only valid when KERNEL_TYPE is 1 or 3. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| COEF_CONST | DOUBLE | 0.0 | Coefficient for the POLY/SIGMOID KERNEL type. | Only valid when KERNEL_TYPE is 1 or 3. |
| SVM_C | DOUBLE | 100.0 | Trade-off between training error and margin.<br>Value range: > 0 | |
| REGRESSION_EPS | DOUBLE | 0.1 | Epsilon width of tube for regression.<br>Value range: > 0 | Only valid when TYPE is 2. |
| SCALE_INFO | INTEGER | 1 | • 0: No scale<br>• 1: Standardization<br>The algorithm transforms the data to have zero mean and unit variance. The general formula is given as:<br>$x_{new}=\frac{x-\mu}{\sigma}$,<br>where x is the origin feature vector, μ is the mean of that feature vector, and σ is its standard deviation.<br>• 2: Rescaling<br>The algorithm rescales the range of the features to scale the range in [-1,1]. The general formula is given as:<br>$x_{new}=-1+\frac{x-\min(x)}{\max(x)-\min(x)}*2$,<br>where x is the origin feature vector. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| SCALE_LABEL | INTEGER | 1 | Controls whether to standardize the label for SVR.<br>• 0: No<br>• 1: Yes | Only valid when TYPE is 2 and SCALE_INFO is 1. |
| SHRINK | INTEGER | 1 | Decides whether to use shrink strategy or not:<br>• 0: Does not use shrink strategy<br>• 1: Uses shrink strategy | Using shrink strategy may accelerate the training process. |
| HAS_ID | INTEGER | 1 | Specifies whether the input data table has an ID column.<br>• 0: No<br>• 1: Yes | |
| HANDLE_MISSING | INTEGER | 1 | Whether to handle missing values:<br>• 0: No<br>• 1: Yes | |
| CATEGORICAL_VARIABLE | VARCHAR | No default value | Indicates whether or not a column data is actually corresponding to a category variable even the data type of this column is INTEGER.<br>By default, 'VARCHAR' or 'NVARCHAR' is category variable, and 'INTEGER' or 'DOUBLE' is continuous variable. | |
| DEPENDENT_VARIABLE | VARCHAR | No default value | Column name in the data table used as dependent variable. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| CATEGORY_WEIGHT | DOUBLE | 0.707 | Represents the weight of category attributes (γ). The value must be greater than 0. | |
| ERROR_TOL | DOUBLE | 0.001 | Specifies the error tolerance in the training process. The value must be greater than 0. | |
| PROBABILITY | INTEGER | 0 | If you want to output probability when scoring, set this to 1.<br>**Note**: Setting this to 1 will severely degrade the training performance. Use the default value (0) unless you want to output probabilities when scoring. | Only valid when TYPE is 1 or 3. |
| COMPRESSION | INTEGER | 0 | Specifies if the model is stored in compressed format.<br>• 0 : no<br>• 1: yes | |
| MAX_BITS | INTEGER | 12 | The maximum number of bits to quantize values of support vectors. Equivalent to use $2^{MAX_BITS}$ bins to quantize values of support vectors. Reduce the size of bins may affect the precision of support vectors and the accuracy in prediction. | Valid only when the value of COMPRESSION is 1. |
| MAX_QUANTIZATION_ITER | INTEGER | 1000 | The maximum iteration number of the quantization process. Too small MAX_QUANTIZATION_ITER may result in not converge of quantization and affects the accuracy further. | Valid only when the value of COMPRESSION is 1. |

Model Evaluation and Parameter Selection

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| RESAMPLING_METHOD | NVARCHAR | No default value | Specifies the resampling method for model evaluation or parameter selection.<br>• cv<br>• cv_sha<br>• cv_hyperband<br>• stratified_cv<br>• stratified_cv_sha<br>• stratified_cv_hyperband<br>• bootstrap<br>• bootstrap_sha<br>• bootstrap_hyperband<br>• stratified_bootstrap<br>• stratified_bootstrap_sha<br>• stratified_bootstrap_hyperband<br>stratified_* can only apply to classification algorithms. If no value is specified for this parameter, neither model evaluation nor parameter selection is activated. | |
| EVALUATION_METRIC | NVARCHAR | No default value | Specifies the evaluation metric for model evaluation or parameter selection.<br>• ACCURACY<br>• F1_SCORE<br>• AUC<br>• RMSE<br>• ERROR_RATE | RMSE only valid for SVR; ERROR_RATE only valid for SVM-Ranking; ACCURACY only valid for SVC and One-Class SVM; F1_SCORE and AUC only valid for SVC. |
| FOLD_NUM | INTEGER | No default value | Specifies the fold number for the cross validation method. | Mandatory and valid only when RESAMPLING_METHOD is set to cv or stratified_cv. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| REPEAT_TIMES | INTEGER | 1 | Specifies the number of repeat times for re-sampling. | |
| PARAM_SEARCH_STRATEGY | NVARCHAR | No default value | Specify the parameter search method:<br>• grid<br>• random | When *_sha is specified in RESAMPLING_METHOD, PARAM_SEARCH_STRATEGY is mandatory. When *_hyperband is specified in RESAMPLING_METHOD, PARAM_SEARCH_STRATEGY is set to random whatever other value is provided. |
| RANDOM_SEARCH_TIMES | INTEGER | No default value | Specifies the number of times to randomly select candidate parameters for selection. | Mandatory and valid when PARAM_SEARCH_STRATEGY is set to random. |
| REDUCTION_RATE | DOUBLE | 3.0 | Specifies reduction rate in SHA or Hyperband method. | Only valid when RESAMPLING_METHOD is specified as *_sha or *_hyberband. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| AGGRESSIVE_ELIMINATION | INTEGER | | Specifies whether to apply aggressive elimination while using SHA method. Aggressive elimination happens when the data size and parameters size to be searched does not match and there are still bunch of parameters to be searched while data size reaches its upper limits. If aggressive elimination is applied, lower bound of limit of data size will be used multiple times first to reduce number of parameters.<br>• 0: doe not apply aggressive elimination<br>• 1: applies aggressive elimination | Only valid when RESAMPLING_METHOD is specified as *_sha. |
| SEED | INTEGER | 0 | Specifies the seed for random generation. Use system time when 0 is specified. | |
| TIMEOUT | INTEGER | 0 | Specifies maximum running time for model evaluation or parameter selection, in seconds. No timeout when 0 is specified. | |
| PROGRESS_INDICATOR_ID | NVARCHAR | No default value | Sets an ID of progress indicator for model evaluation or parameter selection. No progress indicator is active if no value is provided. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| SVM_C_VALUES | NVARCHAR | No default value | Specifies values of SVM_C to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| SVM_C_RANGE | NVARCHAR | No default value | Specifies the range of SVM_C to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| NU_VALUES | NVARCHAR | No default value | Specifies values of NU to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| NU_RANGE | NVARCHAR | No default value | Specifies the range of NU to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| RBF_GAMMA_VALUES | NVARCHAR | No default value | Specifies values of RBF_GAMMA to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| RBF_GAMMA_RANGE | NVARCHAR | No default value | Specifies the range of RBF_GAMMA to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| POLY_DEGREE_VALUES | NVARCHAR | No default value | Specifies values of POLY_DEGREE to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| POLY_DEGREE_RANGE | NVARCHAR | No default value | Specifies the range of POLY_DEGREE to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| COEF_LIN_VALUES | NVARCHAR | No default value | Specifies values of COEF_LIN to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| COEF_LIN_RANGE | NVARCHAR | No default value | Specifies the range of COEF_LIN to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| COEF_CONST_VALUES | NVARCHAR | No default value | Specifies values of COEF_CONST to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| COEF_CONST_RANGE | NVARCHAR | No default value | Specifies the range of COEF_CONST to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |

### Output Tables

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| MODEL | 1st column | INTEGER | ROW_INDEX | ID |
| | 2nd column | NVARCHAR(5000) | MODEL_CONTENT | Model content |
| STATISTIC | 1st column | NVARCHAR(256) | STAT_NAME | |
| | 2nd column | NCLOB | STAT_VALUE | |
| OPTIMAL_PARAM | 1st column | NVARCHAR(256) | PARAM_NAME | |
| | 2nd column | INTEGER | INT_VALUE | |
| | 3rd column | DOUBLE | DOUBLE_VALUE | |
| | 4th column | NVARCHAR(1000) | STRING_VALUE | |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

**Example 1: Support vector classification**

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_SVM_DATA_TBL;
CREATE COLUMN TABLE PAL_SVM_DATA_TBL (
    ID INTEGER,
    ATTRIBUTE1 DOUBLE,
    ATTRIBUTE2 DOUBLE,
    ATTRIBUTE3 DOUBLE,
    ATTRIBUTE4 VARCHAR(10),
    LABEL INTEGER
);
INSERT INTO PAL_SVM_DATA_TBL VALUES (0,1,10,100,'A',1);
INSERT INTO PAL_SVM_DATA_TBL VALUES (1,1.1,10.1,100,'A',1);
INSERT INTO PAL_SVM_DATA_TBL VALUES (2,1.2,10.2,100,'A',1);
INSERT INTO PAL_SVM_DATA_TBL VALUES (3,1.3,10.4,100,'A',1);
INSERT INTO PAL_SVM_DATA_TBL VALUES (4,1.2,10.3,100,'AB',1);
INSERT INTO PAL_SVM_DATA_TBL VALUES (5,4,40,400,'AB',2);
INSERT INTO PAL_SVM_DATA_TBL VALUES (6,4.1,40.1,400,'AB',2);
INSERT INTO PAL_SVM_DATA_TBL VALUES (7,4.2,40.2,400,'AB',2);
INSERT INTO PAL_SVM_DATA_TBL VALUES (8,4.3,40.4,400,'AB',2);
INSERT INTO PAL_SVM_DATA_TBL VALUES (9,4.2,40.3,400,'AB',2);
INSERT INTO PAL_SVM_DATA_TBL VALUES (10,9,90,900,'B',3);
INSERT INTO PAL_SVM_DATA_TBL VALUES (11,9.1,90.1,900,'A',3);
INSERT INTO PAL_SVM_DATA_TBL VALUES (12,9.2,90.2,900,'B',3);
INSERT INTO PAL_SVM_DATA_TBL VALUES (13,9.3,90.4,900,'A',3);
INSERT INTO PAL_SVM_DATA_TBL VALUES (14,9.2,90.3,900,'A',3);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('KERNEL_TYPE',2,NULL,NULL);
```

```
INSERT INTO #PAL_PARAMETER_TBL VALUES ('TYPE',1,NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('RBF_GAMMA',NULL,0.005,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('HAS_ID',1,NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('HANDLE_MISSING', 0, NULL, NULL);
DROP TABLE PAL_SVM_MODEL_TBL_EX1;
CREATE COLUMN TABLE PAL_SVM_MODEL_TBL_EX1 (ROW_INDEX INTEGER, MODEL_CONTENT
NVARCHAR(5000));
CALL
_SYS_AFL.PAL_SVM(PAL_SVM_DATA_TBL,#PAL_PARAMETER_TBL,PAL_SVM_MODEL_TBL_EX1,?,?)
WITH OVERVIEW;
```

**Expected Result**

MODEL:

| | ROW_INDEX | MODEL_CONTENT |
|---|---|---|
| 1 | 0 | {"C":100.0,"attr_number":4,"bias":[-0.01412852776414786,0.00503273643... |

STATISTIC:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | obj | "1<=>2":-91.4992,"1<=>3":-12.908,"2<=>3":-29.3739 |
| 2 | iter | "1<=>2":2,"1<=>3":12,"2<=>3":7 |

**Example 2: Support vector ranking**

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_SVM_DATA_TBL;
CREATE COLUMN TABLE PAL_SVM_DATA_TBL (
    ID INTEGER,
    ATTRIBUTE1 DOUBLE,
    ATTRIBUTE2 DOUBLE,
    ATTRIBUTE3 DOUBLE,
    ATTRIBUTE4 DOUBLE,
    ATTRIBUTE5 DOUBLE,
    QID VARCHAR(50),
    LABEL INTEGER
);
INSERT INTO PAL_SVM_DATA_TBL VALUES(0,1,1,0,0.2,0,'qid:1',3);
INSERT INTO PAL_SVM_DATA_TBL VALUES(1,0,0,1,0.1,1,'qid:1',2);
INSERT INTO PAL_SVM_DATA_TBL VALUES(2,0,0,1,0.3,0,'qid:1',1);
INSERT INTO PAL_SVM_DATA_TBL VALUES(3,2,1,1,0.2,0,'qid:1',4);
INSERT INTO PAL_SVM_DATA_TBL VALUES(4,3,1,1,0.4,1,'qid:1',5);
INSERT INTO PAL_SVM_DATA_TBL VALUES(5,4,1,1,0.7,0,'qid:1',6);
INSERT INTO PAL_SVM_DATA_TBL VALUES(6,0,0,1,0.2,0,'qid:2',1);
INSERT INTO PAL_SVM_DATA_TBL VALUES(7,1,0,1,0.4,0,'qid:2',2);
INSERT INTO PAL_SVM_DATA_TBL VALUES(8,0,0,1,0.2,0,'qid:2',1);
INSERT INTO PAL_SVM_DATA_TBL VALUES(9,1,1,1,0.2,0,'qid:2',3);
INSERT INTO PAL_SVM_DATA_TBL VALUES(10,2,2,1,0.4,0,'qid:2',4);
INSERT INTO PAL_SVM_DATA_TBL VALUES(11,3,3,1,0.2,0,'qid:2',5);
INSERT INTO PAL_SVM_DATA_TBL VALUES(12,0,0,1,0.1,1,'qid:3',2);
INSERT INTO PAL_SVM_DATA_TBL VALUES(13,1,0,0,0.4,1,'qid:3',4);
INSERT INTO PAL_SVM_DATA_TBL VALUES(14,0,1,1,0.5,0,'qid:3',1);
INSERT INTO PAL_SVM_DATA_TBL VALUES(15,1,1,1,0.5,1,'qid:3',3);
INSERT INTO PAL_SVM_DATA_TBL VALUES(16,2,2,0,0.7,1,'qid:3',5);
INSERT INTO PAL_SVM_DATA_TBL VALUES(17,1,3,1,1.5,0,'qid:3',6);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
```

```
    "STRING_VALUE" NVARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES('KERNEL_TYPE',2,NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES('TYPE',3,NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES('RBF_GAMMA',NULL,0.005,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES('HANDLE_MISSING', 0, NULL, NULL);
DROP TABLE PAL_SVM_MODEL_TBL_EX2;
CREATE COLUMN TABLE PAL_SVM_MODEL_TBL_EX2 (ROW_INDEX INTEGER, MODEL_CONTENT
NVARCHAR(5000));
CALL _SYS_AFL.PAL_SVM
(PAL_SVM_DATA_TBL,#PAL_PARAMETER_TBL,PAL_SVM_MODEL_TBL_EX2,?,?) WITH OVERVIEW;
```

**Expected Result**

MODEL:

| | ROW_INDEX | MODEL_CONTENT |
|---|---|---|
| 1 | 0 | {"C":100.0,"attr_number":5,"bias":[0.0],"categorical_attr_number":0,"class_number":2,"contin... |

STATISTIC:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | obj | -531.5556524719772 |
| 2 | iter | 112 |

**Example 3: Support vector classification (training a model with additional information that can be used when calculating scoring probability)**

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_SVM_DATA_TBL;
CREATE COLUMN TABLE PAL_SVM_DATA_TBL (
    ID INTEGER,
    ATTRIBUTE1 DOUBLE,
    ATTRIBUTE2 DOUBLE,
    ATTRIBUTE3 DOUBLE,
    ATTRIBUTE4 VARCHAR(10),
    LABEL INTEGER
);
INSERT INTO PAL_SVM_DATA_TBL VALUES(0,1,10,100,'A',1);
INSERT INTO PAL_SVM_DATA_TBL VALUES(1,1.1,10.1,100,'A',1);
INSERT INTO PAL_SVM_DATA_TBL VALUES(2,1.2,10.2,100,'A',1);
INSERT INTO PAL_SVM_DATA_TBL VALUES(3,1.3,10.4,100,'A',1);
INSERT INTO PAL_SVM_DATA_TBL VALUES(4,1.2,10.3,100,'AB',1);
INSERT INTO PAL_SVM_DATA_TBL VALUES(5,4,40,400,'AB',1);
INSERT INTO PAL_SVM_DATA_TBL VALUES(6,4.1,40.1,400,'AB',1);
INSERT INTO PAL_SVM_DATA_TBL VALUES(7,4.2,40.2,400,'AB',1);
INSERT INTO PAL_SVM_DATA_TBL VALUES(8,4.3,40.4,400,'AB',2);
INSERT INTO PAL_SVM_DATA_TBL VALUES(9,4.2,40.3,400,'AB',2);
INSERT INTO PAL_SVM_DATA_TBL VALUES(10,9,90,900,'B',2);
INSERT INTO PAL_SVM_DATA_TBL VALUES(11,9.1,90.1,900,'A',2);
INSERT INTO PAL_SVM_DATA_TBL VALUES(12,9.2,90.2,900,'B',2);
INSERT INTO PAL_SVM_DATA_TBL VALUES(13,9.3,90.4,900,'A',2);
INSERT INTO PAL_SVM_DATA_TBL VALUES(14,9.2,90.3,900,'A',2);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES('KERNEL_TYPE',2,NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES('TYPE',1,NULL,NULL);
```

```
INSERT INTO #PAL_PARAMETER_TBL VALUES('RBF_GAMMA',NULL,0.005,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES('PROBABILITY',1,NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES('HANDLE_MISSING', 0, NULL, NULL);
DROP TABLE PAL_SVM_MODEL_TBL_EX3;
CREATE COLUMN TABLE PAL_SVM_MODEL_TBL_EX3 (ROW_INDEX INTEGER, MODEL_CONTENT
NVARCHAR(5000));
CALL _SYS_AFL.PAL_SVM
(PAL_SVM_DATA_TBL,#PAL_PARAMETER_TBL,PAL_SVM_MODEL_TBL_EX3,?,?) WITH OVERVIEW;
```

**Expected Result**

MODEL:

| | ROW_INDEX | MODEL_CONTENT |
|---|---|---|
| 1 | 0 | {"C":100.0,"attr_number":4,"bias":[-0.01337641155651498],"categorical_a... |

STATISTIC:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | obj | "1<=>2":-425.301 |
| 2 | iter | "1<=>2":7 |

**Example 4: One class SVM**

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_SVM_DATA_TBL;
CREATE COLUMN TABLE PAL_SVM_DATA_TBL (
    ID INTEGER,
    ATTRIBUTE1 DOUBLE,
    ATTRIBUTE2 DOUBLE,
    ATTRIBUTE3 DOUBLE,
    ATTRIBUTE4 VARCHAR(50)
);
INSERT INTO PAL_SVM_DATA_TBL VALUES(0,1,10,100,'A');
INSERT INTO PAL_SVM_DATA_TBL VALUES(1,1.1,10.1,100,'A');
INSERT INTO PAL_SVM_DATA_TBL VALUES(2,1.2,10.2,100,'A');
INSERT INTO PAL_SVM_DATA_TBL VALUES(3,1.3,10.4,100,'A');
INSERT INTO PAL_SVM_DATA_TBL VALUES(4,1.2,10.3,100,'AB');
INSERT INTO PAL_SVM_DATA_TBL VALUES(5,4,40,400,'AB');
INSERT INTO PAL_SVM_DATA_TBL VALUES(6,4.1,40.1,400,'AB');
INSERT INTO PAL_SVM_DATA_TBL VALUES(7,4.2,40.2,400,'AB');
INSERT INTO PAL_SVM_DATA_TBL VALUES(8,4.3,40.4,400,'AB');
INSERT INTO PAL_SVM_DATA_TBL VALUES(9,4.2,40.3,400,'AB');
INSERT INTO PAL_SVM_DATA_TBL VALUES(10,9,90,900,'B');
INSERT INTO PAL_SVM_DATA_TBL VALUES(11,9.1,90.1,900,'A');
INSERT INTO PAL_SVM_DATA_TBL VALUES(12,9.2,90.2,900,'B');
INSERT INTO PAL_SVM_DATA_TBL VALUES(13,9.3,90.4,900,'A');
INSERT INTO PAL_SVM_DATA_TBL VALUES(14,9.2,90.3,900,'A');
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES('KERNEL_TYPE',2,NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES('TYPE',4,NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES('CATEGORY_WEIGHT',NULL,1,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES('SCALE_INFO',0,NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES('HANDLE_MISSING',0,NULL,NULL);
--INSERT INTO #PAL_PARAMETER_TBL VALUES('RBF_GAMMA',NULL,0.166667,NULL);
```

```
--INSERT INTO #PAL_PARAMETER_TBL VALUES('NU',NULL,0.5,NULL);
DROP TABLE PAL_SVM_MODEL_TBL_EX4;
CREATE COLUMN TABLE PAL_SVM_MODEL_TBL_EX4 (ROW_INDEX INTEGER, MODEL_CONTENT
NVARCHAR(5000));
CALL _SYS_AFL.PAL_SVM
(PAL_SVM_DATA_TBL,#PAL_PARAMETER_TBL,PAL_SVM_MODEL_TBL_EX4,?,?) WITH OVERVIEW;
```

**Expected Result**

MODEL:

| | ROW_INDEX | MODEL_CONTENT |
|---|---|---|
| 1 | 0 | {"C":1.0,"attr_number":4,"bias":[2.250649435837215],"categorical_attr_nu... |

STATISTIC:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | obj | 8.338665524190917 |
| 2 | iter | 16 |

**Example 5: Model evaluation**

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_SVM_DATA_TBL;
CREATE COLUMN TABLE PAL_SVM_DATA_TBL (
    ID INTEGER,
    ATTRIBUTE1 DOUBLE,
    ATTRIBUTE2 DOUBLE,
    ATTRIBUTE3 DOUBLE,
    ATTRIBUTE4 VARCHAR(10),
    LABEL INTEGER
);
INSERT INTO PAL_SVM_DATA_TBL VALUES (0,1,10,100,'A',1);
INSERT INTO PAL_SVM_DATA_TBL VALUES (1,1.1,10.1,100,'A',1);
INSERT INTO PAL_SVM_DATA_TBL VALUES (2,1.2,10.2,100,'A',1);
INSERT INTO PAL_SVM_DATA_TBL VALUES (3,1.3,10.4,100,'A',1);
INSERT INTO PAL_SVM_DATA_TBL VALUES (4,1.2,10.3,100,'AB',1);
INSERT INTO PAL_SVM_DATA_TBL VALUES (5,4,40,400,'AB',2);
INSERT INTO PAL_SVM_DATA_TBL VALUES (6,4.1,40.1,400,'AB',2);
INSERT INTO PAL_SVM_DATA_TBL VALUES (7,4.2,40.2,400,'AB',2);
INSERT INTO PAL_SVM_DATA_TBL VALUES (8,4.3,40.4,400,'AB',2);
INSERT INTO PAL_SVM_DATA_TBL VALUES (9,4.2,40.3,400,'AB',2);
INSERT INTO PAL_SVM_DATA_TBL VALUES (10,9,90,900,'B',3);
INSERT INTO PAL_SVM_DATA_TBL VALUES (11,9.1,90.1,900,'A',3);
INSERT INTO PAL_SVM_DATA_TBL VALUES (12,9.2,90.2,900,'B',3);
INSERT INTO PAL_SVM_DATA_TBL VALUES (13,9.3,90.4,900,'A',3);
INSERT INTO PAL_SVM_DATA_TBL VALUES (14,9.2,90.3,900,'A',3);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('KERNEL_TYPE',2,NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('TYPE',1,NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('RBF_GAMMA',NULL,0.005,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('HAS_ID',1,NULL,NULL);
--model evaluation parameters
INSERT INTO #PAL_PARAMETER_TBL VALUES ('RESAMPLING_METHOD', NULL, NULL, 'cv');
```

```
INSERT INTO #PAL_PARAMETER_TBL VALUES ('EVALUATION_METRIC', NULL, NULL,
'F1_SCORE');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FOLD_NUM', 2, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('REPEAT_TIMES', 2, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SEED', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PROGRESS_INDICATOR_ID', NULL, NULL,
'TEST');

DROP TABLE PAL_SVM_MODEL_TBL_EX1;
CREATE COLUMN TABLE PAL_SVM_MODEL_TBL_EX1 (ROW_INDEX INTEGER, MODEL_CONTENT
NVARCHAR(5000));
CALL
_SYS_AFL.PAL_SVM(PAL_SVM_DATA_TBL,#PAL_PARAMETER_TBL,PAL_SVM_MODEL_TBL_EX1,?,?)
WITH OVERVIEW;
```

**Expected Result**

STATISTICS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | timeout | FALSE |
| 2 | TEST_1_F1_SCORE | 1, 1 |
| 3 | TEST_2_F1_SCORE | 1, 1 |
| 4 | TEST_F1_SCORE.... | 1 |
| 5 | TEST_F1_SCORE.... | 0 |
| 6 | EVAL_RESULTS_1 | {"candidates":[{"TEST_F1_... |
| 7 | obj | "1<=>2":-91.4992,"1<=>... |
| 8 | iter | "1<=>2":2,"1<=>3":12,"2... |

**Example 6: parameter selection**

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_SVM_DATA_TBL;
CREATE COLUMN TABLE PAL_SVM_DATA_TBL (
    ID INTEGER,
    ATTRIBUTE1 DOUBLE,
    ATTRIBUTE2 DOUBLE,
    ATTRIBUTE3 DOUBLE,
    ATTRIBUTE4 VARCHAR(10),
    LABEL INTEGER
);
INSERT INTO PAL_SVM_DATA_TBL VALUES (0,1,10,100,'A',1);
INSERT INTO PAL_SVM_DATA_TBL VALUES (1,1.1,10.1,100,'A',1);
INSERT INTO PAL_SVM_DATA_TBL VALUES (2,1.2,10.2,100,'A',1);
INSERT INTO PAL_SVM_DATA_TBL VALUES (3,1.3,10.4,100,'A',1);
INSERT INTO PAL_SVM_DATA_TBL VALUES (4,1.2,10.3,100,'AB',1);
INSERT INTO PAL_SVM_DATA_TBL VALUES (5,4,40,400,'AB',2);
INSERT INTO PAL_SVM_DATA_TBL VALUES (6,4.1,40.1,400,'AB',2);
INSERT INTO PAL_SVM_DATA_TBL VALUES (7,4.2,40.2,400,'AB',2);
INSERT INTO PAL_SVM_DATA_TBL VALUES (8,4.3,40.4,400,'AB',2);
INSERT INTO PAL_SVM_DATA_TBL VALUES (9,4.2,40.3,400,'AB',2);
INSERT INTO PAL_SVM_DATA_TBL VALUES (10,9,90,900,'B',3);
INSERT INTO PAL_SVM_DATA_TBL VALUES (11,9.1,90.1,900,'A',3);
INSERT INTO PAL_SVM_DATA_TBL VALUES (12,9.2,90.2,900,'B',3);
INSERT INTO PAL_SVM_DATA_TBL VALUES (13,9.3,90.4,900,'A',3);
INSERT INTO PAL_SVM_DATA_TBL VALUES (14,9.2,90.3,900,'A',3);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
```

```
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('KERNEL_TYPE',2,NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('TYPE',1,NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('HAS_ID',1,NULL,NULL);
--parameters to be selected
INSERT INTO #PAL_PARAMETER_TBL VALUES ('RBF_GAMMA_VALUES',NULL,NULL,
'{0.01,0.05,0.07}');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SVM_C_VALUES',NULL,NULL, '{10,50,100}');
--parameter selection parameters
INSERT INTO #PAL_PARAMETER_TBL VALUES ('RESAMPLING_METHOD', NULL, NULL, 'cv');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('EVALUATION_METRIC', NULL, NULL,
'F1_SCORE');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FOLD_NUM', 2, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('REPEAT_TIMES', 2, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SEED', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PROGRESS_INDICATOR_ID', NULL, NULL,
'TEST');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PARAM_SEARCH_STRATEGY', NULL, NULL,
'grid');
DROP TABLE PAL_SVM_MODEL_TBL_EX1;
CREATE COLUMN TABLE PAL_SVM_MODEL_TBL_EX1 (ROW_INDEX INTEGER, MODEL_CONTENT
NVARCHAR(5000));
CALL
_SYS_AFL.PAL_SVM(PAL_SVM_DATA_TBL,#PAL_PARAMETER_TBL,PAL_SVM_MODEL_TBL_EX1,?,?)
WITH OVERVIEW;
```

**Expected Result**

STATISTICS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | timeout | FALSE |
| 2 | TEST_1_F1_SCORE | 1, 1 |
| 3 | TEST_2_F1_SCORE | 1, 1 |
| 4 | TEST_F1_SCORE.... | 1 |
| 5 | TEST_F1_SCORE.... | 0 |
| 6 | EVAL_RESULTS_1 | {"candidat... |
| 7 | EVAL_RESULTS_2 | 0]],"TEST_F... |
| 8 | obj | "1<=>2":-9... |
| 9 | iter | "1<=>2":3,... |

OPTIMAL PARAMETERS:

| | PARAM_NAME | INT_VALUE | DOUBLE_VALUE | STRING_VALUE |
|---|---|---|---|---|
| 1 | RBF_GAMMA | ? | 0.05 | ? |
| 2 | SVM_C | ? | 10 | ? |

## _SYS_AFL.PAL_SVM_PREDICT

This function uses the training model generated by _SYS_AFL.PAL_SVM to make predictive analysis.

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<MODEL table>` | IN |
| 3 | `<PARAMETER table>` | IN |
| 4 | `<RESULT table>` | OUT |

### Input Table

| Table | Column | Reference for Output Table | Data Type | Description |
|---|---|---|---|---|
| DATA | 1st column | ID | INTEGER, VARCHAR, or NVARCHAR | ID.<br>This must be the first column. |
| | Feature columns | | INTEGER, DOUBLE, DECIMAL(p,s), VARCHAR, or NVARCHAR | Feature data.<br>The VARCHAR or NVARCHAR data type is available only when the input model table is not null. |
| | Last column | | VARCHAR or NVARCHAR | Query ID.<br>Only needed for SVM ranking. |
| MODEL | 1st column | | INTEGER | Row index |
| | 2nd column | | NVARCHAR | Model content |

### Parameter Table

Mandatory Parameters

None.

Optional Parameter

The following parameter is optional. If it is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| THREAD_RATIO | DOUBLE | 0.0 | Specifies the ratio of total number of threads that can be used by this function. The value range is from 0 to 1, where 0 means only using 1 thread, and 1 means using at most all the currently available threads. Values outside the range will be ignored and this function heuristically determines the number of threads to use. |
| VERBOSE_OUTPUT | INTEGER | 0 | Output scoring probabilities for each class |

### Output Table

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| RESULT | 1st column | ID (in input table) data type | ID (in input table) column name | ID |
| | 2nd column | NVARCHAR(100) | SCORE | Prediction value.<br>For one class SVM, the values of -1 indicate outliers, while the values of 1 indicate normal points. |
| | 3rd column | DOUBLE | PROBABILITY | Prediction probability.<br>It is necessary when the PROBABILITY parameter is set to 1 in _SYS_AFL.PAL_SVM for SVC. |

### Examples

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

#### Example 1: Support vector classification

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_SVM_PREDICT_DATA_TBL;
```

```
CREATE COLUMN TABLE PAL_SVM_PREDICT_DATA_TBL(ID INTEGER, ATTRIBUTE1 DOUBLE,
ATTRIBUTE2 DOUBLE, ATTRIBUTE3 DOUBLE, ATTRIBUTE4 VARCHAR(10));
INSERT INTO PAL_SVM_PREDICT_DATA_TBL VALUES(0,1,10,100,'A');
INSERT INTO PAL_SVM_PREDICT_DATA_TBL VALUES(1,1.2,10.2,100,'A');
INSERT INTO PAL_SVM_PREDICT_DATA_TBL VALUES(2,4.1,40.1,400,'AB');
INSERT INTO PAL_SVM_PREDICT_DATA_TBL VALUES(3,4.2,40.3,400,'AB');
INSERT INTO PAL_SVM_PREDICT_DATA_TBL VALUES(4,9.1,90.1,900,'A');
INSERT INTO PAL_SVM_PREDICT_DATA_TBL VALUES(5,9.2,90.2,900,'A');
INSERT INTO PAL_SVM_PREDICT_DATA_TBL VALUES(6,4,40,400,'A');
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL(
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES('THREAD_RATIO',NULL,0.1,NULL);
CALL
_SYS_AFL.PAL_SVM_PREDICT(PAL_SVM_PREDICT_DATA_TBL,PAL_SVM_MODEL_TBL_EX1,#PAL_PARA
METER_TBL,?);
```

### Expected Result

RESULT:

| | ID | SCORE | PROBABILITY |
|---|---|---|---|
| 1 | 0 | 1 | ? |
| 2 | 1 | 1 | ? |
| 3 | 2 | 2 | ? |
| 4 | 3 | 2 | ? |
| 5 | 4 | 3 | ? |
| 6 | 5 | 3 | ? |
| 7 | 6 | 2 | ? |

### Example 2: Support vector ranking

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_SVM_PREDICT_DATA_TBL;
CREATE COLUMN TABLE PAL_SVM_PREDICT_DATA_TBL(
    ID INTEGER,
    ATTRIBUTE1 DOUBLE,
    ATTRIBUTE2 DOUBLE,
    ATTRIBUTE3 DOUBLE,
    ATTRIBUTE4 DOUBLE,
    ATTRIBUTE5 DOUBLE,
    QID VARCHAR(50)
);
INSERT INTO PAL_SVM_PREDICT_DATA_TBL VALUES(0,1,1,0,0.2,0,'qid:1');
INSERT INTO PAL_SVM_PREDICT_DATA_TBL VALUES(1,0,0,1,0.1,1,'qid:1');
INSERT INTO PAL_SVM_PREDICT_DATA_TBL VALUES(2,0,0,1,0.3,0,'qid:1');
INSERT INTO PAL_SVM_PREDICT_DATA_TBL VALUES(3,2,1,1,0.2,0,'qid:1');
INSERT INTO PAL_SVM_PREDICT_DATA_TBL VALUES(4,3,1,1,0.4,1,'qid:1');
INSERT INTO PAL_SVM_PREDICT_DATA_TBL VALUES(5,4,1,1,0.7,0,'qid:1');
INSERT INTO PAL_SVM_PREDICT_DATA_TBL VALUES(6,0,0,1,0.2,0,'qid:4');
INSERT INTO PAL_SVM_PREDICT_DATA_TBL VALUES(7,1,0,1,0.4,0,'qid:4');
INSERT INTO PAL_SVM_PREDICT_DATA_TBL VALUES(8,0,0,1,0.2,0,'qid:4');
INSERT INTO PAL_SVM_PREDICT_DATA_TBL VALUES(9,1,1,1,0.2,0,'qid:4');
INSERT INTO PAL_SVM_PREDICT_DATA_TBL VALUES(10,2,2,1,0.4,0,'qid:4');
INSERT INTO PAL_SVM_PREDICT_DATA_TBL VALUES(11,3,3,1,0.2,0,'qid:4');
INSERT INTO PAL_SVM_PREDICT_DATA_TBL VALUES(12,0,0,1,0.1,1,'qid:5');
INSERT INTO PAL_SVM_PREDICT_DATA_TBL VALUES(13,1,0,0,0.4,1,'qid:5');
INSERT INTO PAL_SVM_PREDICT_DATA_TBL VALUES(14,0,1,1,0.5,0,'qid:5');
```

```
INSERT INTO PAL_SVM_PREDICT_DATA_TBL VALUES(15,1,1,1,0.5,1,'qid:5');
INSERT INTO PAL_SVM_PREDICT_DATA_TBL VALUES(16,2,2,0,0.7,1,'qid:5');
INSERT INTO PAL_SVM_PREDICT_DATA_TBL VALUES(17,1,3,1,1.5,0,'qid:5');
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL(
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR (1000)
);
CALL
_SYS_AFL.PAL_SVM_PREDICT(PAL_SVM_PREDICT_DATA_TBL,PAL_SVM_MODEL_TBL_EX2,#PAL_PARA
METER_TBL,?);
```

**Expected Result**

RESULT:

| | ID | SCORE | PROBABILITY |
|---|---|---|---|
| 1 | 0 | -9.85138 | ? |
| 2 | 1 | -10.8657 | ? |
| 3 | 2 | -11.6741 | ? |
| 4 | 3 | -9.33985 | ? |
| 5 | 4 | -7.88839 | ? |
| 6 | 5 | -6.8842 | ? |
| 7 | 6 | -11.7081 | ? |
| 8 | 7 | -10.8003 | ? |
| 9 | 8 | -11.7081 | ? |
| 10 | 9 | -10.2583 | ? |
| 11 | 10 | -8.38342 | ? |
| 12 | 11 | -6.64057 | ? |
| 13 | 12 | -10.8657 | ? |
| 14 | 13 | -9.43532 | ? |
| 15 | 14 | -10.8802 | ? |
| 16 | 15 | -9.53199 | ? |
| 17 | 16 | -7.61764 | ? |
| 18 | 17 | -6.11724 | ? |

**Example 3: Support vector classification (training a model with additional information that can be used when calculating scoring probability)**

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_SVM_PREDICT_DATA_TBL;
CREATE COLUMN TABLE PAL_SVM_PREDICT_DATA_TBL( ID INTEGER, ATTRIBUTE1 DOUBLE,
ATTRIBUTE2 DOUBLE, ATTRIBUTE3 DOUBLE, ATTRIBUTE4 VARCHAR(10));
INSERT INTO PAL_SVM_PREDICT_DATA_TBL VALUES(0,1,10,100,'A');
INSERT INTO PAL_SVM_PREDICT_DATA_TBL VALUES(1,1.2,10.2,100,'A');
INSERT INTO PAL_SVM_PREDICT_DATA_TBL VALUES(2,4.1,40.1,400,'AB');
INSERT INTO PAL_SVM_PREDICT_DATA_TBL VALUES(3,4.2,40.3,400,'AB');
```

```
INSERT INTO PAL_SVM_PREDICT_DATA_TBL VALUES(4,9.1,90.1,900,'A');
INSERT INTO PAL_SVM_PREDICT_DATA_TBL VALUES(5,9.2,90.2,900,'A');
INSERT INTO PAL_SVM_PREDICT_DATA_TBL VALUES(6,4,40,400,'A');
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL(
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES('VERBOSE_OUTPUT',1,NULL,NULL);
CALL
_SYS_AFL.PAL_SVM_PREDICT(PAL_SVM_PREDICT_DATA_TBL,PAL_SVM_MODEL_TBL_EX3,#PAL_PARA
METER_TBL,?);
```

**Expected Result**

RESULT:

| | ID | SCORE | PROBABILITY |
|---|---|---|---|
| 1 | 0 | 1 | 0.913845605225848 |
| 2 | 0 | 2 | 0.08615439477415214 |
| 3 | 1 | 1 | 0.9109325593219613 |
| 4 | 1 | 2 | 0.08906744067803873 |
| 5 | 2 | 1 | 0.7631503475567551 |
| 6 | 2 | 2 | 0.236849652443245 |
| 7 | 3 | 1 | 0.7592599453956861 |
| 8 | 3 | 2 | 0.2407400546043139 |
| 9 | 4 | 1 | 0.16378038436447268 |
| 10 | 4 | 2 | 0.8362196156355274 |
| 11 | 5 | 1 | 0.16110939586122852 |
| 12 | 5 | 2 | 0.8388906041387716 |
| 13 | 6 | 1 | 0.7149531325348973 |
| 14 | 6 | 2 | 0.28504686746510266 |

**Example 4: One class SVM**

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_SVM_PREDICT_DATA_TBL;
CREATE COLUMN TABLE PAL_SVM_PREDICT_DATA_TBL(
    ID INTEGER,
    ATTRIBUTE1 DOUBLE,
    ATTRIBUTE2 DOUBLE,
    ATTRIBUTE3 DOUBLE,
    ATTRIBUTE4 VARCHAR(50)
);
INSERT INTO PAL_SVM_PREDICT_DATA_TBL VALUES(0,1,10,100,'A');
INSERT INTO PAL_SVM_PREDICT_DATA_TBL VALUES(1,1.1,10.1,100,'A');
INSERT INTO PAL_SVM_PREDICT_DATA_TBL VALUES(2,1.2,10.2,100,'A');
INSERT INTO PAL_SVM_PREDICT_DATA_TBL VALUES(3,1.3,10.4,100,'A');
INSERT INTO PAL_SVM_PREDICT_DATA_TBL VALUES(4,1.2,10.3,100,'AB');
INSERT INTO PAL_SVM_PREDICT_DATA_TBL VALUES(5,4,40,400,'AB');
INSERT INTO PAL_SVM_PREDICT_DATA_TBL VALUES(6,4.1,40.1,400,'AB');
```

```
INSERT INTO PAL_SVM_PREDICT_DATA_TBL VALUES(7,4.2,40.2,400,'AB');
INSERT INTO PAL_SVM_PREDICT_DATA_TBL VALUES(8,4.3,40.4,400,'AB');
INSERT INTO PAL_SVM_PREDICT_DATA_TBL VALUES(9,4.2,40.3,400,'AB');
INSERT INTO PAL_SVM_PREDICT_DATA_TBL VALUES(10,9,90,900,'B');
INSERT INTO PAL_SVM_PREDICT_DATA_TBL VALUES(11,9.1,90.1,900,'A');
INSERT INTO PAL_SVM_PREDICT_DATA_TBL VALUES(12,9.2,90.2,900,'B');
INSERT INTO PAL_SVM_PREDICT_DATA_TBL VALUES(13,9.3,90.4,900,'A');
INSERT INTO PAL_SVM_PREDICT_DATA_TBL VALUES(14,9.2,90.3,900,'A');
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES('THREAD_RATIO',NULL,0.1,NULL);
CALL
_SYS_AFL.PAL_SVM_PREDICT(PAL_SVM_PREDICT_DATA_TBL,PAL_SVM_MODEL_TBL_EX4,#PAL_PARA
METER_TBL,?);
```

**Expected Result**

RESULT:

| | ID | SCORE | PROBABILITY |
|---|---|---|---|
| 1 | 0 | -1 | ? |
| 2 | 1 | 1 | ? |
| 3 | 2 | 1 | ? |
| 4 | 3 | -1 | ? |
| 5 | 4 | -1 | ? |
| 6 | 5 | -1 | ? |
| 7 | 6 | -1 | ? |
| 8 | 7 | 1 | ? |
| 9 | 8 | -1 | ? |
| 10 | 9 | -1 | ? |
| 11 | 10 | -1 | ? |
| 12 | 11 | 1 | ? |
| 13 | 12 | -1 | ? |
| 14 | 13 | -1 | ? |
| 15 | 14 | 1 | ? |

> **Note**
>
> The wrapper procedure generation approach to call PAL functions is still supported. For more information, refer to the older versions (prior to SAP HANA Platform 2.0 SPS 02) of this document.

## Related Information

It is possible to perform scoring function using state enabled functions. For more information, see:
State Enabled Real-Time Scoring Functions [page 41]

## 5.2.13.1 Model Compression

Model compression technique aims to reduce the size of model with minimum loss of the accuracy.

### Relevant Parameters

- COMPRESSION
  Specifies if the model is stored in compressed format.
- MAX_BITS
  The maximum number of bits to quantize values of support vectors. Equivalent to use $2^{MAX_BITS}$ bins to quantize values of support vectors. Reduce the size of bins may affect the precision of support vectors and the accuracy in prediction.
- MAX_QUANTIZATION_ITER
  The maximum iteration number of the quantization process. Too small MAX_QUANTIZATION_ITER may result in not converge of quantization and affects the accuracy further.

### Example

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_SVM_DATA_TBL;
CREATE COLUMN TABLE PAL_SVM_DATA_TBL (
    ID INTEGER,
    ATTRIBUTE1 DOUBLE,
    ATTRIBUTE2 DOUBLE,
    ATTRIBUTE3 DOUBLE,
    ATTRIBUTE4 VARCHAR(10),
    LABEL INTEGER
);
INSERT INTO PAL_SVM_DATA_TBL VALUES (0,1,10,100,'A',1);
INSERT INTO PAL_SVM_DATA_TBL VALUES (1,1.1,10.1,100,'A',1);
INSERT INTO PAL_SVM_DATA_TBL VALUES (2,1.2,10.2,100,'A',1);
INSERT INTO PAL_SVM_DATA_TBL VALUES (3,1.3,10.4,100,'A',1);
INSERT INTO PAL_SVM_DATA_TBL VALUES (4,1.2,10.3,100,'AB',1);
INSERT INTO PAL_SVM_DATA_TBL VALUES (5,4,40,400,'AB',2);
INSERT INTO PAL_SVM_DATA_TBL VALUES (6,4.1,40.1,400,'AB',2);
INSERT INTO PAL_SVM_DATA_TBL VALUES (7,4.2,40.2,400,'AB',2);
INSERT INTO PAL_SVM_DATA_TBL VALUES (8,4.3,40.4,400,'AB',2);
INSERT INTO PAL_SVM_DATA_TBL VALUES (9,4.2,40.3,400,'AB',2);
INSERT INTO PAL_SVM_DATA_TBL VALUES (10,9,90,900,'B',3);
INSERT INTO PAL_SVM_DATA_TBL VALUES (11,9.1,90.1,900,'A',3);
INSERT INTO PAL_SVM_DATA_TBL VALUES (12,9.2,90.2,900,'B',3);
INSERT INTO PAL_SVM_DATA_TBL VALUES (13,9.3,90.4,900,'A',3);
INSERT INTO PAL_SVM_DATA_TBL VALUES (14,9.2,90.3,900,'A',3);
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL (
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR (1000)
);
INSERT INTO PAL_PARAMETER_TBL VALUES ('KERNEL_TYPE',2,NULL,NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('TYPE',1,NULL,NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('RBF_GAMMA',NULL,0.005,NULL);
```

```
INSERT INTO PAL_PARAMETER_TBL VALUES ('HAS_ID',1,NULL,NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('HANDLE_MISSING', 0, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('PROBABILITY', 1, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('COMPRESSION', 1, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('MAX_BITS', 10, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('MAX_QUANTIZATION_ITER', 500, NULL, NULL);
DROP TABLE PAL_SVM_MODEL_TBL_EX1;
CREATE COLUMN TABLE PAL_SVM_MODEL_TBL_EX1 (ROW_INDEX INTEGER, MODEL_CONTENT
NVARCHAR(5000));
DROP TABLE PAL_SVM_STAT_TBL;
CREATE COLUMN TABLE PAL_SVM_STAT_TBL (STAT_NAME NVARCHAR(256), STAT_VALUE NCLOB);
DO BEGIN
    lt_data = SELECT * FROM PAL_SVM_DATA_TBL;
    lt_param = SELECT * FROM PAL_PARAMETER_TBL;
    CALL _SYS_AFL.PAL_SVM(:lt_data,:lt_param,lt_model,lt_stat,lt_opt);
    INSERT INTO PAL_SVM_MODEL_TBL_EX1 SELECT * FROM :lt_model;
    INSERT INTO PAL_SVM_STAT_TBL SELECT * FROM :lt_stat;
END;
SELECT * FROM PAL_SVM_MODEL_TBL_EX1;
```

**Expected Result**

| | ROW_INDEX | MODEL_CONTENT |
|---|---|---|
| 1 | 0 | {"means":[4.826666666666667,46.86666666666667,466.6666666666667],"deviations":[3.4172810 |

## 5.2.14 Incremental Classification on SAP HANA Streaming Analytics

For information on incremental classification on SAP HANA Streaming Analytics, see the section on Hoeffding tree training and scoring example in the *SAP HANA Streaming Analytics: Developer Guide*.

### Related Information

SAP HANA Streaming Analytics: Developer Guide

## 5.3 Regression Algorithms

This section describes the regression algorithms that are provided by the Predictive Analysis Library.

## 5.3.1 Bi-Variate Geometric Regression

Geometric regression is an approach used to model the relationship between a scalar variable y and a variable denoted X. In geometric regression, data is modeled using geometric functions, and unknown model parameters are estimated from the data. Such models are called geometric models.

In PAL, the implementation of geometric regression is to transform to linear regression and solve it:

$y = \beta_0 \times x^{\beta_1}$

Where $\beta_0$ and $\beta_1$ are parameters that need to be calculated.

The steps are:

1. Put natural logarithmic operation on both sides: $\ln(y) = \ln(\beta_0 \times x^{\beta_1})$
2. Transform it into: $\ln(y) = \ln(\beta_0) + \beta_1 \times \ln(x)$
3. Let $y' = \ln(y)$, $x' = \ln(x)$, $\beta_0' = \ln(\beta_0)$
   $y' = \beta_0' + \beta_1 \times x'$

Thus, `y'` and `x'` is a linear relationship and can be solved with the linear regression method.

The implementation also supports calculating the F value and R^2 to determine statistical significance.

### Prerequisites

- No missing or null data in the inputs.
- The data is numeric, not categorical.
- Given the structure as Y and X1...Xn, there are at least n+1 records available for analysis.

### _SYS_AFL.PAL_GEOMETRIC_REGRESSION

This is a geometric regression function.

**Overview of all Tables**

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<COEFFICIENT table>` | OUT |
| 4 | `<FITTED table>` | OUT |
| 5 | `<STATISTICS table>` | OUT |

| Position | Table Name | Parameter Type |
|---|---|---|
| 6 | `<PMML table>` | OUT |

### Input Table

| Table | Column | Reference for Output Table | Data Type | Description |
|---|---|---|---|---|
| DATA | 1st column | ID | INTEGER, VARCHAR, or NVARCHAR | ID.<br>If this column is absent, the `HAS_ID` parameter must be set to 0. |
| | 2nd column | | INTEGER, DOUBLE, or DECIMAL(p, s) | Variable y |
| | 3rd column | | INTEGER, DOUBLE, or DECIMAL(p, s) | Variable x |

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| ALG | INTEGER | 1 | Specifies decomposition method:<br>• 0: LU Decomposition (Fast, but requires matrix invertible, can not ensure numerical stable)<br>• 1: QR Decomposition (Fast, does not require matrix invertible, numerical stable)<br>• 2: Singular Value Decomposition (Slow, but does not require matrix invertible, numerical stable)<br>• 5: Cholesky Decomposition (Very Fast, but requires matrix invertible, numerical stable) |
| ADJUSTED_R2 | INTEGER | 0 | • 0: Does not output adjusted R square<br>• 1: Outputs adjusted R square |
| PMML_EXPORT | INTEGER | 0 | • 0: Does not export geometric regression model in PMML.<br>• 1: Exports geometric regression model in PMML in single row.<br>• 2: Exports geometric regression model in several rows, and the minimum length of each row is 5000 characters. |
| SELECTED_FEATURES | VARCHAR | No default value | A string to specify the features that will be processed. The pattern is "$X_1,...,X_n$", where $X_i$ is the corresponding column name in the data table. If this parameter is not specified, all the features will be processed, and the third column data will be processed as independent variables. |

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| DEPENDENT_VARIABLE | VARCHAR | No default value | (Optional) Column name in the data table used as dependent variable. If this parameter is not specified, the second column data will be processed as dependent variables. |
| HAS_ID | INTEGER | 1 | Specifies whether the input data table has an ID column. If it exists, the ID column must be the first column.<br>• 0: No ID column<br>• 1: Has an ID column |

### Output Tables

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| COEFFICIENT | 1st column | NVARCHAR(1000) | VARIABLE_NAME | Name of variable |
| | 2nd column | DOUBLE | COEFFICIENT_VALUE | Values of the coefficients |
| FITTED | 1st column | ID (in input table) data type | ID (in input table) column name | ID |
| | 2nd column | DOUBLE | VALUE | Value $Y_i$ |
| STATISTICS | 1st column | NVARCHAR(100) | STAT_NAME | Name |
| | 2nd column | DOUBLE | STAT_VALUE | Value |
| PMML | 1st column | INTEGER | ROW_INDEX | ID |
| | 2nd column | NVARCHAR(5000) | MODEL_CONTENT | Geometric regression model in PMML format |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA DM_PAL;
```

```
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE
    #PAL_PARAMETER_TBL
    ("PARAM_NAME" VARCHAR(256), "INT_VALUE" INTEGER, "DOUBLE_VALUE" DOUBLE,
"STRING_VALUE" VARCHAR(1000));
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PMML_EXPORT',2,NULL,NULL);
DROP TABLE PAL_GR_DATA_TBL;
CREATE COLUMN TABLE PAL_GR_DATA_TBL ( "ID" INT,"Y" DOUBLE,"X1" DOUBLE);
INSERT INTO PAL_GR_DATA_TBL VALUES (0,1.1,1);
INSERT INTO PAL_GR_DATA_TBL VALUES (1,4.2,2);
INSERT INTO PAL_GR_DATA_TBL VALUES (2,8.9,3);
INSERT INTO PAL_GR_DATA_TBL VALUES (3,16.3,4);
INSERT INTO PAL_GR_DATA_TBL VALUES (4,24,5);
INSERT INTO PAL_GR_DATA_TBL VALUES (5,36,6);
INSERT INTO PAL_GR_DATA_TBL VALUES (6,48,7);
INSERT INTO PAL_GR_DATA_TBL VALUES (7,64,8);
INSERT INTO PAL_GR_DATA_TBL VALUES (8,80,9);
INSERT INTO PAL_GR_DATA_TBL VALUES (9,101,10);
CALL _SYS_AFL.PAL_GEOMETRIC_REGRESSION(PAL_GR_DATA_TBL,
"#PAL_PARAMETER_TBL", ?, ?, ?, ?);
```

**Expected Result**

COEFFICIENT:

| | VARIABLE_NAME | COEFFIFICENT_VALUE |
|---|---|---|
| 1 | __PAL_INTERCEPT__ | 1.0722194976977595 |
| 2 | X1 | 1.9596331198574095 |

FITTED:

| | ID | VALUE |
|---|---|---|
| 1 | 0 | 1.0722194976977595 |
| 2 | 1 | 4.1705376859163685 |
| 3 | 2 | 9.231373297672475 |
| 4 | 3 | 16.221850681689016 |
| 5 | 4 | 25.119354247561184 |
| 6 | 5 | 35.90663135054978 |
| 7 | 6 | 48.56974340546408 |
| 8 | 7 | 63.09700555581822 |
| 9 | 8 | 79.47836533840201 |
| 10 | 9 | 97.70500700675362 |

STATISTICS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | R2 | 0.9996968130780067 |
| 2 | F | 26,378.36240443944 |
| 3 | MSE | 0.0007040126288691115 |
| 4 | RMSE | 0.026533236305982567 |
| 5 | MAE | 0.018788222738093427 |
| 6 | MAPE | 3.2194294449046965 |
| 7 | EXPLAINED_VARIANCE | 0.9996968130780067 |
| 8 | MAX_ERROR | 0.04558480421626232 |

PMML:

| | ROW_INDEX | MODEL_CONTENT |
|---|---|---|
| 1 | 1 | <PMML version="4.0" xmlns="http://www.dmg.org/PMML-4_0" xmlns:xsi="http://www.... |

## _SYS_AFL.PAL_GEOMETRIC_REGRESSION_PREDICT

This function performs prediction with the geometric regression result.

### Overview of all Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<COEFFICIENT table>` | IN |
| 3 | `<PARAMETER table>` | IN |
| 4 | `<FITTED table>` | OUT |

### Input Tables

| Table | Column | Reference for Output Table | Data Type | Description |
|---|---|---|---|---|
| DATA | 1st column | ID | INTEGER, VARCHAR, or NVARCHAR | ID |
| | Feature column | | INTEGER, DOUBLE, or DECIMAL(p, s) | Variable X |
| COEFFICIENT | 1st column | | INTEGER, VARCHAR, or NVARCHAR | Variable name or ROW_INDEX for PMML model |
| | 2nd column | | INTEGER, DOUBLE, VARCHAR, NVARCHAR, or CLOB | Values of the coefficients or PMML model content |

The Coefficient Input Table accepts both model formats from the output tables Coefficient and PMML generated by the previous training procedure. To differentiate the two model formats, the predict procedure uses a combination of two column types to identify the model format. Once the model format is determined, the procedure parses the content. Ensure that the column types in the input table correspond to those in either the Coefficient or PMML Model Table, depending on the intended model format.

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| THREAD_RATIO | DOUBLE | 1.0 | Specifies the ratio of total number of threads that can be used by this function. The value range is from 0 to 1, where 0 means only using 1 thread, and 1 means using at most all the currently available threads. Values outside the range will be ignored and this function heuristically determines the number of threads to use. |

### Output Table

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| FITTED | 1st column | ID (in input table) data type | ID (in input table) column name | ID |
| | 2nd column | DOUBLE | VALUE | Value $Y_i$ |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA DM_PAL;
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE
    #PAL_PARAMETER_TBL
    ("PARAM_NAME" VARCHAR(256), "INT_VALUE" INTEGER, "DOUBLE_VALUE" DOUBLE,
"STRING_VALUE" VARCHAR(1000));
INSERT INTO #PAL_PARAMETER_TBL VALUES ('THREAD_RATIO',NULL,0.2, NULL);
DROP TABLE PAL_FGR_DATA_TBL;
CREATE COLUMN TABLE PAL_FGR_DATA_TBL ( "ID" INT,"X1" DOUBLE);
INSERT INTO PAL_FGR_DATA_TBL VALUES (0,1);
INSERT INTO PAL_FGR_DATA_TBL VALUES (1,2);
INSERT INTO PAL_FGR_DATA_TBL VALUES (2,3);
INSERT INTO PAL_FGR_DATA_TBL VALUES (3,4);
INSERT INTO PAL_FGR_DATA_TBL VALUES (4,5);
```

```
DROP TABLE PAL_FGR_COEFFICIENT_TBL;
CREATE COLUMN TABLE PAL_FGR_COEFFICIENT_TBL ("VARIABLE_NAME" NVARCHAR(128),
"COEFFICIENT_VALUE" DOUBLE);
INSERT INTO PAL_FGR_COEFFICIENT_TBL VALUES ('__PAL_INTERCEPT__', 1);
INSERT INTO PAL_FGR_COEFFICIENT_TBL VALUES ('X1', 1.99);
CALL _SYS_AFL.PAL_GEOMETRIC_REGRESSION_PREDICT(PAL_FGR_DATA_TBL,
PAL_FGR_COEFFICIENT_TBL, "#PAL_PARAMETER_TBL", ?);
```

**Expected Result**

FITTED:

| | ID | Fitted |
|---|---|---|
| 1 | 0 | 1 |
| 2 | 1 | 3.9723699817481437 |
| 3 | 2 | 8.901666037549536 |
| 4 | 3 | 15.779723271893747 |
| 5 | 4 | 24.60086108408644 |

> **Note**
>
> The wrapper procedure generation approach to call PAL functions is still supported. For more information, refer to the older versions (prior to SAP HANA Platform 2.0 SPS 02) of this document.

## 5.3.2 Bi-Variate Natural Logarithmic Regression

Bi-variate natural logarithmic regression is an approach to modeling the relationship between a scalar variable y and one variable denoted X. In natural logarithmic regression, data is modeled using natural logarithmic functions, and unknown model parameters are estimated from the data. Such models are called natural logarithmic models.

In PAL, the implementation of natural logarithmic regression is to transform to linear regression and solve it:

$y = \beta 1 \ln(x) + \beta 0$

Where $\beta 0$ and $\beta 1$ are parameters that need to be calculated.

Let $x' = \ln(x)$

Then $y = \beta 0 + \beta 1 \times x'$

Thus, y and $x'$ is a linear relationship and can be solved with the linear regression method.

The implementation also supports calculating the F value and R^2 to determine statistical significance.

### Prerequisites

- No missing or null data in the inputs.
- The data is numeric, not categorical.
- Given the structure as Y and X, there are more than 2 records available for analysis.

# _SYS_AFL.PAL_LOGARITHMIC_REGRESSION

This is a logarithmic regression function.

### Overview of all Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<COEFFICIENT table>` | OUT |
| 4 | `<FITTED table>` | OUT |
| 5 | `<STATISTICS table>` | OUT |
| 6 | `<PMML table>` | OUT |

### Input Table

| Table | Column | Reference for Output Table | Data Type | Description |
|---|---|---|---|---|
| DATA | 1st column | ID | INTEGER, VARCHAR, or NVARCHAR | ID.<br>If this column is absent, the `HAS_ID` parameter must be set to 0. |
| | 2nd column | | INTEGER, DOUBLE, or DECIMAL(p, s) | Variable y |
| | 3rd column | | INTEGER, DOUBLE, or DECIMAL(p, s) | Variable X |

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| ALG | INTEGER | 1 | Specifies decomposition method:<br>• 0: LU Decomposition (Fast, but requires matrix invertible, can not ensure numerical stable)<br>• 1: QR Decomposition (Fast, does not require matrix invertible, numerical stable)<br>• 2: Singular Value Decomposition (Slow, but does not require matrix invertible, numerical stable)<br>• 5: Cholesky Decomposition (Very Fast, but requires matrix invertible, numerical stable) |
| ADJUSTED_R2 | INTEGER | 0 | • 0: Does not output adjusted R square<br>• 1: Outputs adjusted R square |
| PMML_EXPORT | INTEGER | 0 | • 0: Does not export logarithmic regression model in PMML.<br>• 1: Exports logarithmic regression model in PMML in single row.<br>• 2: Exports logarithmic regression model in PMML in several rows, and the minimum length of each row is 5000 characters. |
| SELECTED_FEATURES | VARCHAR | No default value | A string to specify the features that will be processed. The pattern is "$X_1,...,X_n$", where $X_i$ is the corresponding column name in the data table. If this parameter is not specified, all the features will be processed, and the third column data will be processed as independent variables. |

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| DEPENDENT_VARIABLE | VARCHAR | No default value | Column name in the data table used as dependent variable. If this parameter is not specified, the second column data will be processed as dependent variables. |
| HAS_ID | INTEGER | 1 | Specifies whether the input data table has an ID column. If it exists, the ID column must be the first column.<br>• 0: No ID column<br>• 1: Has an ID column |

### Output Tables

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| COEFFICIENT | 1st column | NVARCHAR(1000) | VARIABLE_NAME | Name of variable |
| | 2nd column | DOUBLE | COEFFICIENT_VALUE | Values of the coefficients |
| FITTED | 1st column | ID (in input table) data type | ID (in input table) column name | ID |
| | 2nd column | DOUBLE | VALUE | Value $Y_i$ |
| STATISTICS | 1st column | NVARCHAR(100) | STAT_NAME | Name |
| | 2nd column | DOUBLE | STAT_VALUE | Value |
| PMML | 1st column | INTEGER | ROW_INDEX | ID |
| | 2nd column | NVARCHAR(5000) | MODEL_CONTENT | Logarithmic regression model in PMML format |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA DM_PAL;
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE
```

```
    #PAL_PARAMETER_TBL
    ("PARAM_NAME" VARCHAR(256), "INT_VALUE" INTEGER, "DOUBLE_VALUE" DOUBLE,
"STRING_VALUE" VARCHAR(1000));
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PMML_EXPORT',2, NULL,NULL);
DROP TABLE PAL_NLR_DATA_TBL;
CREATE COLUMN TABLE PAL_NLR_DATA_TBL ( "ID" INT,"Y" DOUBLE,"X1" DOUBLE);
INSERT INTO PAL_NLR_DATA_TBL VALUES (0,10,1);
INSERT INTO PAL_NLR_DATA_TBL VALUES (1,80,2);
INSERT INTO PAL_NLR_DATA_TBL VALUES (2,130,3);
INSERT INTO PAL_NLR_DATA_TBL VALUES (3,160,4);
INSERT INTO PAL_NLR_DATA_TBL VALUES (4,180,5);
INSERT INTO PAL_NLR_DATA_TBL VALUES (5,190,6);
INSERT INTO PAL_NLR_DATA_TBL VALUES (6,192,7);
CALL _SYS_AFL.PAL_LOGARITHMIC_REGRESSION(PAL_NLR_DATA_TBL,
"#PAL_PARAMETER_TBL", ?, ?, ?, ?);
```

**Expected Result**

COEFFICIENT:

| | VARIABLE_NAME | COEFFICIENT_VALUE |
|---|---|---|
| 1 | __PAL_INTERCEPT__ | 14.861602985473446 |
| 2 | X1 | 98.2935974594811 |

FITTED:

| | ID | VALUE |
|---|---|---|
| 1 | 0 | 14.861602985473446 |
| 2 | 1 | 82.99353293160698 |
| 3 | 2 | 122.84815705185588 |
| 4 | 3 | 151.1254628777405 |
| 5 | 4 | 173.0590452862985 |
| 6 | 5 | 190.9800869979894 |
| 7 | 6 | 206.13211186903527 |

STATISTICS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | R2 | 0.985094537389156 |
| 2 | F | 330.4474886517373 |
| 3 | MSE | 82.27133968609414 |
| 4 | RMSE | 9.070354992286362 |
| 5 | MAE | 6.5620956526014576 |
| 6 | MAPE | 10.734049359859908 |
| 7 | EXPLAINED_VARIANCE | 0.985094537389156 |
| 8 | MAX_ERROR | 14.13211186903527 |

PMML:

| | ROW_INDEX | MODEL_CONTENT |
|---|---|---|
| 1 | 1 | <PMML version="4.0" xmlns="http://www.dmg.org/PMML-4_0" xmlns:xsi="http://www.... |

## _SYS_AFL.PAL_LOGARITHMIC_REGRESSION_PREDICT

This function performs prediction with the natural logarithmic regression result.

### Overview of all Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<COEFFICIENT table>` | IN |
| 3 | `<PARAMETER table>` | IN |
| 4 | `<FITTED table>` | OUT |

### Input Tables

| Table | Column | Reference for Output Table | Data Type | Description |
|---|---|---|---|---|
| DATA | 1st column | ID | INTEGER, VARCHAR, or NVARCHAR | ID |
| | Feature column | | INTEGER, DOUBLE, or DECIMAL(p, s) | Variable X |
| COEFFICIENT | 1st column | | INTEGER, VARCHAR, or NVARCHAR | Variable name or ROW_INDEX for PMML |
| | 2nd column | | INTEGER, DOUBLE, VARCHAR, NVARCHAR, or CLOB | Values of the coefficients or PMML model content |

The Coefficient Input Table accepts both model formats from the output tables Coefficient and PMML generated by the previous training procedure. To differentiate the two model formats, the predict procedure uses a combination of two column types to identify the model format. Once the model format is determined, the procedure parses the content. Ensure that the column types in the input table correspond to those in either the Coefficient or PMML Model Table, depending on the intended model format.

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| THREAD_RATIO | DOUBLE | 1.0 | Specifies the ratio of total number of threads that can be used by this function. The value range is from 0 to 1, where 0 means only using 1 thread, and 1 means using at most all the currently available threads. Values outside the range will be ignored and this function heuristically determines the number of threads to use. |

### Output Table

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| FITTED | 1st column | ID (in input table) data type | ID (in input table) column name | ID |
| | 2nd column | DOUBLE | VALUE | Value $Y_i$ |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA DM_PAL;
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE
    #PAL_PARAMETER_TBL
    ("PARAM_NAME" VARCHAR(256), "INT_VALUE" INTEGER, "DOUBLE_VALUE" DOUBLE,
"STRING_VALUE" VARCHAR(1000));
INSERT INTO #PAL_PARAMETER_TBL VALUES ('THREAD_RATIO',NULL,0.2,NULL);
DROP TABLE PAL_FNLR_DATA_TBL;
CREATE COLUMN TABLE PAL_FNLR_DATA_TBL ( "ID" INT,"X1" DOUBLE);
INSERT INTO PAL_FNLR_DATA_TBL VALUES (0,1);
INSERT INTO PAL_FNLR_DATA_TBL VALUES (1,2);
INSERT INTO PAL_FNLR_DATA_TBL VALUES (2,3);
INSERT INTO PAL_FNLR_DATA_TBL VALUES (3,4);
INSERT INTO PAL_FNLR_DATA_TBL VALUES (4,5);
INSERT INTO PAL_FNLR_DATA_TBL VALUES (5,6);
INSERT INTO PAL_FNLR_DATA_TBL VALUES (6,7);
DROP TABLE PAL_FNLR_COEFFICIENT_TBL;
CREATE COLUMN TABLE PAL_FNLR_COEFFICIENT_TBL ("VARIABLE_NAME" NVARCHAR(128),
"COEFFICIENT_VALUE" DOUBLE);
INSERT INTO PAL_FNLR_COEFFICIENT_TBL VALUES ('__PAL_INTERCEPT__', 14.86160299);
INSERT INTO PAL_FNLR_COEFFICIENT_TBL VALUES ('X1', 98.29359746);
```

```
CALL _SYS_AFL.PAL_LOGARITHMIC_REGRESSION_PREDICT(PAL_FNLR_DATA_TBL,
PAL_FNLR_COEFFICIENT_TBL, "#PAL_PARAMETER_TBL", ?);
```

**Expected Result**

FITTED:

| | ID | VALUE |
|---|---|---|
| 1 | 0 | 14.86160299 |
| 2 | 1 | 82.9935329364932 |
| 3 | 2 | 122.8481570569... |
| 4 | 3 | 151.1254628829... |
| 5 | 4 | 173.0590452916... |
| 6 | 5 | 190.9800870034... |
| 7 | 6 | 206.1321118745... |

> **Note**
>
> The wrapper procedure generation approach to call PAL functions is still supported. For more information, refer to the older versions (prior to SAP HANA Platform 2.0 SPS 02) of this document.

## 5.3.3 Cox Proportional Hazard Model

Cox proportional hazard model (CoxPHM) is a special generalized linear model. It is a well-known realization-of-survival model that demonstrates failure or death at a certain time.

CoxPHM has the following generalization:

$$h(t,x)=h_0(t,\alpha)\exp(x^T\beta)$$

where $h_0$ is called baseline hazard function, and $\alpha$ is a parameter influencing the baseline hazard function. In contrast to standard generalized linear models, CoxPHM does not have an intercept, as it is eliminated by division.

Parameter estimation is attained by maximum likelihood estimation, but in this case partial likelihood function is used instead. As for the tied events, whose failure or death time are identical, approximates such as Breslow's method, Efron's method, etc. are employed.

### Prerequisites

- An ID column must be provided for both estimation and prediction, and no null value is allowed in this column.
- Coefficients are assigned to columns' names, so all covariates' columns used in estimation must be defined in prediction.
- For censored data, only right-censored data is allowed.

# _SYS_AFL.PAL_COXPH

This function estimates the Cox proportional hazard model by maximum likelihood estimation.

### Overview of all Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<STATISTICS table>` | OUT |
| 4 | `<COEFFICIENT table>` | OUT |
| 5 | `<COVARIANCE_VARIANCE table>` | OUT |
| 6 | `<HAZARD table>` | OUT |
| 7 | `<FITTED table>` | OUT |

### Input Table

| Table | Column | Reference for Output Table | Data Type | Description |
|---|---|---|---|---|
| DATA | 1st column | DID | INTEGER, DOUBLE, DECIMAL(p, s), VARCHAR, or NVARCHAR | Data ID |
| | 2nd column | TID | INTEGER, DOUBLE, or DECIMAL(p, s) | The time before a failure/death event occurs or data is right censored. |
| | 3rd column (optional) | | INTEGER | A status column that indicates if the individual is an event or right-censored data:<br>• 0: right-censored data<br>• 1: failure/death<br>This column is optional. If the column is absent, all values are set to 1. |

| Table | Column | Reference for Output Table | Data Type | Description |
|---|---|---|---|---|
| | Other columns | | INTEGER, DOUBLE, or DECIMAL(p, s) | Covariates |

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| TIE_METHOD | VARCHAR or NVARCHAR | efron | The method to deal with tied events. The values are:<br>• breslow<br>• efron |
| STATUS_COL | INTEGER | 1 | If a status column is defined for right-censored data:<br>• 0: No status column. All response times are failure/death.<br>• 1: The 3rd column of the data input table is a status column, of which 0 indicates right-censored data and 1 indicates failure/death. |
| MAX_ITERATION | INTEGER | 100 | Maximum number of iterations for numeric optimization. |
| CONVERGENCE_CRITERION | DOUBLE | 1e-8 | Convergence criterion of coefficients for numeric optimization. |
| SIGNIFICANCE_LEVEL | DOUBLE | 0.05 | Significance level for the confidence interval of estimated coefficients. |

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| CALCULATE_HAZARD | INTEGER | 1 | Controls whether to calculate hazard function as well as survival function.<br>• 0: Does not calculate hazard function.<br>• 1: Calculates hazard function. |
| OUTPUT_FITTED | INTEGER | 0 | Controls whether to output the fitted response:<br>• 0: Does not output the fitted response.<br>• 1: Outputs the fitted response. |

### Output Tables

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| STATISTICS | 1st column | NVARCHAR(1000) | STAT_NAME | Name |
| | 2nd column | NVARCHAR(1000) | STAT_VALUE | Value |
| COEFFICIENT | 1st column | NVARCHAR(256) | VARIABLE_NAME | Variable name |
| | 2nd column | DOUBLE | MEAN | Means of covariates |
| | 3rd column | DOUBLE | COEFFICIENT | Coefficient beta |
| | 4th column | DOUBLE | SE | Corresponding standard error |
| | 5th column | DOUBLE | SCORE | Score (z-value) |
| | 6th column | DOUBLE | PROBABILITY | Probability Pr(>\|z\|) |
| | 7th column | DOUBLE | CI_LOWER | Lower bound of the confidence interval |
| | 8th column | DOUBLE | CI_UPPER | Upper bound of the confidence interval |
| COVARIANCE_VARIANCE | 1st column | NVARCHAR(256) | VARIABLE_I | i-th variable name |
| | 2nd column | NVARCHAR(256) | VARIABLE_J | j-th variable name |

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| | 3rd column | DOUBLE | COVARIANCE | Element of variance-covariance matrix.<br>Low-triangular as it is a symmetric matrix. |
| HAZARD | 1st column | TID (in input table) data type | TID (in input table) column name | Time to event |
| | 2nd column | DOUBLE | HAZARD | Cumulative baseline hazard function |
| | 3rd column | DOUBLE | SURVIVAL | Survival function |
| FITTED | 1st column | DID (in input table) data type | DID (in input table) column name | ID, in consistent with that in the data input table |
| | 2nd column | DOUBLE | PREDICTOR | Fitted linear predictor $x\beta$ |
| | 3rd column | DOUBLE | RESPONSE | Fitted response in risk space `exp(xβ)` |

> **Note**
>
> In the Statistics table, there are 3 properties who have two values, i.e. rsquare, log-likelihood, and aic. They have the following formats:
>
> - rsquare: the estimated R-square; the max possible R-square.
> - log-likelihood: the log likelihood estimated; the log likelihood with initial coefficient β.
> - aic: the AIC estimated; the AIC with initial coefficient β.

**Example**

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_COX_DATA_TBL;
CREATE COLUMN TABLE PAL_COX_DATA_TBL(
    "ID" INTEGER,
    "TIME" DOUBLE,
    "STATUS" INTEGER, -- if a failure/death, or right-censored, optional
    "X1" DOUBLE,
    "X2" DOUBLE
);
INSERT INTO PAL_COX_DATA_TBL VALUES(1, 4, 1, 0, 0);
INSERT INTO PAL_COX_DATA_TBL VALUES(2, 3, 1, 2, 0);
INSERT INTO PAL_COX_DATA_TBL VALUES(3, 1, 1, 1, 0);
```

```
INSERT INTO PAL_COX_DATA_TBL VALUES(4, 1, 0, 1, 0);
INSERT INTO PAL_COX_DATA_TBL VALUES(5, 2, 1, 1, 1);
INSERT INTO PAL_COX_DATA_TBL VALUES(6, 2, 1, 0, 1);
INSERT INTO PAL_COX_DATA_TBL VALUES(7, 3, 0, 0, 1);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR(100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (100)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('TIE_METHOD', null, null, 'efron');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('CALCULATE_HAZARD', 1, null, null);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('OUTPUT_FITTED', 1, null, null);
DROP TABLE PAL_COX_STATS_TBL;
CREATE COLUMN TABLE PAL_COX_STATS_TBL (
    "STAT_NAME" NVARCHAR(1000),
    "STAT_VALUE" NVARCHAR(1000)
);
DROP TABLE PAL_COX_COEFF_TBL;
CREATE COLUMN TABLE PAL_COX_COEFF_TBL (
    "VARIABLE_NAME" NVARCHAR(256),
    "COEFFICIENT" DOUBLE,
    "MEAN" DOUBLE,
    "SE" DOUBLE,
    "SCORE" DOUBLE,
    "PROBABILITY" DOUBLE,
    "CI_LOWER"    DOUBLE,
    "CI_UPPER" DOUBLE
);
DROP TABLE PAL_COX_VCOV_TBL;
CREATE COLUMN TABLE PAL_COX_VCOV_TBL (
    "VARIABLE_I" NVARCHAR(256),
    "VARIABLE_J" NVARCHAR(256),
    "COVARIANCE" DOUBLE
);
CALL _SYS_AFL.PAL_COXPH(PAL_COX_DATA_TBL, "#PAL_PARAMETER_TBL",
PAL_COX_STATS_TBL, PAL_COX_COEFF_TBL, PAL_COX_VCOV_TBL, ?, ?) WITH OVERVIEW;
```

### Expected Result

STATISTICS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | method | efron |
| 2 | n | 7 |
| 3 | nevent | 5 |
| 4 | ntime | 4 |
| 5 | p | 2 |
| 6 | iter | 4 |
| 7 | converged | 1 |
| 8 | concordance | 0.571428 |
| 9 | rsquare | 0.149665;0.821967 |
| 10 | log-likelihood | -5.47281;-6.04025 |
| 11 | aic | 14.9456;16.0805 |
| 12 | likelihood-ratio-test;p-value | 1.13488;0.566974 |
| 13 | wald-test;p-value | 0.959349;0.618984 |
| 14 | score-test;p-value | 1.04624;0.592665 |

COEFFICIENT:

| | VARIABLE_NAME | MEAN | COEFFICIENT | SE | SCORE | PROBABILITY | CI_LOWER | CI_UPPER |
|---|---|---|---|---|---|---|---|---|
| 1 | X1 | 0.7142857142857143 | 0.7811819306367978 | 0.797568915454227 | 0.9794538321392621 | 0.3273557896284399 | -0.7820244188421583 | 2.344388280115754 |
| 2 | X2 | 0.42857142857142855 | 0.9337832209779552 | 1.4081100181779653 | 0.6631464934723148 | 0.5072367361038571 | -1.8260617009208975 | 3.6936281428768076 |

COVARIANCE_VARIANCE:

| | VARIABLE_I | VARIABLE_J | COVARIANCE |
|---|---|---|---|
| 1 | X1 | X1 | 0.6361161748988319 |
| 2 | X1 | X2 | 0.7641169676775483 |
| 3 | X2 | X2 | 1.98277382329315 |

HAZARD:

| | TIME | HAZARD | SURVIVAL |
|---|---|---|---|
| 1 | 1 | 0.12543711558432336 | 0.8821112337324323 |
| 2 | 2 | 0.4950953393720672 | 0.6095127939590658 |
| 3 | 3 | 0.8086489405359019 | 0.4454595021198358 |
| 4 | 4 | 3.4155961945540407 | 0.0328568118086462 |

FITTED

| | ID | PREDICTOR | RESPONSE |
|---|---|---|---|
| 1 | 1 | -0.9581799023025506 | 0.3835904230354797 |
| 2 | 2 | 0.6041839589710447 | 1.8297584415195811 |
| 3 | 3 | -0.17699797166575... | 0.8377814838221458 |
| 4 | 4 | -0.17699797166575... | 0.8377814838221458 |
| 5 | 5 | 0.7567852493122023 | 2.1314132328539284 |
| 6 | 6 | -0.02439668132459... | 0.9758985122514662 |
| 7 | 7 | -0.02439668132459... | 0.9758985122514662 |

## _SYS_AFL.PAL_COXPH_PREDICT

This function makes prediction based on the Cox proportional hazard function.

### Overview of all Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<STATISTICS table>` | IN |
| 3 | `<COEFFICIENT table>` | IN |
| 4 | `<COVARIANCE_VARIANCE table>` | IN |

| Position | Table Name | Parameter Type |
|---|---|---|
| 5 | `<PARAMETER table>` | IN |
| 6 | `<RESULT table>` | OUT |

### Input Tables

| Table | Column | Reference for Output Table | Data Type | Description |
|---|---|---|---|---|
| DATA | 1st column | ID | INTEGER, DOUBLE, DECIMAL(p, s), VARCHAR, or NVARCHAR | ID.<br>This must be the first column. |
| | Other columns | | INTEGER, DOUBLE, or DECIMAL(p, s) | Covariates.<br>Should include all variables in estimate stage. |
| STATISTICS | 1st column | | VARCHAR or NVARCHAR | Name |
| | 2nd column | | VARCHAR or NVARCHAR | Value |
| COEFFICIENT | 1st column | | VARCHAR or NVARCHAR | Variable name |
| | 2nd column | | DOUBLE | Coefficient beta |
| | 3rd column | | DOUBLE | Mean of covariates |
| | 4th column | | DOUBLE | Corresponding standard error |
| | 5th column (optional) | | DOUBLE | Score (z-value) |
| | 6th column (optional) | | DOUBLE | Probability Pr(>\|z\|) |
| | 7th column (optional) | | DOUBLE | Lower bound of the confidence interval |
| | 8th column (optional) | | DOUBLE | Upper bound of the confidence interval |
| COVARIANCE_VARIANCE | 1st column | | VARCHAR or NVARCHAR | Variable name |

| Table | Column | Reference for Output Table | Data Type | Description |
|---|---|---|---|---|
| | 2nd column | | VARCHAR or NVARCHAR | Variable name |
| | 3rd column | | DOUBLE | Element of variance-covariance matrix. Low-triangular. |

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| TYPE | VARCHAR or NVARCHAR | risk | The prediction type:<br>• risk: Predicts in risk space<br>• lp: Predicts in linear predictor space |
| SIGNIFICANCE_LEVEL | DOUBLE | 0.05 | The significance level for the confidence interval and prediction interval |

### Output Table

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| RESULT | 1st column | ID (in input table) column data type | ID (in input table) column name | Data ID, in consistent with that in the data input table. |
| | 2nd column | DOUBLE | PREDICTION | Predicted value |
| | 3rd column | DOUBLE | SE | Standard error |
| | 4th column | DOUBLE | CI_LOWER | Lower of the confidence interval |

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| | 5th column | DOUBLE | CI_UPPER | Upper of the confidence interval |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_COX_DATA_TBL;
CREATE COLUMN TABLE PAL_COX_DATA_TBL (
    "ID" integer,
    "X1" DOUBLE,
    "X2" DOUBLE
);
INSERT INTO PAL_COX_DATA_TBL VALUES(1, 0, 0);
INSERT INTO PAL_COX_DATA_TBL VALUES(2, 2, 0);
INSERT INTO PAL_COX_DATA_TBL VALUES(3, 1, 0);
INSERT INTO PAL_COX_DATA_TBL VALUES(4, 1, 0);
INSERT INTO PAL_COX_DATA_TBL VALUES(5, 1, 1);
INSERT INTO PAL_COX_DATA_TBL VALUES(6, 0, 1);
INSERT INTO PAL_COX_DATA_TBL VALUES(7, 0, 1);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR(100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (100)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SIGNIFICANCE_LEVEL', null, 0.05, null);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('TYPE', null, null, 'risk');
CALL _SYS_AFL.PAL_COXPH_PREDICT (PAL_COX_DATA_TBL, PAL_COX_STATS_TBL,
PAL_COX_COEFF_TBL, PAL_COX_VCOV_TBL, "#PAL_PARAMETER_TBL",  ?);
```

### Expected Result

RESULT:

| | ID | PREDICTION | SE | CI_LOWER | CI_UPPER |
|---|---|---|---|---|---|
| 1 | 1 | 0.3835904230354797 | 0.41252626192492103 | 0.046607574260162196 | 3.15703219874087 |
| 2 | 2 | 1.8297584415195811 | 1.385833777663278 | 0.4146727188947353 | 8.07387561746511 |
| 3 | 3 | 0.8377814838221458 | 0.40089407670341654 | 0.3279555100683299 | 2.140161677689... |
| 4 | 4 | 0.8377814838221458 | 0.40089407670341654 | 0.3279555100683299 | 2.140161677689... |
| 5 | 5 | 2.1314132328539284 | 2.076210694360804 | 0.3158724940516992 | 14.38213980240... |
| 6 | 6 | 0.9758985122514662 | 0.5758764354246778 | 0.3069811712810188 | 3.102398437794... |
| 7 | 7 | 0.9758985122514662 | 0.5758764354246778 | 0.3069811712810188 | 3.102398437794... |

> **Note**
>
> The wrapper procedure generation approach to call PAL functions is still supported. For more information, refer to the older versions (prior to SAP HANA Platform 2.0 SPS 02) of this document.

## 5.3.4 Exponential Regression

Exponential regression is an approach to modeling the relationship between a scalar variable y and one or more variables denoted X. In exponential regression, data is modeled using exponential functions, and unknown model parameters are estimated from the data. Such models are called exponential models.

In PAL, the implementation of exponential regression is to transform to linear regression and solve it:

$$y = \beta 0 \times \exp(\beta 1 \times x1 + \beta 2 \times x2 + \ldots + \beta n \times xn)$$

Where $\beta 0 \ldots \beta n$ are parameters that need to be calculated.

The steps are:

1. Put natural logarithmic operation on both sides:
   $\ln(y) = \ln(\beta 0 \times \exp(\beta 1 \times x1 + \beta 2 \times x2 + \ldots + \beta n \times xn))$
2. Transform it into: $\ln(y) = \ln(\beta 0) + \beta 1 \times x1 + \beta 2 \times x2 + \ldots + \beta n \times xn$
3. Let $y' = \ln(y)$, $\beta 0' = \ln(\beta 0)$
   $y' = \beta 0' + \beta 1 \times x1 + \beta 2 \times x2 + \ldots + \beta n \times xn$

Thus, $y'$ and $x1 \ldots xn$ is a linear relationship and can be solved using the linear regression method.

The implementation also supports calculating the F value and R^2 to determine statistical significance.

### Prerequisites

- No missing or null data in the inputs.
- The data is numeric, not categorical.
- Given the structure as Y and X1...Xn, there are at least n+1 records available for analysis.

### _SYS_AFL.PAL_EXPONENTIAL_REGRESSION

This is an exponential regression function.

**Overview of All Tables**

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<COEFFICIENT table>` | OUT |
| 4 | `<FITTED table>` | OUT |
| 5 | `<STATISTICS table>` | OUT |

| Position | Table Name | Parameter Type |
|---|---|---|
| 6 | `<PMML table>` | OUT |

### Input Table

| Table | Column | Reference for Output Table | Data Type | Description |
|---|---|---|---|---|
| DATA | 1st column | ID | INTEGER, VARCHAR, or NVARCHAR | ID.<br>If this column is absent, the `HAS_ID` parameter must be set to 0. |
| | 2nd column | | INTEGER, DOUBLE, or DECIMAL(p, s) | Variable y |
| | FEATURE columns | | INTEGER, DOUBLE, or DECIMAL(p, s) | Variable X*n* |

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| ALG | INTEGER | 1 | Specifies decomposition method:<br>• 0: LU Decomposition (Fast, but requires matrix invertible, can not ensure numerical stable)<br>• 1: QR Decomposition (Fast, does not require matrix invertible, numerical stable)<br>• 2: Singular Value Decomposition (Slow, but does not require matrix invertible, numerical stable)<br>• 5: Cholesky Decomposition (Very Fast, but requires matrix invertible, numerical stable) |
| ADJUSTED_R2 | INTEGER | 0 | • 0: Does not output adjusted R square<br>• 1: Outputs adjusted R square |
| PMML_EXPORT | INTEGER | 0 | • 0: Does not export exponential regression model in PMML.<br>• 1: Exports exponential regression model in PMML in single row.<br>• 2: Exports exponential regression model in PMML in several rows, and the minimum length of each row is 5000 characters. |

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| SELECTED_FEATURES | VARCHAR | No default value | A string to specify the features that will be processed. The pattern is "$X_1,...,X_n$", where $X_i$ is the corresponding column name in the data table. If this parameter is not specified, all the features will be processed, and all columns except for the first and second column data will be processed as independent variables. |
| DEPENDENT_VARIABLE | VARCHAR | No default value | Column name in the data table used as dependent variable. If this parameter is not specified, the second column data will be processed as dependent variables. |
| HAS_ID | INTEGER | 1 | Specifies whether the input data table has an ID column. If it exists, the ID column must be the first column.<br>• 0: No ID column<br>• 1: Has an ID column |

## Output Tables

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| COEFFICIENT | 1st column | NVARCHAR(100) | VARIABLE_NAME | Name of variable |
| | 2nd column | DOUBLE | COEFFICIENT_VALUE | The values of the coefficients |
| FITTED | 1st column | ID (in input table) data type | ID (in input table) column name | ID |
| | 2nd column | DOUBLE | VALUE | Value $Y_i$ |
| STATISTICS | 1st column | NVARCHAR(100) | STAT_NAME | Name |
| | 2nd column | DOUBLE | STAT_VALUE | Value |

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| PMML | 1st column | INTEGER | ROW_INDEX | ID |
| | 2nd column | NVARCHAR(5000) | MODEL_CONTENT | Exponential regression model in PMML format |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA DM_PAL;
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE
    #PAL_PARAMETER_TBL
    ("PARAM_NAME" VARCHAR(256), "INT_VALUE" INTEGER, "DOUBLE_VALUE" DOUBLE,
"STRING_VALUE" VARCHAR(1000));
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PMML_EXPORT',2,NULL,NULL);
DROP TABLE PAL_ER_DATA_TBL;
CREATE COLUMN TABLE PAL_ER_DATA_TBL ( "ID" INT,"Y" DOUBLE,"X1" DOUBLE, "X2"
DOUBLE);
INSERT INTO PAL_ER_DATA_TBL VALUES (0,0.5,0.13,0.33);
INSERT INTO PAL_ER_DATA_TBL VALUES (1,0.15,0.14,0.34);
INSERT INTO PAL_ER_DATA_TBL VALUES (2,0.25,0.15,0.36);
INSERT INTO PAL_ER_DATA_TBL VALUES (3,0.35,0.16,0.35);
INSERT INTO PAL_ER_DATA_TBL VALUES (4,0.45,0.17,0.37);
INSERT INTO PAL_ER_DATA_TBL VALUES (5,0.55,0.18,0.38);
INSERT INTO PAL_ER_DATA_TBL VALUES (6,0.65,0.19,0.39);
INSERT INTO PAL_ER_DATA_TBL VALUES (7,0.75,0.19,0.31);
INSERT INTO PAL_ER_DATA_TBL VALUES (8,0.85,0.11,0.32);
INSERT INTO PAL_ER_DATA_TBL VALUES (9,0.95,0.12,0.33);
CALL _SYS_AFL.PAL_EXPONENTIAL_REGRESSION(PAL_ER_DATA_TBL,
"#PAL_PARAMETER_TBL", ?, ?, ?, ?);
```

### Expected Result

COEFFICIENT:

| | VARIABLE_NAME | COEFFICIENT_VALUE |
|---|---|---|
| 1 | __PAL_INTERCEPT__ | 2.727731253549633 |
| 2 | X1 | 2.6741406885809225 |
| 3 | X2 | -6.1804265606950315 |

FITTED:

| | ID | VALUE |
|---|---|---|
| 1 | 0 | 0.5023630294167875 |
| 2 | 1 | 0.4850539713017914 |
| 3 | 2 | 0.4402721425239593 |
| 4 | 3 | 0.48103436571450... |
| 5 | 4 | 0.4366236405659918 |
| 6 | 5 | 0.4215796514059779 |
| 7 | 6 | 0.4070540070830719 |
| 8 | 7 | 0.6673936134887157 |
| 9 | 8 | 0.5065608609727283 |
| 10 | 9 | 0.4891071654021395 |

STATISTICS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | R2 | 0.05843444838327394 |
| 2 | F | 0.21721330924892443 |
| 3 | MSE | 0.40167819174428016 |
| 4 | RMSE | 0.6337808704467816 |
| 5 | MAE | 0.4124554961633766 |
| 6 | MAPE | 194.38888646018327 |
| 7 | EXPLAINED_VARIANCE | 0.05843444838327394 |
| 8 | MAX_ERROR | 1.1736248716845736 |

PMML:

| | ROW_INDEX | MODEL_CONTENT |
|---|---|---|
| 1 | 1 | <PMML version="4.0" xmlns="http://www.dmg.org/PMML-4_0" xmlns:xsi="http://www.... |

## _SYS_AFL.PAL_EXPONENTIAL_REGRESSION_PREDICT

This function performs prediction with the exponential regression result.

### Overview of all Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<COEFFICIENT table>` | IN |
| 3 | `<PARAMETER table>` | IN |
| 4 | `<FITTED table>` | OUT |

### Input Tables

| Table | Column | Reference for Output Table | Column Data Type | Description |
|---|---|---|---|---|
| DATA | 1st column | ID | INTEGER, VARCHAR, or NVARCHAR | ID |
| | FEATURE columns | | INTEGER, DOUBLE, or DECIMAL(p, s) | Variable $X_n$ |
| COEFFICIENT | 1st column | | INTEGER, VARCHAR, or NVARCHAR | Variable name or ROW_INDEX for PMML model |
| | 2nd column | | INTEGER, DOUBLE, VARCHAR, NVARCHAR, or CLOB | Values of the coefficients or PMML model content |

The Coefficient Input Table accepts both model formats from the output tables Coefficient and PMML generated by the previous training procedure. To differentiate the two model formats, the predict procedure uses a combination of two column types to identify the model format. Once the model format is determined, the procedure parses the content. Ensure that the column types in the input table correspond to those in either the Coefficient or PMML Model Table, depending on the intended model format.

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| THREAD_RATIO | DOUBLE | 1.0 | Specifies the ratio of total number of threads that can be used by this function. The value range is from 0 to 1, where 0 means only using 1 thread, and 1 means using at most all the currently available threads. Values outside this range will be ignored and this function heuristically determines the number of threads to use. |

### Output Table

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| FITTED | 1st column | ID (in input table) data type | ID (in input table) column name | ID |
| | 2nd column | DOUBLE | VALUE | Value $Y_i$ |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_FER_DATA_TBL;
CREATE COLUMN TABLE PAL_FER_DATA_TBL ("ID" INT,"X1" DOUBLE, "X2" DOUBLE);
INSERT INTO PAL_FER_DATA_TBL VALUES (0,0.5,0.3);
INSERT INTO PAL_FER_DATA_TBL VALUES (1,4,0.4);
INSERT INTO PAL_FER_DATA_TBL VALUES (2,0,1.6);
INSERT INTO PAL_FER_DATA_TBL VALUES (3,0.3,0.45);
INSERT INTO PAL_FER_DATA_TBL VALUES (4,0.4,1.7);
DROP TABLE PAL_FER_COEEFICIENT_TBL;
CREATE COLUMN TABLE PAL_FER_COEEFICIENT_TBL ("VARIABLE_NAME" NVARCHAR(128),
"COEFFICIENT_VALUE" DOUBLE);
INSERT INTO PAL_FER_COEEFICIENT_TBL VALUES ('__PAL_INTERCEPT__',
1.7120914258645001);
INSERT INTO PAL_FER_COEEFICIENT_TBL VALUES ('X1', 0.2652771198483208);
INSERT INTO PAL_FER_COEEFICIENT_TBL VALUES ('X2', -3.471103742302148);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE
    #PAL_PARAMETER_TBL
    ("PARAM_NAME" VARCHAR(256), "INT_VALUE" INTEGER, "DOUBLE_VALUE" DOUBLE,
"STRING_VALUE" VARCHAR(1000));
INSERT INTO #PAL_PARAMETER_TBL VALUES ('THREAD_RATIO',NULL,0.2, NULL);
CALL _SYS_AFL.PAL_EXPONENTIAL_REGRESSION_PREDICT(PAL_FER_DATA_TBL,
PAL_FER_COEEFICIENT_TBL, "#PAL_PARAMETER_TBL", ?);
```

### Expected Result

FITTED:

| | ID | VALUE |
|---|---|---|
| 1 | 0 | 0.6900598931338715 |
| 2 | 1 | 1.2341502316656843 |
| 3 | 2 | 0.00663066413618... |
| 4 | 3 | 0.3887970208571841 |
| 5 | 4 | 0.00521065435714... |

> **Note**
>
> The wrapper procedure generation approach to call PAL functions is still supported. For more information, refer to the older versions (prior to SAP HANA Platform 2.0 SPS 02) of this document.

## 5.3.5 Generalised Linear Models

Generalised linear models (GLM) is used to regress responses satisfying exponential distributions, for example, Normal, Poisson, Binomial, Gamma, inverse Gaussian (IG), and negative binomial (NB). We implement ordinal regression in here. Although it is not a typical generalised linear model, it can be solved similarly, hence we implement it together.

Compared with the classical linear regression, GLM regresses a linear predictor $\eta$ instead of the response itself. The linear predictor and the expected response $\mu$ is connected via link function, for example, $\eta=g(\mu)$ or $\mu=g^{-1}(\eta)$, which guarantees that the regressed responses are in the valid range. In detail, the distribution and their appropriate link function are shown in the following table:

| | | Normal | Poisson | Binomial | Gamma | IG | NB | Ordinal |
|---|---|---|---|---|---|---|---|---|
| Identity | $\eta=\mu$ | Y | Y | N | Y | Y | Y | N |
| Log | $\eta=\log(\mu)$ | Y | Y | Y | Y | Y | Y | N |
| Reciprocal | $\eta=1/\mu$ | Y | N | N | Y | Y | N | N |
| Logit | $\eta=\log\{\mu/(1-\mu)\}$ | N | N | Y | N | N | N | Y |
| Probit | $\eta=\Phi^{-1}(\mu)$ | N | N | Y | N | N | N | Y |
| Comple-mentary log-log | $\eta=\log\{-\log(1-\mu)\}$ | N | N | Y | N | N | N | Y |
| Inverse square | $\eta=1/\mu^2$ | N | N | N | N | Y | N | N |
| Square root | $\eta=\sqrt{\mu}$ | N | N | N | N | N | Y | N |

Given observations $y_i$, $i=1,2,\ldots,n$ of responses, and covariates $x_i$, $i=1, 2,\ldots, n$, where $x_i$ is a p-dimensional vector, the coefficients are to estimated,

$$g(y_i)=\beta_0+x_i^T\beta$$

where $\beta_0$ is the intercept, and $\beta$ is a p-dimensional vector, corresponding to the coefficients with respect to the covariates.

Conventionally, the maximum likelihood estimation (MLE) is adopted. Due to the elegant representation of the exponential distributions and link functions, the Newton-Raphson method for numerical optimisation finally develops to the so-called iteratively reweighted least square (IRLS) equation,

$$X^TWX\beta=X^TWz$$

where $W$ is a diagonal weight matrix, and $z$ is an $n\times1$ vector determined by $\beta$. Therefore, IRLS is an iterative procedure, and degenerates to classical linear regression as normal distribution with identity link function.

Unlike classification and regression problems, specifically, ranking carries out tasks that the response variable with an ordered categorical scale called ordinal, such as "too low", "about right", or "too high". For ordinal scales, there is a clear ordering of the levels, but the absolute distance among them are unknown, unlike interval scales.

One approach for ranking is by ordinal regression, which is solved via cumulative link model (CLM), $g[P(Y_j \le j] = \alpha_j - x_i^T\beta$, where $j=1,2,\ldots,c-1$, $c$ is the number of response categories, and the regression acts on the probabilities of response rather than response itself. Note that each category enjoys its own intercept $\alpha_j$, and it satisfies that $\alpha_1 \le \alpha_2 \le \ldots \le \alpha_{c-1}$. Hence, the objective of ordinal regression takes the cumulative log-likelihood,

$$l(\alpha,\beta)=\sum_{i=1}^{n}\sum_{j=1}^{c} y_{ij}\log\left[g^{-1}(\alpha_j-\beta' x_i)-g^{-1}(\alpha_{j-1}-\beta' x_i)\right]$$

This time, the optimisation problem is solved using Newton-Raphson method, instead of IRLS.

For GLM, as $\beta$ has been estimated, linear predictor can be predicted when new covariates come, and then it can be transformed back to response via inverse link function straightforwardly.

More often than not, GLM might lead to overfitting, which reduces its robustness. Penalty part, both L1 and L2, is usually included in the likelihood function, and such a procedure is called regularization. For example, the elastic net regularization is equivalent to solve

$$\min_{(\beta_0,\beta)\in R^{p+1}} \left[\frac{1}{2N}\sum_{i=1}^{N}(y_i-\beta_0-x_i^T\beta)^2+\lambda P_\alpha(\beta)\right]$$

Where regularization part $P_\alpha(\beta)=(1-\alpha)\frac{1}{2}\|\beta\|_{l_2}^2+\alpha\|\beta\|_{l_1}$, the former of which is the L2 part, and the latter part is L1. When $\alpha=1$, the elastic net regularization degenerates to LASSO regularization.

Due to the non-differentiability of L1 regularization whenever some parameter goes to zero, the IRLS algorithm cannot be used directly. Then in PAL pathwise coordinate descent is employed instead to solve the regularization problem. On the other side, the parameter $\lambda$ is decreased through the pathwise coordinate descent process.

## Prerequisites

- The number of categories should be at least 2 for ranking.
- An ID column should be provided for prediction, and no NULL value is allowed in this column.
- Coefficients are assigned to independent variables, hence all independent variables used in estimation should be present in prediction.

## _SYS_AFL.PAL_GLM

This function estimates the generalised linear models with the maximum likelihood estimation method, using IRLS, Newton-Raphson, or coordinate descent algorithm.

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<STATISTICS table>` | OUT |
| 4 | `<COEFFICIENT table>` | OUT |
| 5 | `<COVARIANCE_VARIANCE table>` | OUT |
| 6 | `<FIT table>` | OUT |

### Signature

### Input Table

| Table | Column | Reference | Data Type | Description | Constraint |
|---|---|---|---|---|---|
| DATA | 1st column | ID | INTEGER, BIGINT, DOUBLE, DECIMAL(p, s), VARCHAR, or NVARCHAR | Data ID. | This must be the first column. |
| | 2nd column | | INTEGER, DOUBLE, DECIMAL(p, s), VARCHAR, or NVARCHAR | Response. | If DEPENDENT_VARIABLE is set to other columns, this column is used as covariates.<br>String type for ordinal regression only. |
| | 3rd column | | INTEGER, DOUBLE or DECIMAL(p, s) | Response | Optional; for Binomial distribution only. |
| | Other columns | | INTEGER, DOUBLE, DECIMAL(p, s), VARCHAR, or NVARCHAR | Covariates. | Optional; when absent, only intercept is estimated. |

> **Note**
>
> For two-column response of binomial distribution $P(k_i, n_i)$, the 2nd column is for $n_i$, whereas the 3rd column is for $k_i$.

**Parameter Table**

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| SOLVER | VARCHAR, or NVARCHAR | irls | Specifies the optimisation algorithm<br>• irls: iterative reweighted least square<br>• nr: Newton-Raphson method<br>• cd: coordinate descent, solves ENET problem | Currently, ordinal regression is solved only via Newton-Raphson method. Meanwhile, Newton-Raphson is only for ordinal regression. |
| FAMILY | VARCHAR or NVARCHAR | gaussian | Specifies the distribution:<br>• gaussian, normal<br>• poisson<br>• binomial<br>• gamma<br>• inversegaussian<br>• negativebinomial<br>• ordinal | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| LINK | VARCHAR or NVARCHAR | gaussian: identity<br>poisson: log<br>binomial: logit<br>gamma: reciprocal<br>inversegaussian: inversesquare<br>negativebinomial: log;<br>ordinal: logit; | Specifies the link function:<br>• identity<br>• log<br>• logit<br>• probit<br>• comploglog<br>• reciprocal, inverse<br>• inversesquare<br>• sqrt | The supported link functions for each distribution are:<br>• gaussian, normal: identity, log, reciprocal, inverse<br>• poisson: identity, log<br>• binomial: logit, probit, comploglog, log<br>• gamma: identity, reciprocal, inverse, log<br>• inversegaussian: inversesquare, identity, reciprocal, inverse, log<br>• negativebinomial: {identity, log, sqrt};<br>• ordinal: {logit, probit, comploglog}; |
| HANDLE_MISSING | INTEGER | 1 | Specify how to handle missing covariates<br>• 0: throw error<br>• 1: remove missing rows<br>• 2: replace missing value with 0 | |
| QUASI | INTEGER | 0 | Indicates whether to use the quasi-likelihood to estimate over-dispersion.<br>• 0: Does not consider over-dispersion<br>• 1: Considers over-dispersion | Valid only for Poisson and Negative Binomial distribution. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| GROUP_RESPONSE | INTEGER | 0 | Indicates whether the response consists of two columns.<br>• 0: Consists of only one column (the 2nd column)<br>• 1: Consists of two columns (the 2nd and 3rd columns) | Valid only for Binomial distribution, n,k for $\binom{n}{k}$, respectively. |
| MAX_ITERATION | INTEGER | 100 for IRLS and Newton-Raphson;<br>1e5 for coordinate descent | The maximum number of iterations for numeric optimization. | |
| CONVERGENCE_CRITERION | DOUBLE | 1e-8 for IRLS;<br>1e-6 for Newton-Raphson;<br>1e-7 for coordinate descent | The convergence criterion of coefficients for numeric optimization. | IRLS and coordinate descent evaluate the relative change of objective, whereas Newton-Raphson evaluates the absolute multitude of gradient<br>. |
| SIGNIFICANCE_LEVEL | DOUBLE | 0.05 | The significance level for the confidence interval of estimated coefficients. | Not for coordinate descent. |
| OUTPUT_FITTED | INTEGER | 0 | Indicates whether to output the fitted response.<br>• 0: Does not output the fitted response<br>• 1: Outputs the fitted response | Valid only if HAS_ID is 1. |
| ENET_ALPHA | DOUBLE | 1.0 | The elastic net mixing parameter. The value range is between 0 and 1 inclusively. | Valid only for coordinate descent |
| ENET_LAMBDA | DOUBLE | No default value | Coefficient for L1 and L2 regularization. | Valid only for coordinate descent. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| ENET_NUM_LAMBDA | INTEGER | 100 | The number of lambda values. | Valid only for coordinate descent. Takes effect only when ENET_LAMBDA is not given explicitly. |
| LAMBDA_MIN_RATIO | DOUBLE | 1e-2 or 1e-4 | The smallest value of lambda, as a fraction of the maximum lambda, where $\lambda_{max}$ is the smallest value for which all coefficients are zero. | Valid only for coordinate descent. Takes effect only when ENET_LAMBDA is not given explicitly.<br>When the number of observations is smaller than the number of covariates, LAMBDA_MIN_RATIO defaults to 1e-2; Otherwise, it defaults to 1e-4. |
| HAS_ID | INTEGER | 1 | Specifies if the first column of the DATA table is used as ID column<br>• 0: no<br>• 1: yes | If HAS_ID is 0, no fitted is output. |
| DEPENDENT_VARIABLE | VARCHAR or NVARCHAR | The second column name of the DATA table | Specifies the variable used as dependent. | Cannot be the first column name if HAS_ID is 1. Not used if GROUP_RESPONSE is 1 in Binomial regression. |
| CATEGORICAL_VARIABLE | VARCHAR, or NVARCHAR | Detected from input data | Indicate which variables are cheated as categorical. The default behaviour is:<br>• string: categorical<br>• integer and double: continuous | Valid only for integer or string column. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| ORDERING | VARCHAR, or NVARCHAR | No default value | Specifies the categories orders for ordinal regression. | For string category only, separated by comma.<br>The default behavior is that integer category order by its numeric value, whereas the string category order by its alphabet order. |
| EXCLUDED_FEATURE | VARCHAR or NVARCHAR | No default value | Excludes the indicated feature column. | Cannot be the ID column name and the column name of dependent variable. |

### Model Evaluation and Parameter Selection

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| RESAMPLING_METHOD | NVARCHAR | No default value | Specifies the resampling method for model evaluation or parameter selection.<br>• cv<br>• bootstrap<br>If no value is specified for this parameter, neither model evaluation nor parameter selection is activated. | |
| EVALUATION_METRIC | NVARCHAR | No default value | Specifies the evaluation metric for model evaluation or parameter selection.<br>• RMSE<br>• MAE<br>• ERROR_RATE | Only ordinal regression use ERROR_RATE. |
| FOLD_NUM | INTEGER | No default value | Specifies the fold number for a cross validation method. | Mandatory and valid only when RESAMPLING_METHOD is set to cv. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| REPEAT_TIMES | INTEGER | 1 | Specifies the number of repeat times for resampling. | |
| PARAM_SEARCH_STRATEGY | NVARCHAR | No default value | Specifies the method to activate parameter selection.<br>• grid<br>• random | |
| RANDOM_SEARCH_TIMES | INTEGER | No default value | Specifies the number of times to randomly select candidate parameters for selection. | Mandatory and valid when PARAM_SEARCH_STRATEGY is set to random. |
| SEED | INTEGER | 0 | Specifies the seed for random generation. Use system time when 0 is specified. | |
| TIMEOUT | INTEGER | 0 | Specifies maximum running time for model evaluation or parameter selection, in seconds. No timeout when 0 is specified. | |
| PROGRESS_INDICATOR_ID | NVARCHAR | No default value | Sets an ID of progress indicator for model evaluation or parameter selection. No progress indicator is active if no value is provided. | |
| LINK_VALUES | NVARCHAR | No default value | The discrete set for candidate of link functions. | |
| ENET_LAMBDA_VALUES | NVARCHAR | No default value | The discrete set for ENET_LAMBDA candidate. | |
| ENET_LAMBDA_RANGE | NVARCHAR | No default value | The range for ENET_LAMBDA candidate. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| ENET_ALPHA_VALUES | NVARCHAR | No default value | The discrete set for ALPHA_VALUES candidate. | |
| ENET_ALPHA_RANGE | NVARCHAR | No default value | The range for ENET_ALPHA candidate. | |

### Output Tables

| Table | Column | Data Type | Column Name | Description | Constraint |
|---|---|---|---|---|---|
| STATISTICS | 1st column | NVARCHAR(1000) | STAT_NAME | Name | |
| | 2nd column | NVARCHAR(1000) | STAT_VALUE | Value | |
| COEFFICIENT | 1st column | NVARCHAR(256) | VARIABLE_NAME | variable name; for the intercept, specified "__PAL_INTCP__" is used. | |
| | 2nd column | DOUBLE | COEFFICIENT | Coefficient beta | |
| | 3rd column | DOUBLE | SE | The corresponding standard error. | Valid only for IRLS and NR |
| | 4th column | DOUBLE | SCORE | Score (z-value, or t-value). | Valid only for IRLS and NR |
| | 5th column | DOUBLE | PROBABILITY | Probability Pr(>\|z\|), or Pr(>\|t\|). | Valid only for IRLS and NR |
| | 6th column | DOUBLE | CI_LOWER | The lower of confidence interval. | Valid only for IRLS and NR |
| | 7th column | DOUBLE | CI_UPPER | The upper of confidence interval. | Valid only for IRLS and NR |
| VCOV (Valid only for IRLS and NR) | 1st column | NVARCHAR(256) | VARIABLE_I | i-th variable name. | |
| | 2nd column | NVARCHAR(256) | VARIABLE_J | j-th variable name. | |
| | 3rd column | DOUBLE | COVARIANCE | Element of variance-covariance matrix. | Low-triangular as it is a symmetric matrix. |

| Table | Column | Data Type | Column Name | Description | Constraint |
|---|---|---|---|---|---|
| FIT | 1st column | ID column type | ID column name | ID, in respect with that in the DATA table. | |
| | 2nd column | DOUBLE | PREDICTOR | The fitted linear predictor η. | Not for ordinal regression |
| | 3rd column | DOUBLE | RESPONSE | The fitted response μ. | For ordinal regression, it indicates the estimated probability that the instance belonging to its category. |

### Examples

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_GLM_DATA_TBL;
CREATE COLUMN TABLE PAL_GLM_DATA_TBL(
    "ID" INTEGER,
    "Y" INTEGER,
    "X" INTEGER
);
INSERT INTO PAL_GLM_DATA_TBL VALUES (1, 0, -1);
INSERT INTO PAL_GLM_DATA_TBL VALUES (2, 0, -1);
INSERT INTO PAL_GLM_DATA_TBL VALUES (3, 1, 0);
INSERT INTO PAL_GLM_DATA_TBL VALUES (4, 1, 0);
INSERT INTO PAL_GLM_DATA_TBL VALUES (5, 1, 0);
INSERT INTO PAL_GLM_DATA_TBL VALUES (6, 1, 0);
INSERT INTO PAL_GLM_DATA_TBL VALUES (7, 2, 1);
INSERT INTO PAL_GLM_DATA_TBL VALUES (8, 2, 1);
INSERT INTO PAL_GLM_DATA_TBL VALUES (9, 2, 1);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR(100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRIN_VALUE" VARCHAR (100)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SOLVER', null, null, 'irls'); -- IRLS
--INSERT INTO #PAL_PARAMETER_TBL VALUES ('SOLVER', null, null, 'cd'); --
coordinate descent
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FAMILY', null, null, 'poisson');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('LINK', null, null, 'log');
/*----- this part for ordinal -------
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SOLVER', null, null, 'nr'); -- Newton-
Raphson
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FAMILY', null, null, 'ordinal');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('LINK', null, null, 'logit');
*/-------- end ordinal ---------
INSERT INTO #PAL_PARAMETER_TBL VALUES ('OUTPUT_FITTED', 1, null, null);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SIGNIFICANCE_LEVEL', null, 0.05, null);
```

```
--INSERT INTO #PAL_PARAMETER_TBL VALUES ('HAS_ID', 1, null, null);
--INSERT INTO #PAL_PARAMETER_TBL VALUES ('CATEGORICAL_VARIABLE', null, null,
'X');
--INSERT INTO #PAL_PARAMETER_TBL VALUES ('DEPENDENT_VARIABLE', null, null, 'Y');
--INSERT INTO #PAL_PARAMETER_TBL VALUES ('ORDERING', null, null, 'low, right,
high'); -- specify ordinal
DROP TABLE PAL_GLM_STATS_TBL;
CREATE COLUMN TABLE PAL_GLM_STATS_TBL (
    "STAT_NAME" NVARCHAR(1000),
    "STAT_VALUE" NVARCHAR(1000)
);
DROP TABLE PAL_GLM_COEFF_TBL;
CREATE COLUMN TABLE PAL_GLM_COEFF_TBL (
    "VARIABLE_NAME" NVARCHAR(256),
    "COEFFICIENT" DOUBLE,
    "SE" DOUBLE,
    "SCORE" DOUBLE,
    "PROBABILITY" DOUBLE,
    "CI_LOWER"    DOUBLE,
    "CI_UPPER" DOUBLE
);
DROP TABLE PAL_GLM_VCOV_TBL;
CREATE COLUMN TABLE PAL_GLM_VCOV_TBL (
    "VARIABLE_I" NVARCHAR(256),
    "VARIABLE_J" NVARCHAR(256),
    "COVARIANCE" DOUBLE
);
CALL _SYS_AFL.PAL_GLM(PAL_GLM_DATA_TBL, "#PAL_PARAMETER_TBL", PAL_GLM_STATS_TBL,
PAL_GLM_COEFF_TBL, PAL_GLM_VCOV_TBL, ?) WITH OVERVIEW;
```

**Expected Result**

**Expected Result (Poisson, IRLS):**

STATISTICS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | family | poisson |
| 2 | link | log |
| 3 | solver | IRLS |
| 4 | quasi | 0 |
| 5 | p0 | 1 |
| 6 | p | 1 |
| 7 | c | 1 |
| 8 | iter | 4 |
| 9 | converged | 1 |
| 10 | intercept | -0.294958 |
| 11 | phi | 1 |
| 12 | null-degree | 8 |
| 13 | null-devia... | 6.21055 |
| 14 | residual-d... | 7 |
| 15 | residual-d... | 1.37909 |
| 16 | log-lik | -8.6101 |
| 17 | aic | 21.2202 |

COEFFICIENT:

| | VARIABLE_NAME | COEFFICIENT | SE | SCORE | PROBABILITY | CI_LOWER | CI_UPPER |
|---|---|---|---|---|---|---|---|
| 1 | __PAL_INTCP__ | -0.2949582846116588 | 0.45300037602353704 | -0.6511215006062894 | 0.5149680583498653 | -1.1828227066008932 | 0.5929061373775757 |
| 2 | X | 1.0698198067208362 | 0.5405998230178941 | 1.978949605918433 | 0.04782168036295946 | 0.010263623557036361 | 2.129375989884636 |

VCOV:

| | VARIABLE_I | VARIABLE_J | COVARIANCE |
|---|---|---|---|
| 1 | __PAL_INTCP__ | __PAL_INTCP__ | 0.20520934067746596 |
| 2 | __PAL_INTCP__ | X | -0.1753489011651363 |
| 3 | X | X | 0.2922481686469784 |

FITTED:

| | ID | PREDICTOR | RESPONSE |
|---|---|---|---|
| 1 | 1 | -1.364778091332495 | 0.25543735346197155 |
| 2 | 2 | -1.364778091332495 | 0.25543735346197155 |
| 3 | 3 | -0.2949582846116... | 0.744562646538029 |
| 4 | 4 | -0.2949582846116... | 0.744562646538029 |
| 5 | 5 | -0.2949582846116... | 0.744562646538029 |
| 6 | 6 | -0.2949582846116... | 0.744562646538029 |
| 7 | 7 | 0.77486152210917... | 2.170291568974647 |
| 8 | 8 | 0.77486152210917... | 2.170291568974647 |
| 9 | 9 | 0.77486152210917... | 2.170291568974647 |

**Expected Result (Poisson, Coordinate descent):**

STATISTICS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | family | poisson |
| 2 | link | log |
| 3 | solver | Coordinate... |
| 4 | quasi | 0 |
| 5 | p0 | 1 |
| 6 | p | 1 |
| 7 | c | 1 |
| 8 | iter | 276 |
| 9 | converged | 1 |
| 10 | intercept | -0.290265 |
| 11 | phi | 1 |
| 12 | null-degree | 0 |
| 13 | null-devia... | 6.21055 |
| 14 | residual-d... | 1 |
| 15 | residual-d... | 1.3793 |
| 16 | log-lik | -8.61021 |
| 17 | aic | 21.2204 |
| 18 | alpha | 1 |
| 19 | lambda | 0.0037956 |

COEFFICIENT:

| | VARIABLE_NAME | COEFFICIENT | SE | SCORE | PROBABILITY | CI_LOWER | CI_UPPER |
|---|---|---|---|---|---|---|---|
| 1 | __PAL_INTCP__ | -0.29026539088181846 | ? | ? | ? | ? | ? |
| 2 | X | 1.0619807656046651 | ? | ? | ? | ? | ? |

VCOV:

empty

FIT:

| | ID | PREDICTOR | RESPONSE |
|---|---|---|---|
| 1 | 1 | -1.3522461564864832 | 0.2586586199254647 |
| 2 | 2 | -1.3522461564864832 | 0.2586586199254647 |
| 3 | 3 | -0.2902653908818184 | 0.7480650115991716 |
| 4 | 4 | -0.2902653908818184 | 0.7480650115991716 |
| 5 | 5 | -0.2902653908818184 | 0.7480650115991716 |
| 6 | 6 | -0.2902653908818184 | 0.7480650115991716 |
| 7 | 7 | 0.7717153747228472 | 2.1634742416089763 |
| 8 | 8 | 0.7717153747228472 | 2.1634742416089763 |
| 9 | 9 | 0.7717153747228472 | 2.1634742416089763 |

**Expected Result (Ordinal, Newton-Raphson):**

STATS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | family | ordinal |
| 2 | link | logit |
| 3 | solver | Newton-R... |
| 4 | p0 | 1 |
| 5 | p | 1 |
| 6 | c | 2 |
| 7 | iter | 16 |
| 8 | converged | 1 |
| 9 | intercept | -16.6563, 1... |
| 10 | factor-type | integer |
| 11 | factor | 0, 1, 2 |
| 12 | max.g | 5.99185e-07 |
| 13 | cond.H | 4.40955 |
| 14 | phi | 9.67094e-15 |
| 15 | log-lik | -5.24718e-07 |
| 16 | aic | 6 |

COEFFICIENT:

| | VARIABLE_NAME | COEFFICIENT | SE | SCORE | PROBABILITY | CI_LOWER | CI_UPPER |
|---|---|---|---|---|---|---|---|
| 1 | __PAL_INTCP__.\|:\|.I:0\|1 | -16.65633004002157 | 2,024.9740549715766 | -0.008225453555381658 | 0.9934371116080548 | -3,985.5325474123433 | 3,952.2198873323 |
| 2 | __PAL_INTCP__.\|:\|.I:1\|2 | 16.60177938489661 | 1,972.8079963115092 | 0.008415304183649084 | 0.9932856379663817 | -3,850.0308417982897 | 3,883.2344005680825 |
| 3 | X | 35.15883798436384 | 4,954.122624492695 | 0.007096884887455552 | 0.9943375526508023 | -9,674.743081016368 | 9,745.060756985094 |

VCOV:

| | VARIABLE_I | VARIABLE_J | COVARIANCE |
|---|---|---|---|
| 1 | __PAL_INTCP__.\|:\|.I:0\|1 | __PAL_INTCP__.\|:\|.I:0\|1 | 4,100,519.9233080293 |
| 2 | __PAL_INTCP__.\|:\|.I:0\|1 | __PAL_INTCP__.\|:\|.I:1\|2 | -172,270.81893778697 |
| 3 | __PAL_INTCP__.\|:\|.I:1\|2 | __PAL_INTCP__.\|:\|.I:1\|2 | 3,891,971.390310632 |
| 4 | __PAL_INTCP__.\|:\|.I:0\|1 | X | -1,795,268.90693765... |
| 5 | __PAL_INTCP__.\|:\|.I:1\|2 | X | 2,355,137.3569104075 |
| 6 | X | X | 24,543,330.978510387 |

FIT:

| | ID | PREDICTOR | RESPONSE |
|---|---|---|---|
| 1 | 1 | ? | 0.9999999907856884 |
| 2 | 2 | ? | 0.9999999907856884 |
| 3 | 3 | ? | 0.9999998799713853 |
| 4 | 4 | ? | 0.9999998799713853 |
| 5 | 5 | ? | 0.9999998799713853 |
| 6 | 6 | ? | 0.9999998799713853 |
| 7 | 7 | ? | 0.9999999912748712 |
| 8 | 8 | ? | 0.9999999912748712 |
| 9 | 9 | ? | 0.9999999912748712 |

### Model Evaluation and Parameter Selection

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_GLM_DATA_TBL;
CREATE COLUMN TABLE PAL_GLM_DATA_TBL(
    "ID" INTEGER,
    "Y1" INTEGER,
    "Y2" INTEGER,
    "X" DOUBLE
);
INSERT INTO PAL_GLM_DATA_TBL VALUES (1, 59, 56, 1.6907);
INSERT INTO PAL_GLM_DATA_TBL VALUES (2, 60, 53, 1.7242);
INSERT INTO PAL_GLM_DATA_TBL VALUES (3, 62, 58, 1.7552);
INSERT INTO PAL_GLM_DATA_TBL VALUES (4, 56, 48, 1.7842);
INSERT INTO PAL_GLM_DATA_TBL VALUES (5, 63, 62, 1.8113);
INSERT INTO PAL_GLM_DATA_TBL VALUES (6, 59, 53, 1.8369);
INSERT INTO PAL_GLM_DATA_TBL VALUES (7, 62, 61, 1.8610);
INSERT INTO PAL_GLM_DATA_TBL VALUES (8, 60, 59, 1.8839);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR(100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRIN_VALUE" VARCHAR (100)
);
--INSERT INTO #PAL_PARAMETER_TBL VALUES ('SOLVER', null, null, 'irls'); -- IRLS
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SOLVER', null, null, 'cd'); --
coordinate descent
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FAMILY', null, null, 'binomial');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('GROUP_RESPONSE', 1, null, null);
--INSERT INTO #PAL_PARAMETER_TBL VALUES ('LINK', null, null, 'log');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('ENET_ALPHA_VALUES', null, null, '{0.16,
0.15, 0.2}');
--INSERT INTO #PAL_PARAMETER_TBL VALUES ('ENET_LAMBDA_VALUES', null, null,
'{0.0012, 0.001, 0.0015}');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('ENET_LAMBDA_RANGE', null, null,
'[0.0005, 0.001, 0.003]');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('LINK_VALUES', null, null, '{"probit",
"log", "logit", "comploglog"}');
/*cross validation*/
INSERT INTO #PAL_PARAMETER_TBL VALUES ('RESAMPLING_METHOD', NULL, NULL, 'cv');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('TIMEOUT',100, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('EVALUATION_METRIC', NULL, NULL, 'RMSE');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FOLD_NUM', 4, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('REPEAT_TIMES', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SEED', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PROGRESS_INDICATOR_ID', NULL, NULL,
'TEST');
```

```
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PARAM_SEARCH_STRATEGY', NULL, NULL,
'grid');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('OUTPUT_FITTED', 1, null, null);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SIGNIFICANCE_LEVEL', null, 0.05, null);
DROP TABLE PAL_GLM_STATS_TBL;
CREATE COLUMN TABLE PAL_GLM_STATS_TBL (
    "STAT_NAME" NVARCHAR(1000),
    "STAT_VALUE" NVARCHAR(1000)
);
DROP TABLE PAL_GLM_COEFF_TBL;
CREATE COLUMN TABLE PAL_GLM_COEFF_TBL (
    "VARIABLE_NAME" NVARCHAR(256),
    "COEFFICIENT" DOUBLE,
    "SE" DOUBLE,
    "SCORE" DOUBLE,
    "PROBABILITY" DOUBLE,
    "CI_LOWER"    DOUBLE,
    "CI_UPPER" DOUBLE
);
DROP TABLE PAL_GLM_VCOV_TBL;
CREATE COLUMN TABLE PAL_GLM_VCOV_TBL (
    "VARIABLE_I" NVARCHAR(256),
    "VARIABLE_J" NVARCHAR(256),
    "COVARIANCE" DOUBLE
COEFFICIENT:
);
--CALL _SYS_AFL.PAL_GLM(PAL_GLM_DATA_TBL, "#PAL_PARAMETER_TBL",
PAL_GLM_STATS_TBL, PAL_GLM_COEFF_TBL, PAL_GLM_VCOV_TBL, ?) WITH OVERVIEW;
CALL _SYS_AFL.PAL_GLM(PAL_GLM_DATA_TBL, "#PAL_PARAMETER_TBL", ?, ?, ?, ?)
```

**Expected Result (Binomial, Coordinate descent, cross validation):**

STATS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | timeout | FALSE |
| 2 | TEST_RMSE | 0.0622041, 0.0861057, 0.... |
| 3 | TEST_RMS... | 0.067863 |
| 4 | TEST_RMS... | 0.000193345 |
| 5 | EVAL_RES... | {"candidates":[{"TEST_R... |
| 6 | EVAL_RES... | 067515547,"TEST_RMSE.... |
| 7 | EVAL_RES... | ,0.05427278630631054,0.... |
| 8 | family | binomial |
| 9 | link | log |
| 10 | solver | Coordinate descent |
| 11 | grouped | 1 |
| 12 | quasi | 0 |
| 13 | p0 | 1 |
| 14 | p | 1 |
| 15 | c | 1 |
| 16 | iter | 126 |
| 17 | converged | 1 |
| 18 | intercept | -0.860677 |
| 19 | phi | 1 |
| 20 | null-degree | 0 |
| 21 | null-devia... | 41.8321 |
| 22 | residual-d... | 1 |
| 23 | residual-d... | 13.6881 |
| 24 | log-lik | -18.3985 |
| 25 | aic | 40.7971 |
| 26 | alpha | 0.2 |
| 27 | lambda | 0.0209784 |

COEFFICIENT:

| | VARIABLE_NAME | COEFFICIENT | SE | SCORE | PROBABILITY | CI_LOWER | CI_UPPER |
|---|---|---|---|---|---|---|---|
| 1 | __PAL_INTCP__ | -0.8606772691656173 | ? | ? | ? | ? | ? |
| 2 | X | 0.44196147482193515 | ? | ? | ? | ? | ? |

VCOV:

empty

FIT:

| | ID | PREDICTOR | RESPONSE |
|---|---|---|---|
| 1 | 1 | -0.114218042083277 | 0.8920634273073659 |
| 2 | 2 | -0.09920907856574324 | 0.9055533564325918 |
| 3 | 3 | -0.08532018695250294 | 0.9182182358300485 |
| 4 | 4 | -0.0723273528626976 | 0.9302263337109908 |
| 5 | 5 | -0.060185773420155... | 0.941589595015835 |
| 6 | 6 | -0.04871623712018922 | 0.9524513616970613 |
| 7 | 7 | -0.03791874396279924 | 0.9627911703080014 |
| 8 | 8 | -0.027658885319470... | 0.9727201193279509 |

## _SYS_AFL.PAL_GLM_PREDICT

This function predicts based on generalised linear models.

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<STATISTICS table>` | IN |
| 3 | `<COEFFICIENT table>` | IN |
| 4 | `<COVARIANCE_VARIANCE table>` | IN |
| 5 | `<PARAMETER table>` | IN |
| 6 | `<RESULT table>` | OUT |

### Signature

### Input Tables

| Table | Column | Reference for Output Table | Data Type | Description | Constraint |
|---|---|---|---|---|---|
| DATA | 1st column | ID | INTEGER, BIGINT, DOUBLE, DECIMAL(p, s), VARCHAR, or NVARCHAR | ID. | This column should be the first one. |

| Table | Column | Reference for Output Table | Data Type | Description | Constraint |
|---|---|---|---|---|---|
| | Other columns | | INTEGER, DOUBLE, or DECIMAL(p, s) | Covariates. | Should include all variables in estimate stage. |
| STATS | 1st column | | VARCHAR or NVARCHAR | Stats name. | |
| | 2nd column | | VARCHAR or NVARCHAR | Stats value. | |
| COEFFICIENT | 1st column | | VARCHAR or NVARCHAR | Variable name. | |
| | 2nd column | | DOUBLE | Coefficient beta. | |
| | 3rd column | | DOUBLE | The corresponding standard error | |
| | 4th column (optional) | | DOUBLE | Score (z-value or t-value).. | |
| | 5th column (optional) | | DOUBLE | Probability Pr(>\|z\|), or Pr(>\|t\|). | |
| | 6th column (optional) | | DOUBLE | The lower limit of confidence interval. | |
| | 7th column (optional) | | DOUBLE | The upper limit of confidence interval. | |
| VCOV | 1st column | | VARCHAR or NVARCHAR | Variable name. | |
| | 2nd column | | VARCHAR or NVARCHAR | Variable name. | |
| | 3rd column | | DOUBLE | Element of variance-covariance matrix. | |

> **Note**
>
> The 4th to 7th columns of the Coefficient table are optional, only its first 3 columns are used.

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description | Constraint |
|---|---|---|---|---|
| TYPE | VARCHAR or NVARCHAR | response | The prediction type:<br>• response: Predicts response μ<br>• link: Predicts linear predictor η | Not for ordinal regression. |
| SIGNIFICANCE_LEVEL | DOUBLE | 0.05 | The significance level for the confidence interval and prediction interval. | Valid only for IRLS. |
| HANDLE_MISSING | INTEGER | 2 | Specify how to handle missing covariates.<br>• 1: Remove missing rows.<br>• 2: Replace missing with 0. | |

### Output Table

| Table | Column | Data Type | Column Name | Description | Constraint |
|---|---|---|---|---|---|
| RESULT | 1st column | ID column type | ID column name | ID, in consistent with that in the DATA table. | |
| | 2nd column | NVARCHAR(100) | PREDICTION | Predicted value. | |
| | 3rd column | DOUBLE | SE | Standard error for GLM prediction, or probability the instance belonging to the estimated category for ordinal regression | Valid only for IRLS |
| | 4th column | DOUBLE | CI_LOWER | Lower limit of confidence interval. | Valid only for IRLS. |

| Table | Column | Data Type | Column Name | Description | Constraint |
|---|---|---|---|---|---|
| | 5th column | DOUBLE | CI_UPPER | Upper limit of confidence interval. | Valid only for IRLS. |
| | 6th column | DOUBLE | PI_LOWER | Lower limit of prediction interval. | Valid only for IRLS. |
| | 7th column | DOUBLE | PI_UPPER | Upper limit of prediction interval. | Valid only for IRLS. |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_GLM_DATA_TBL;
CREATE COLUMN TABLE PAL_GLM_DATA_TBL (
    "ID" INTEGER,
    "X" INTEGER
);
INSERT INTO PAL_GLM_DATA_TBL VALUES (1, -1);
INSERT INTO PAL_GLM_DATA_TBL VALUES (2, -1);
INSERT INTO PAL_GLM_DATA_TBL VALUES (3, 0);
INSERT INTO PAL_GLM_DATA_TBL VALUES (4, 0);
INSERT INTO PAL_GLM_DATA_TBL VALUES (5, 0);
INSERT INTO PAL_GLM_DATA_TBL VALUES (6, 0);
INSERT INTO PAL_GLM_DATA_TBL VALUES (7, 1);
INSERT INTO PAL_GLM_DATA_TBL VALUES (8, 1);
INSERT INTO PAL_GLM_DATA_TBL VALUES (9, 1);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR(100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRIN_VALUE" VARCHAR (100)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SIGNIFICANCE_LEVEL', null, 0.05, null);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('TYPE', null, null, 'response');
CALL _SYS_AFL.PAL_GLM_PREDICT (PAL_GLM_DATA_TBL, PAL_GLM_STATS_TBL,
PAL_GLM_COEFF_TBL, PAL_GLM_VCOV_TBL, "#PAL_PARAMETER_TBL", ?);
```

### Expected Result

RESULT (Poisson regression with IRLS):

| | ID | PREDICTION | SE | CI_LOWER | CI_UPPER | PI_LOWER | PI_UPPER |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 0.25543735346197155 | 0.23524591949996063 | 0.042011422102219524 | 1.5531071855862983 | 0.017786991550023282 | 3.6683123933665307 |
| 2 | 2 | 0.25543735346197155 | 0.23524591949996063 | 0.042011422102219524 | 1.5531071855862983 | 0.017786991550023282 | 3.6683123933665307 |
| 3 | 3 | 0.744562646538029 | 0.33728715885480703 | 0.30641260387456243 | 1.8092386788588513 | 0.08658346030084753 | 6.4027648316833 |
| 4 | 4 | 0.744562646538029 | 0.33728715885480703 | 0.30641260387456243 | 1.8092386788588513 | 0.08658346030084753 | 6.4027648316833 |
| 5 | 5 | 0.744562646538029 | 0.33728715885480703 | 0.30641260387456243 | 1.8092386788588513 | 0.08658346030084753 | 6.4027648316833 |
| 6 | 6 | 0.744562646538029 | 0.33728715885480703 | 0.30641260387456243 | 1.8092386788588513 | 0.08658346030084753 | 6.4027648316833 |
| 7 | 7 | 2.1702915689746476 | 0.8314219794103891 | 1.024305088120602 | 4.5984009539625195 | 0.2660682292934585 | 17.70284827643734 |
| 8 | 8 | 2.1702915689746476 | 0.8314219794103891 | 1.024305088120602 | 4.5984009539625195 | 0.2660682292934585 | 17.70284827643734 |
| 9 | 9 | 2.1702915689746476 | 0.8314219794103891 | 1.024305088120602 | 4.5984009539625195 | 0.2660682292934585 | 17.70284827643734 |

RESULT (Poisson regression with coordinate descent):

| | ID | PREDICTION | SE | CI_LOWER | CI_UPPER | PI_LOWER | PI_UPPER |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 0.2586586199254646 | ? | ? | ? | ? | ? |
| 2 | 2 | 0.2586586199254646 | ? | ? | ? | ? | ? |
| 3 | 3 | 0.7480650115991715 | ? | ? | ? | ? | ? |
| 4 | 4 | 0.7480650115991715 | ? | ? | ? | ? | ? |
| 5 | 5 | 0.7480650115991715 | ? | ? | ? | ? | ? |
| 6 | 6 | 0.7480650115991715 | ? | ? | ? | ? | ? |
| 7 | 7 | 2.1634742416089754 | ? | ? | ? | ? | ? |
| 8 | 8 | 2.1634742416089754 | ? | ? | ? | ? | ? |
| 9 | 9 | 2.1634742416089754 | ? | ? | ? | ? | ? |

RESULT (Ordinal regression with Newton-Raphson):

| | ID | PREDICTION | SE | CI_LOWER | CI_UPPER | PI_LOWER | PI_UPPER |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 0.9999999907856884 | ? | ? | ? | ? |
| 2 | 2 | 0 | 0.9999999907856884 | ? | ? | ? | ? |
| 3 | 3 | 1 | 0.9999998799713853 | ? | ? | ? | ? |
| 4 | 4 | 1 | 0.9999998799713853 | ? | ? | ? | ? |
| 5 | 5 | 1 | 0.9999998799713853 | ? | ? | ? | ? |
| 6 | 6 | 1 | 0.9999998799713853 | ? | ? | ? | ? |
| 7 | 7 | 2 | 0.9999999912748712 | ? | ? | ? | ? |
| 8 | 8 | 2 | 0.9999999912748712 | ? | ? | ? | ? |
| 9 | 9 | 2 | 0.9999999912748712 | ? | ? | ? | ? |

## Related Information

Cox Proportional Hazard Model [page 438]
Field-Aware Factorization Machine [page 1354]

# 5.3.6 Multiple Linear Regression

Linear regression is an approach to model the linear relationship between a variable $\mathbf{Y} \in \mathbf{R}$, usually referred to as dependent variable, and one or more variables, usually referred to as independent variables, denoted as predictor vector $\mathbf{X} \in \mathbf{R}^n$.

In linear regression, data are modeled using linear functions, and unknown model parameters are estimated from the data. Such models are called linear models.

Assume we have `m` observation pairs $(x_i, y_i)$. Then we obtain an overdetermined linear system $[\ 1\ \ X\ ]\begin{bmatrix}\beta_0\\ \beta\end{bmatrix} = \mathbf{Y}$, with $[\ 1\ \ X\ ]$ is $m \times (n+1)$ matrix, $\begin{bmatrix}\beta_0\\ \beta\end{bmatrix}$ is $(n+1) \times 1$ matrix, and $\mathbf{Y}$ is $m \times 1$ matrix, where $m > n+1$. Since

equality is usually not exactly satisfiable, when `m > n+1`, the least squares solution minimizes the squared Euclidean norm of the residual vector $r\left(\begin{bmatrix}\beta_0\\ \boldsymbol{\beta}\end{bmatrix}\right)$ so that

$$\min_{(\beta_0,\boldsymbol{\beta})\in\mathbb{R}^{n+1}} \left\| r\left(\begin{bmatrix}\beta_0\\ \boldsymbol{\beta}\end{bmatrix}\right)\right\|_2^2 = \min_{(\beta_0,\boldsymbol{\beta})\in\mathbb{R}^{n+1}} \sum_{i=1}^{m} (y_i - \beta_0 - x_i\ \boldsymbol{\beta})^2$$

Current implementation provides two stored procedures for `Multiple Linear Regression` algorithm.

`_SYS_AFL.PAL_LINEAR_REGRESSION` takes training data and configurations as input and output the fitted linear model and also various statistics measuring the model.

`_SYS_AFL.PAL_LINEAR_REGRESSION_PREDICTION` takes prediction data and linear model as input and output the prediction result for each row of prediction data.

For details about the interface of stored procedures and guides to use these procedures, please refer to description for each stored procedure below.

## _SYS_AFL.PAL_LINEAR_REGRESSION

Prerequisites

- Given `n` independent variables, if all numerical, there must be at least `n+1` records available for analysis.

This is fitting function for multiple linear regression.

**Overview of All Tables**

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | DATA table | IN |
| 2 | PARAMETER table | IN |
| 3 | COEFFICIENT table | OUT |
| 4 | SEMISTRUCTURED_RESULT table | OUT |
| 5 | FITTED table | OUT |
| 6 | STATISTICS table | OUT |
| 7 | OPTIMAL_PARAM table | OUT |

### Input Table

### Data Table

| Column | Reference for Output Table | Data Type | Description |
|---|---|---|---|
| 1st column | ID | INTEGER, VARCHAR, or NVARCHAR | If this column is absent, the HAS_ID parameter must be set to 0. |

| Column | Reference for Output Table | Data Type | Description |
|---|---|---|---|
| 2nd column | None | INTEGER, DOUBLE, or DECIMAL(p, s) | Variable $y$ |
| Feature columns | None | INTEGER, DOUBLE, DECIMAL(p, s), VARCHAR, or NVARCHAR | Variable $x_m$ |

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| THREAD_RATIO | DOUBLE | 1.0 | Specifies the ratio of total number of threads that can be used by this function. The value range is from 0 to 1, where 0 means only using 1 thread, and 1 means using at most all the currently available threads. Values outside the range will be ignored and this function heuristically determines the number of threads to be used. | Only valid when ALG is 1, 4, 5 or 6. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| ALG | INTEGER | 1 | Specifies algorithms for solving the least square problem:<br>• 1: QR decomposition<br>• 2: SVD (numerically stable and can handle rank deficiency but computationally<br>• 4: Cyclical coordinate descent method to solve elastic net regularized multiple linear regression<br>• 5: Cholesky decomposition (fast but numerically unstable)<br>• 6: Alternating direction method of multipliers (ADMM) to solve elastic net regularized multiple linear regression. This method is faster than the cyclical coordinate descent method in many cases and is recommended. | The value 4 and value 6 are supported only when VARIABLE_SELECTION is 0. |
| CATEGORICAL_VARIABLE | VARCHAR | No default value | Indicates by the column name whether or not a column data is actually corresponding to a category variable even the data type of this column is INTEGER. By default, `VARCHAR` or `NVARCHAR` is category variable, and `INTEGER` or `DOUBLE` is continuous variable. | The data type of the column specified by this parameter must be INTEGER. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| VARIABLE_SELECTION | INTEGER | 0 | • 0: All variables are included<br>• 1: Forward selection<br>• 2: Backward selection<br>• 3: Stepwise selection | The value 1 or value 2 or value 3 are supported only when ALG is not 4 or 6 and NO_INTERCEPT is 0. |
| MANDATORY_FEATURE | VARCHAR | No default value | Specifies the column name that needs to be included in the final training model when executing the variable selection. This parameter can be specified multiple times, each time with one column name as feature.<br>**Note**<br>This parameter is a hint. There are exceptional cases that the specified mandatory feature is excluded in the final model, for instance, some mandatory features can be represented as a linear combination of other features, among which some are also mandatory features. | Only valid when VARIABLE_SELECTION is not 0. |
| NO_INTERCEPT | INTEGER | 0 | Specifies whether the intercept term should be ignored in the model.<br>• 0: Does not ignore the intercept term<br>• 1: Ignores the intercept term | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| ALPHA_TO_ENTER | DOUBLE | 0.05 or 0.15 | P-value for forward selection and stepwise selection. | When VARIABLE_SELECTION is 1, default value is 0.05; when VARIABLE_SELECTION is 3, default value is 0.15. |
| ALPHA_TO_REMOVE | DOUBLE | 0.1 or 0.15 | P-value for backward selection and stepwise selection. | When VARIABLE_SELECTION is 2, default value is 0.1; when VARIABLE_SELECTION is 3, default value is 0.15. |
| ENET_LAMBDA | DOUBLE | No default value | Penalized weight. The value should be equal to or greater than 0. | Only valid when ALG is 4 or 6. |
| ENET_ALPHA | DOUBLE | 1 | The elastic net mixing parameter. The value range is between 0 and 1 inclusively.<br>• 0: Ridge penalty<br>• 1: LASSO penalty | Only valid when ALG is 4 or 6. |
| HANDLE_MISSING | INTEGER | 1 | Whether to handle missing value:<br>• 0: No<br>• 1: Yes | |
| MAX_ITERATION | INTEGER | 1.00E+05 | Maximum number of passes over training data. | Only valid when ALG is 4 or 6. |
| THRESHOLD | DOUBLE | 1.00E-05 | Convergence threshold for coordinate descent. | Only valid when ALG is 4 or 6. |
| PHO | DOUBLE | 1.8 | Step size for ADMM. Generally, the value should be greater than 1. | Only valid when ALG is 6. |
| STAT_INF | INTEGER | 0 | Specifies whether to output t-value and Pr(>\|t\|) of coefficients in the result table or not.<br>• 0: No<br>• 1: Yes | |
| ADJUSTED_R2 | INTEGER | 0 | Specifies whether to output adjusted R square or not.<br>• 0: No<br>• 1: Yes | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| DW_TEST | INTEGER | 0 | Specifies whether to do Durbin-Watson test under the null hypothesis that the errors do not follow a first order autoregressive process.<br>• 0: No<br>• 1: Yes | Not available if elastic net regularization is enabled or NO_INTERCEPT = 1. |
| RESET_TEST | INTEGER | 1 | Specifies the order of Ramsey RESET test. Choosing 1 means this test will not be conducted. If you specify an INTEGER larger than 1, then the MLR function will run Ramsey RESET test with power of variables ranging from 2 to the value you specified. | Not available if elastic net regularization is enabled or NO_INTERCEPT = 1. |
| BP_TEST | INTEGER | 0 | Specifies whether or not to do Breusch-Pagan test under the null hypothesis that homoscedasticity is satisfied.<br>• 0: No<br>• 1: Yes | Not available if elastic net regularization is enabled or NO_INTERCEPT = 1. |
| KS_TEST | INTEGER | 0 | Specifies whether or not to do Kolmogorov-Smirnov normality test under the null hypothesis that if errors of MLR follow a normal distribution.<br>• 0: No<br>• 1: Yes | Not available if elastic net regularization is enabled or NO_INTERCEPT = 1. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| PMML_EXPORT | INTEGER | 0 | • 0: Does not export multiple linear regression model in PMML.<br>• 2: Exports multiple linear regression model in PMML. The maximum length of each row is 5000 characters. | |
| JSON_EXPORT | INTEGER | 0 | • 0: Does not export multiple linear regression model in JSON.<br>• 1: Exports multiple linear regression model in JSON.<br>**Note**<br>Currently either PMML or JSON format model can be exported. JSON format is preferred if both formats are to be exported. | |
| HAS_ID | INTEGER | 1 | Specifies whether the input data table has an ID column. If the ID column exists, the ID column must be the first column.<br>• 0: No ID column<br>• 1: Has an ID column | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| SELECTED_FEATURES | VARCHAR | No default value | A string to specify the features that will be processed. The pattern is "X1,...,Xn", where Xi is the corresponding column name in the data table. If this parameter is not specified, all the features will be processed, and all columns except the first and second column data will be processed as independent variables. | |
| DEPENDENT_VARIABLE | VARCHAR | No default value | Column name in the data table used as dependent variable. If this parameter is not specified, the second column data will be processed as dependent variables. | |

Model Evaluation and Parameter Selection

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| RESAMPLING_METHOD | NVARCHAR | No default value | Specifies the resampling method for model evaluation or parameter selection.<br>• cv<br>• bootstrap<br>• cv_sha<br>• bootstrap_sha<br>• cv_hyperband<br>• bootstrap_hyperband<br>If no value is specified for this parameter, neither model evaluation nor parameter selection is activated. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| RESOURCE | NVARCHAR | No default value | Resource type used in successive-halving and hyperband algorithm for parameter selection.<br>• MAX_ITERATION | Mandatory and valid only when RESAMPLING_METHOD parameter is cv_sha, cv_hyperband, bootstrap_sha, or bootstrap_hyperband. |
| MAX_RESOURCE | INTEGER | No default value | Maximum resource budget for single hyper-parameter candidate.<br>Valid value must be larger than 0. | Mandatory and valid only when RESOURCE parameter is set. |
| MIN_RESOURCE_RATE | DOUBLE | 0 | Minimum required resource budget comparing to maximum resource for single hyper-parameter candidate.<br>Valid value should be within [0, 1). | Valid only when RESOURCE parameter is set. |
| REDUCTION_RATE | DOUBLE | 3 | This parameter specifies the reduction rate of available size of hyper-parameter candidates. For each round, the available parameter candidate size will be divided by value of this parameter. Thus, valid value for this parameter must be larger than 1.0. | Valid only when RESOURCE parameter is set. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| AGGRESSIVE_ELIMINATION | INTEGER | 0 | Boolean value indicates whether aggressive elimination will be performed for successive-halving algorithm. When set to 1, it will eliminate more parameter candidates than expected elimination defined by REDUCTION_RATE parameter; This can enhance the run-time performance but may result in sub-optimal hyper-parameter candidate.<br>• 0: Does not perform aggressive elimination<br>• 1: Performs aggressive elimination | Valid only when RESAMPLING_METHOD parameter is cv_sha or bootstrap_sha. |
| PS_VERBOSE | INTEGER | 1 | Boolean value indicates whether optimal hyper-parameter and all evaluation statistics of related hyper-parameter candidates will be output in Statistics output table.<br>• 0: Does not output evaluation statistics<br>• 1: Outputs evaluation statistics | Valid only when RESAMPLING_METHOD is set. |
| EVALUATION_METRIC | NVARCHAR | No default value | Specifies the evaluation metric for model evaluation or parameter selection.<br>• RMSE | |
| FOLD_NUM | INTEGER | No default value | Specifies the fold number for a cross validation method. | Mandatory and valid only when RESAMPLING_METHOD is set to cv, cv_sha, or cv_hyperband. |
| REPEAT_TIMES | INTEGER | 1 | Specifies the number of repeat times for resampling. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| PARAM_SEARCH_STRATEGY | NVARCHAR | No default value | Specifies the method to activate parameter selection.<br>• grid<br>• random | This parameter is only valid when RESAMPLING_METHOD is not cv_hyperband nor bootstrap_hyperband; Otherwise, this parameter is always considered to have value random. |
| RANDOM_SEARCH_TIMES | INTEGER | No default value | Specifies the number of times to randomly select candidate parameters for selection. | Mandatory and valid when PARAM_SEARCH_STRATEGY is set to random. |
| SEED | INTEGER | 0 | Specifies the seed for random generation. Use system time when 0 is specified. | |
| TIMEOUT | INTEGER | 0 | Specifies maximum running time for model evaluation or parameter selection, in seconds. No timeout when 0 is specified. | |
| PROGRESS_INDICATOR_ID | NVARCHAR | No default value | Sets an ID of progress indicator for model evaluation or parameter selection. No progress indicator is active if no value is provided. | |
| ENET_LAMBDA_VALUES | NVARCHAR | No default value | Specifies values of ENET_LAMBDA to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| ENET_LAMBDA_RANGE | NVARCHAR | No default value | Specifies the range of ENET_LAMBDA to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| ENET_ALPHA_VALUES | NVARCHAR | No default value | Specifies values of ENET_ALPHA to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| ENET_ALPHA_RANGE | NVARCHAR | No default value | Specifies the range of parameter ENET_ALPHA to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| PRECOMPUTE_LMS_SKETCH | INTEGER | 1 | • 0: Does not perform LMS sketch<br>• 1: Performs LMS sketch | LMS sketch will only perform when RESAMPLING_METHOD is set and Data Row Number is larger than Data Feature Number. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| STABLE_SKETCH_ALG | INTEGER | 1 | When computing LMS sketch, there are two algorithms to choose. One algorithm is more numerical stable than the other one, but is slower.<br>• 0: Does not use stable algorithm<br>• 1: Uses stable algorithm | Only valid when LMS sketch is performed and SPARSE_SKETCH_ALG is 0. |
| SPARSE_SKETCH_ALG | INTEGER | 0 | This is a specific LMS sketch algorithm to cope with sparse data.<br>• 0: Does not use sparse LMS sketch algorithm<br>• 1: Uses sparse LMS sketch algorithm<br>**Note**<br>Use this option with caution. If the provided data are not sparse, turning on this option will cause LMS sketch to be extremely slow! | Only valid when LMS sketch is performed. |

### Output Table

### Coefficient Table

| Column | Data Type | Column Name | Description |
|---|---|---|---|
| 1st column | NVARCHAR(1000) | VARIABLE_NAME | Name of variable |
| 2nd column | DOUBLE | COEFFICIENT_VALUE | Value of the coefficients |
| 3rd column | DOUBLE | T_VALUE | T-value of coefficients.<br>**Note**<br>This column contains NULL value when the STAT_INF parameter is set to 0. |

| Column | Data Type | Column Name | Description |
|---|---|---|---|
| 4th column | DOUBLE | P_VALUE | Pr(>\|t\|) of coefficients.<br>**Note**<br>This column contains NULL value when the STAT_INF parameter is set to 0. |

### Semistructured_Result Table

| Column | Data Type | Column Name | Description |
|---|---|---|---|
| 1st column | INTEGER | ROW_INDEX | Row index |
| 2nd column | NVARCHAR(5000) | MODEL_CONTENT | Linear regression model in PMML or JSON format |

### Fitted Table

| Column | Data Type | Column Name | Description |
|---|---|---|---|
| 1st column | ID (in input table) data type | ID (in input table) column name | "ID" |
| 2nd column | DOUBLE | VALUE | Fitted value $Y_i$ |

### Statistics Table

| Column | Data Type | Column Name | Description |
|---|---|---|---|
| 1st column | NVARCHAR(256) | STAT_NAME | Name |
| 2nd column | NVARCHAR(1000) | STAT_VALUE | Value |

### Optimal_Param Table

| Column | Data Type | Column Name | Description |
|---|---|---|---|
| 1st column | NVARCHAR(256) | PARAM_NAME | |
| 2nd column | INTEGER | INT_VALUE | |
| 3rd column | DOUBLE | DOUBLE_VALUE | |
| 4th column | NVARCHAR(1000) | STRING_VALUE | |

### General Guides

This fitting procedure supports several linear models and hyper-parameter selection for regularization related parameters.

This section will only discuss about the most general part for the usage of this procedure and concern only ordinary least square estimates.

Among tens of configuration parameters, some parameters are general and are used to specify input and control output.

**Parameters for Specifying Input**

- HAS_ID: Specifies whether the first column of input data is an ID column.
- SELECTED_FEATURES: Specifies the columns enumerating all feature columns; Otherwise by default all provided columns except for ID and DEPENDENT column will be considered as feature columns.
- DEPENDENT_VARIABLE: Specifies the dependent column; Otherwise by default the column comes after the optional ID column will be considered as DEPENDENT column.

**Parameters for Controlling Output Model Format**

Now, the fitting procedure support three model format output:

- Key-value pair format
- PMML format
- JSON format

Among them, the key-value pair format model will always be output. For PMML format and JSON format, users can only choose at most one format. The `PMML_EXPORT` parameter is the switch to control PMML format output, and `JSON_EXPORT` controls JSON format output. If both switches are on, current implementation will output JSON format.

**Parameters for Controlling Statistics Output**

This procedure will output some statistics to measure the output model. Some statistics will always be output while others are optional and has switch to control its output.

Following statistics will be unconditionally output:

- R2
- F
- AIC
- BIC
- MSE
- RMSE
- MAE
- MAPE
- EXPLAINED_VARIANCE
- MAX_ERROR

Following parameters are switch to control the output of corresponding statistics:

- STAT_INF
- ADJUSTED_R2
- DW_TEST
- RESET_TEST
- BP_TEST
- KS_TEST

In above parameters, all except `RESET_TEST` take value 0 and 1. When setting the parameter to value 1, the corresponding statistic will be output in STATISTIC output table. For `RESET_TEST` parameter, it takes value

larger than 0. When it takes value 1, the Ramsey RESET test will not be output; For other values the test will be run and corresponding result will be output, refer to the definition of this parameter for detail.

The `STAT_INF` parameter needs a little more explanation. The statistics will be output at the key-value format model, one statistic per coefficient. It will output t-test statistic and its p-value with the null hypothesis that corresponding coefficient value is 0. Thus, the smaller value of p-value statistic, the more significant of the test result to reject the null hypothesis.

The fitting function has following preprocessing steps:

- Missing value handling
- Categorical variable handling

Current fitting function allows users to input missing value (input as NULL value) for all columns in data table, however, the imputation strategy for all columns are preconfigured and do no support customization. The function will first compute an imputed value for all columns (if needed) and then impute that value for all NULL values for that column. The current preconfigured imputation strategies for all types of columns are:

- ID Column: Does not impute missing value
- Dependent Column: Deletes the entire row
- Continuous Feature: Uses the mean value of all non-missing value in that column
- Categorical Feature: Uses the mode value of all non-missing value in that column

Since `Multiple Linear Regression` algorithm only supports numerical value, thus for categorical value, the fitting function will use dummy encoding (one variant of one-hot encoding) to turn one categorical feature into multiple numerical features.

**Parameters for Specifying Preprocessing Behaviour**

- HANDLE_MISSING
- CATEGORICAL_VARIABLE

The `HANDLE_MISSING` parameter controls the output of missing value handling related information in output model. Currently PMML format model does not support output missing value handling information, while two other formats support this parameter.

The `CATEGORICAL_VARIABLE` parameter specifies INTEGER type categorical feature, one feature per time. VARIABLE type features are automatically treated as categorical feature, thus there is no needs to specify them with this parameter. INTEGER type features will be treated as continuous feature by default if they are not specified by this parameter.

The most basic usage of this procedure is to obtain an ordinary least square estimates without any regularization.

**Parameters for Ordinary Least Square Estimates**

- ALG
  Current implementation provides three algorithms to solve the model based on ordinary least square: QR Decomposition, SVD, Cholesky Decomposition. Among them, SVD algorithm is most stable but slower comparing to other two algorithms. QR Decomposition is faster than SVD, but less numerical stable when input data is rank-deficient.
- NO_INTERCEPT
  This parameter controls whether the intercept item in linear model exists.

**Examples**

### Example: Fitting Linear Model with Ordinary Least Square Only

```
SET SCHEMA DM_PAL;
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE
    #PAL_PARAMETER_TBL
    ("PARAM_NAME" VARCHAR(256), "INT_VALUE" INTEGER, "DOUBLE_VALUE" DOUBLE,
"STRING_VALUE" VARCHAR(1000));
INSERT INTO #PAL_PARAMETER_TBL VALUES ('THREAD_RATIO',NULL,0.5,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('JSON_EXPORT',1,NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('CATEGORICAL_VARIABLE',NULL,NULL,'X3');
DROP TABLE PAL_MLR_DATA_TBL;
CREATE COLUMN TABLE PAL_MLR_DATA_TBL
    ("ID" varchar(50), "Y" DOUBLE, "X1" DOUBLE,
    "X2" varchar(100), "X3" INT
    );
INSERT INTO PAL_MLR_DATA_TBL VALUES (0, -6.879, 0.00, 'A', 1);
INSERT INTO PAL_MLR_DATA_TBL VALUES (1, -3.449, 0.50, 'A', 1);
INSERT INTO PAL_MLR_DATA_TBL VALUES (2,  6.635, 0.54, 'B', 1);
INSERT INTO PAL_MLR_DATA_TBL VALUES (3, 11.844, 1.04, 'B', 1);
INSERT INTO PAL_MLR_DATA_TBL VALUES (4,  2.786, 1.50, 'A', 1);
INSERT INTO PAL_MLR_DATA_TBL VALUES (5,  2.389, 0.04, 'B', 2);
INSERT INTO PAL_MLR_DATA_TBL VALUES (6, -0.011, 2.00, 'A', 2);
INSERT INTO PAL_MLR_DATA_TBL VALUES (7,  8.839, 2.04, 'B', 2);
INSERT INTO PAL_MLR_DATA_TBL VALUES (8,  4.689, 1.54, 'B', 1);
INSERT INTO PAL_MLR_DATA_TBL VALUES (9, -5.507, 1.00, 'A', 2);
CALL
_SYS_AFL.PAL_LINEAR_REGRESSION(PAL_MLR_DATA_TBL,"#PAL_PARAMETER_TBL", ?, ?, ?, ?,
?);
```

### Expected Results

### Coefficient

| | VARIABLE_NAME | COEFFICIENT_VALUE | T_VALUE | P_VALUE |
|---|---|---|---|---|
| 1 | __PAL_INTERCEPT__ | -5.166599999999998 | *NULL* | *NULL* |
| 2 | X1 | 3.6304000000000007 | *NULL* | *NULL* |
| 3 | X2__PAL_DELIMIT__A | 0 | *NULL* | *NULL* |
| 4 | X2__PAL_DELIMIT__B | 9.345983999999998 | *NULL* | *NULL* |
| 5 | X3__PAL_DELIMIT__1 | 0 | *NULL* | *NULL* |
| 6 | X3__PAL_DELIMIT__2 | -2.6894999999999993 | *NULL* | *NULL* |
| 7 | {"CONTENT":"{\"impute_model\":{\"column_statistic | *NULL* | *NULL* | *NULL* |

### JSON

| | ROW_INDEX | |
|---|---|---|
| 1 | 1 | {"Metadata":{"ContinuousFeature":["X1"],"CategoricalFeature":["X2","X3"],"CategoricalValue":[["A","B"],["1","2"]]},"Coefficient":{"InterceptCoefficient":-5.16 |

### Fitted

| | ID | VALUE |
|---|---|---|
| 1 | 0 | -5.166599999999998 |
| 2 | 1 | -3.3513999999999977 |
| 3 | 2 | 6.1398 |
| 4 | 3 | 7.955 |
| 5 | 4 | 0.2790000000000026 |
| 6 | 5 | 1.6351000000000013 |
| 7 | 6 | -0.5952999999999959 |
| 8 | 7 | 8.895900000000003 |
| 9 | 8 | 9.7702 |
| 10 | 9 | -4.225699999999996 |

### Statistics

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | R2 | 0.8450760805642397 |
| 2 | F | 10.909562366380138 |
| 3 | AIC | 55.05013252618991 |
| 4 | BIC | 56.56305799116014 |
| 5 | MSE | 8.828294166666666 |
| 6 | RMSE | 2.9712445484454264 |
| 7 | MAE | 1.64588 |
| 8 | MAPE | 563.3657308374987 |
| 9 | F_PVALUE | 0.007645757368786004 |
| 10 | EXPLAINED_VARIANCE | 0.8450760805642397 |
| 11 | MAX_ERROR | 5.081200000000001 |

## Biased Model Guides

Refer to Biased Model [page 497] guide of this procedure.

## Model Evaluation and Parameter Selection Guides

Refer to Model Evaluation and Parameter Selection [page 501] guide of this procedure.

## _SYS_AFL.PAL_LINEAR_REGRESSION_PREDICT

This function performs prediction with the given linear regression model.

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | DATA table | IN |
| 2 | COEFFICIENT table | IN |
| 3 | PARAMETER table | IN |
| 4 | FITTED table | OUT |

### Input Tables

### Data Table

| Column | Reference for Output Table | Data Type | Description |
|---|---|---|---|
| 1st column | ID | INTEGER, VARCHAR, or NVARCHAR | ID |
| Feature columns | None | INTEGER, DOUBLE, or DECIMAL(p, s) | Variable $X_n$ |

### Coefficient Table

| Column | Reference for Output Table | Data Type | Description |
|---|---|---|---|
| 1st column | None | INTEGER, VARCHAR, or NVARCHAR | Variable name or ROW_INDEX for the PMML or JSON model |
| 2nd column | None | INTEGER, DOUBLE, VARCHAR, NVARCHAR, or CLOB | Values of the coefficients or the content of PMML or JSON model |

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| THREAD_RATIO | DOUBLE | 1.0 | Specifies the ratio of total number of threads that can be used by this function. The value range is from 0 to 1, where 0 means only using 1 thread, and 1 means using at most all the currently available threads. Values outside the range will be ignored and this function heuristically determines the number of threads to be used. |

### Output Table

### Fitted Table

| Column | Data Type | Column Name | Description |
|---|---|---|---|
| 1st column | ID (in input table) data type | ID (in input table) column name | ID |
| 2nd column | DOUBLE | VALUE | Value $Y_i$ |

### General Guides

This procedure is used to predict target value with provided data instance and a previously trained linear model.

The feature name and its type should precisely match corresponding feature in training data that is used to obtain current trained model, otherwise unexpected results will happen.

The input model table supports three formats:

- Key-value format
- PMML format
- JSON format

Current implementation will use the field type and field contents of the coefficient input table to automatically judge the model content.

In fitted output table, each predict data instance is marked by its ID, together the predicted target value will be output.

**Example**

```
SET schema DM_PAL;
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE
    #PAL_PARAMETER_TBL
    ("PARAM_NAME" VARCHAR(256), "INT_VALUE" INTEGER, "DOUBLE_VALUE" DOUBLE,
"STRING_VALUE" VARCHAR(1000));
INSERT INTO #PAL_PARAMETER_TBL VALUES ('THREAD_RATIO',NULL,0.1,NULL);
DROP TABLE PAL_FMLR_PREDICTDATA_TBL;
CREATE COLUMN TABLE PAL_FMLR_PREDICTDATA_TBL ( "ID" INT, "X1" DOUBLE, "X2"
VARCHAR(100), "X3" INT);
INSERT INTO PAL_FMLR_PREDICTDATA_TBL VALUES (0, 1.690, 'B', 1);
INSERT INTO PAL_FMLR_PREDICTDATA_TBL VALUES (1, 0.054, 'B', 2);
INSERT INTO PAL_FMLR_PREDICTDATA_TBL VALUES (2, 0.123, 'A', 2);
INSERT INTO PAL_FMLR_PREDICTDATA_TBL VALUES (3, 1.980, 'A', 1);
INSERT INTO PAL_FMLR_PREDICTDATA_TBL VALUES (4, 0.563, 'A', 1);
DROP TABLE PAL_FMLR_COEFICIENT_TBL;
CREATE COLUMN TABLE PAL_FMLR_COEFICIENT_TBL ("Coefficient" varchar(50),
"CoefficientValue" DOUBLE);
INSERT INTO PAL_FMLR_COEFICIENT_TBL VALUES ('__PAL_INTERCEPT__',
-5.7044999999999995);
INSERT INTO PAL_FMLR_COEFICIENT_TBL VALUES ('X1', 3.092499999999999);
INSERT INTO PAL_FMLR_COEFICIENT_TBL VALUES ('X2__PAL_DELIMIT__A', 0);
INSERT INTO PAL_FMLR_COEFICIENT_TBL VALUES ('X2__PAL_DELIMIT__B',
9.367500000000001);
INSERT INTO PAL_FMLR_COEFICIENT_TBL VALUES ('X3__PAL_DELIMIT__1', 0);
INSERT INTO PAL_FMLR_COEFICIENT_TBL VALUES ('X3__PAL_DELIMIT__2',
-2.6894999999999984);
CALL _SYS_AFL.PAL_LINEAR_REGRESSION_PREDICT(PAL_FMLR_PREDICTDATA_TBL,
PAL_FMLR_COEFICIENT_TBL, "#PAL_PARAMETER_TBL", ?);
```

**Expected Result**

| | ID | VALUE |
|---|---|---|
| 1 | 0 | 8.889325 |
| 2 | 1 | 1.1404950000000036 |
| 3 | 2 | -8.013622499999999 |
| 4 | 3 | 0.41864999999999863 |
| 5 | 4 | -3.9634225 |

## 5.3.6.1 Biased Model

The pure least square estimates have issues:

- Prediction Accuracy: least square estimates often have low bias but high variance. Prediction accuracy can be improved by shrinking or setting some coefficients to zero.
- Interpretation: with a large number of features, users often want to determine a much smaller subset that exhibits the strongest effects.

Current implementation supports both biased models.

For the first case, current implementation support Ridge Penalty form and Lasso Penalty form and their mixed form elastic net.

For the second case, current implementation support variable selection with three options: forward selection, backward selection and stepwise selection.

## Elastic Net Regularization

Elastic net regularization for multiple linear regression seeks to find $\begin{bmatrix} \beta_0 \\ \boldsymbol{\beta} \end{bmatrix}$ that minimizes:

$$\min_{\binom{\beta_0}{\boldsymbol{\beta}} \in \mathbb{R}^{n+1}} \frac{1}{2m} \sum_{i=1}^{m} \left(y_i - \beta_0 - \mathbf{x}_i^T \beta_i\right)^2 + \lambda \cdot \mathbf{P}_\alpha(\boldsymbol{\beta})$$

$$\mathbf{P}_\alpha(\boldsymbol{\beta}) = (1-\alpha) \cdot \frac{1}{2} \|\boldsymbol{\beta}\|_2^2 + \alpha \cdot \|\boldsymbol{\beta}\|_1 .$$

Where

**Parameters to Control Elastic Net Regularization**

- ENET_LAMBDA
- ENET_ALPHA

These two parameters represent the λ and α in above math formula. From the math formula, it shows that elastic net is a mixed form of Ridge Regularization and Lasso Regularization. The `ENET_ALPHA` represents the percentage of Lasso regularization in the elastic net. When its value is 1, it represents a pure Lasso Regularization. When it is 0, it represents pure Ridge Regularization.

The Lasso regularization can achieve a sparse model effect, which functions similar to variable selection below. However, the two models are different. In variable selection each feature is given two options, either selected or not. While in Lasso regularization the coefficient for each feature will change continuously with λ.

Currently there are two methods to solve elastic net model. Users can use the ALG (4 or 6) configuration parameter to select one. Both algorithms are iteration-based so they both have one parameter `MAX_ITERATION` to control the maximum allowed iteration steps. If this parameter was set too small, the final obtained model may not be theoretically optimal since the method may not converge.

**Parameters to Control Behaviour of Cyclical Coordinate Descent Method**

- MAX_ITERATION
- THRESHOLD

The `THRESHOLD` parameter controls the accuracy of obtained model, the smaller this parameter is, the more accurate of the obtained model, but also means the more iteration steps are needed.

**Parameters to Control Behaviour of Alternating Direction Method of Multipliers Method**

- MAX_ITERATION
- PHO

The `PHO` parameter specifies the penalty parameter for ADMM method.

**Example for Regularization**

### Example: Fitting Multiple Linear Regression Model with Elastic Net Penalties

```
SET SCHEMA DM_PAL;
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE
    #PAL_PARAMETER_TBL
    ("PARAM_NAME" VARCHAR(256), "INT_VALUE" INTEGER, "DOUBLE_VALUE" DOUBLE,
"STRING_VALUE" VARCHAR(1000));
INSERT INTO #PAL_PARAMETER_TBL VALUES ('ALG', 6, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('ENET_LAMBDA', NULL, 0.003194, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('ENET_ALPHA', NULL, 0.95, NULL);
DROP TABLE PAL_ENET_MLR_DATA_TBL;
CREATE COLUMN TABLE PAL_ENET_MLR_DATA_TBL ( "ID" INT,"Y" DOUBLE,"V1" DOUBLE,"V2"
DOUBLE,"V3" DOUBLE);
INSERT INTO PAL_ENET_MLR_DATA_TBL VALUES (0, 1.2, 0.1, 0.205, 0.9);
INSERT INTO PAL_ENET_MLR_DATA_TBL VALUES (1, 0.2, -1.705, -3.4, 1.7);
INSERT INTO PAL_ENET_MLR_DATA_TBL VALUES (2, 1.1, 0.4, 0.8, 0.5);
INSERT INTO PAL_ENET_MLR_DATA_TBL VALUES (3, 1.1, 0.1, 0.201, 0.8);
INSERT INTO PAL_ENET_MLR_DATA_TBL VALUES (4, 0.3, -0.306, -0.6, 0.2);
CALL
_SYS_AFL.PAL_LINEAR_REGRESSION(PAL_ENET_MLR_DATA_TBL,"#PAL_PARAMETER_TBL", ?, ?,
?,?,? );
```

### Expected Results

### Coefficient

| | VARIABLE_NAME | COEFFICIENT_VALUE | T_VALUE | P_VALUE |
|---|---|---|---|---|
| 1 | __PAL_INTERCEPT__ | 0.42185781023319235 | *NULL* | *NULL* |
| 2 | V1 | 0.4267152562543356 | *NULL* | *NULL* |
| 3 | V2 | 0.21276196632197145 | *NULL* | *NULL* |
| 4 | V3 | 0.728600758368901 | *NULL* | *NULL* |
| 5 | {"CONTENT":"{\"impute_model\":{\"column_statis | *NULL* | *NULL* | *NULL* |

### Fitted

| | ID | VALUE |
|---|---|---|
| 1 | 0 | 1.1638862214866408 |
| 2 | 1 | 0.20953890205197911 |
| 3 | 2 | 1.1270538649769541 |
| 4 | 3 | 1.090175097784463 |
| 5 | 4 | 0.309345913699963 |

### Statistics

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | R2 | 0.997562255203067 |
| 2 | F | 131.39648197843866 |
| 3 | AIC | -14.208217766226475 |
| 4 | BIC | -16.161028204055974 |
| 5 | MSE | 0.0023109820674924947 |
| 6 | RMSE | 0.04807267485269043 |
| 7 | MAE | 0.01837547229155846 |
| 8 | MAPE | 2.849370466948232 |
| 9 | EXPLAINED_VARIANCE | 0.997562255203067 |
| 10 | MAX_ERROR | 0.036113778513359174 |

## Variable Selection Model

Normally a linear model may contain thousand features and users usually want to know only the most effective features. For this scenario, Variable Selection method can generate such sparse model.

Current implementation supports three methods of variable selection:

- Forward selection
- Backward selection
- Stepwise selection

Forward selection initially assumes an empty model, only intercept is included. For each iteration it includes the most promising candidate if this candidate is significant than preconfigured threshold.

On the contrary, backward selection initially includes all features into the model. For each iteration, it tries to remove the least promising candidate if this candidate is less significant than preconfigured threshold.

Finally, stepwise selection mixed forward selection and backward selection. For each iteration it tries to include one candidate like forward selection, then removes all least promising candidates like backward selection.

Current implementation uses F-test to compare two models with the only difference the considered candidate included or not and use the p-value of F-test to compare with preconfigured threshold to make decision.

Variable selection is a greedy algorithm, so it may not give the best subset selection with the same size of obtained model.

**Parameters to Control Variable Selection Model**

- VARIABLE_SELECTION
- MANDATORY_FEATURE
- ALPHA_TO_ENTER
- ALPHA_TO_REMOVE

Sometimes users may want some features to be always included in final model, the `MANDATORY_FEATURE` parameter is introduced for this purpose. Users can specify multiple times for this parameter, one per mandatory feature.

Parameter `ALPHA_TO_ENTER` is the threshold significance bound for p-value of F-test in forward selection. Parameter `ALPHA_TO_REMOVE` is the threshold significance bound for p-value of F-test in backward selection. For both threshold parameters, the smaller they are, the more sparse model user will obtain.

**Example for Variable Selection**

**Example: Fitting Multiple Linear Regression Model with Forward Selection**

```
SET SCHEMA DM_PAL;
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE
    #PAL_PARAMETER_TBL
    ("PARAM_NAME" VARCHAR(256), "INT_VALUE" INTEGER, "DOUBLE_VALUE" DOUBLE,
"STRING_VALUE" VARCHAR(1000));
INSERT INTO #PAL_PARAMETER_TBL VALUES ('VARIABLE_SELECTION', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('ALPHA_TO_ENTER', NULL, 0.1, NULL);
DROP TABLE PAL_ENET_MLR_DATA_TBL;
CREATE COLUMN TABLE PAL_ENET_MLR_DATA_TBL ( "ID" INT,"Y" DOUBLE,"V1" DOUBLE,"V2"
DOUBLE,"V3" DOUBLE);
INSERT INTO PAL_ENET_MLR_DATA_TBL VALUES (0, 1.2, 0.1, 0.205, 0.9);
INSERT INTO PAL_ENET_MLR_DATA_TBL VALUES (1, 0.2, -1.705, -3.4, 1.7);
INSERT INTO PAL_ENET_MLR_DATA_TBL VALUES (2, 1.1, 0.4, 0.8, 0.5);
INSERT INTO PAL_ENET_MLR_DATA_TBL VALUES (3, 1.1, 0.1, 0.201, 0.8);
INSERT INTO PAL_ENET_MLR_DATA_TBL VALUES (4, 0.3, -0.306, -0.6, 0.2);
CALL
_SYS_AFL.PAL_LINEAR_REGRESSION(PAL_ENET_MLR_DATA_TBL,"#PAL_PARAMETER_TBL", ?, ?,
?,?,? );
```

### Expected Results

### Coefficient

| | VARIABLE_NAME | COEFFICIENT_VALUE | T_VALUE | P_VALUE |
|---|---|---|---|---|
| 1 | __PAL_INTERCEPT__ | 0.4057457491098929 | *NULL* | *NULL* |
| 2 | V1 | 0.8699415278550149 | *NULL* | *NULL* |
| 3 | V2 | 0 | *NULL* | *NULL* |
| 4 | V3 | 0.7557948171351128 | *NULL* | *NULL* |
| 5 | {"CONTENT":"{\"impute_model\":{\"colu | *NULL* | *NULL* | *NULL* |

### Fitted

| | ID | VALUE |
|---|---|---|
| 1 | 0 | 1.1729552373169958 |
| 2 | 1 | 0.20734663324678415 |
| 3 | 2 | 1.1316197688194554 |
| 4 | 3 | 1.0973757556034847 |
| 5 | 4 | 0.29070260501328093 |

### Statistics

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | R2 | 0.9980184280579849 |
| 2 | F | 503.64985842658854 |
| 3 | AIC | -17.24413230884153 |
| 4 | BIC | -18.806380659105127 |
| 5 | MSE | 0.0009392651005151377 |
| 6 | RMSE | 0.030647432201004014 |
| 7 | MAE | 0.01558656082649561 |
| 8 | MAPE | 2.427854123951645 |
| 9 | F_PVALUE | 0.00198157194201506 |
| 10 | EXPLAINED_VARIANCE | 0.9980184280579849 |
| 11 | MAX_ERROR | 0.031619768819455274 |

## 5.3.6.2 Model Evaluation and Parameter Selection

For elastic-net model, it usually requires user's prior knowledge to set configuration parameter `ENET_LAMBDA` and `ENET_ALPHA`.

However, users normally may not have such prior knowledge so they have to test various values to guess. To alleviate the continual try that is time-consuming and tedious, current implementation provides model evaluation and parameter selection function.

With parameter selection function, instead of providing a single value for `ENET_LAMBDA` and `ENET_ALPHA`, users can provide a set of candidates and use stored procedure to select the best one based on model evaluation function.

The parameter selection process is time-consuming and linearly proportional to the candidate space size, in order to accelerate this process, current function support two ways to boost runtime performance:

- preprocess the training data by compressing them with LMS sketch to boost the training runtime performance for single candidate.
- early-stop training computation by successive-halving and hyperband algorithm. Note this is a greedy algorithm and may fail to find best candidates.

## LMS Sketch

LMS sketch is a data compression technique. Since this happens at the initial step of data loading, later training step can be boosted tremendously. Current implementation uses QR sketch, which is lossless compression so in principle there is no accuracy loss.

**Parameters to Control LMS Sketch**

- PRECOMPUTE_LMS_SKETCH
- STABLE_SKETCH_ALG
- SPARSE_SKETCH_ALG

`PRECOMPUTE_LMS_SKETCH` is a switch for LMS sketch function. By default it is turned on. The only overhead is extra memory is required if it is turned on.

Parameter `STABLE_SKETCH_ALG` relates to QR decomposition in QR sketch. When this switch is turned on, it uses a more stable algorithm.

Users should be cautious to use the `SPARSE_SKETCH_ALG` parameter since if the data is not sparse, the runtime performance will degrade tremendously when this parameter is set.

## Successive-Halving and Hyperband

Successive-halving and hyperband are two algorithms to boost the parameter selection process.

Successive-halving algorithm is iteration based. With fixed total resource budget `B` and `n` candidates, initially algorithm will uniformly allocate resource `B/n` to each candidate. By evaluating all models trained with the given resource, the algorithm will reduce the candidates by preconfigured reduction rate by evaluation results, then in the next iteration, total resource budget `B` will be allocated uniformly to the survival candidates. This process will continue until only one candidate survives.

The above process is greedy and the final survival may not be optimal since the optimal candidate may falsely be eliminated in the intermediate iteration. In order to increase the chances of getting the optimal candidates, the hyperband algorithm is introduced. With fixed total resource budget `B`, the algorithm will try different `n`. The larger value `n` is, the smaller value of `B/n` is, this is more aggressive for the early-stop. For each value of `n`, hyperband internally calls the successive-halving algorithm to get the optimal candidates. Finally, by comparing the evaluation result for different `n`, hyperband will get its finall optiaml candidates.

Note, both of them are approximate algorithms, so they can't guarantee the best candidate is obtained.

For details about the algorithm, refer to Successive Halving and Hyperband for Model Evaluation and Parameter Selection [page 2240].

**Parameters to Select Early-Stop Algorithm**

- RESAMPLING_METHOD

The early-stop algorithm is specified together with the sampling method. There are two supported sampling methods: cross validation and bootstraping. Each of them can be combined with early-stop algorithm successive-halving or hyperband. Only limitation is that when using hyperband algorithm, the only supported candidate search method is random.

**Parameters to Control Early-Stop**

- RESOURCE
- MAX_RESOURCE
- MIN_RESOURCE_RATE
- REDUCTION_RATE
- AGGRESSIVE_ELIMINATION

Current implementation only supports iteration step number as resource. Both `MAX_RESOURCE` and `MIN_RESOURCE_RATE` are used to constrain the available resource for single candidates. The `MIN_RESOURCE_RATE` represents the minimum required resource divided by maximum allowed resource.

`REDUCTION_RATE` parameter and `AGGRESSIVE_ELIMINATION` parameter control the speed of candidate reductions and resource increase between iteration steps of the whole algorithm. `AGGRESSIVE_ELIMINATION` is normally used when candidate size is too large comparing to relative small maximum allowed resource. In this case, after several iteration steps, each candidate will be allocated with maximum allowed resource and never increases in following iterations if this parameter is turned off. To avoid this case, users can turn on this parameter so the algorithm behaviour will be more aggressive, so as to reduce the candidates in initial steps.

## Example for Successive-Halving and Hyperband

### Example: Parameter Selection with Successive-Halving

```
SET SCHEMA DM_PAL;
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL
    ("PARAM_NAME" VARCHAR(256), "INT_VALUE" INTEGER, "DOUBLE_VALUE" DOUBLE,
"STRING_VALUE" VARCHAR(1000));
INSERT INTO #PAL_PARAMETER_TBL VALUES ('ALG', 6, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('HAS_ID', 1, NULL, NULL);
--parameter selection parameters
INSERT INTO #PAL_PARAMETER_TBL VALUES ('RESAMPLING_METHOD', NULL, NULL,
'cv_sha');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PARAM_SEARCH_STRATEGY', NULL, NULL,
'grid');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('EVALUATION_METRIC', NULL, NULL, 'RMSE');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('REPEAT_TIMES', 2, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FOLD_NUM', 2, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('ENET_LAMBDA_VALUES', NULL, NULL,
'{0.1,0.01,0.05}');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('ENET_ALPHA_VALUES', NULL, NULL,
'{0.5,0.2,0.1}');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('RESOURCE', NULL, NULL, 'MAX_ITERATION');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MAX_RESOURCE', 100, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MIN_RESOURCE_RATE', NULL, 0.05, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('REDUCTION_RATE', NULL, 2.0, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('AGGRESSIVE_ELIMINATION', 0, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PS_VERBOSE', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SEED', 2022, NULL, NULL);
DROP TABLE PAL_ENET_MLR_DATA_TBL;
CREATE COLUMN TABLE PAL_ENET_MLR_DATA_TBL ( "ID" INT,"Y" DOUBLE,"V1" DOUBLE,"V2"
DOUBLE,"V3" DOUBLE);
INSERT INTO PAL_ENET_MLR_DATA_TBL VALUES (0,0.5,0.13,0.13,0.33);
INSERT INTO PAL_ENET_MLR_DATA_TBL VALUES (1,0.15,0.14,2.33,0.34);
INSERT INTO PAL_ENET_MLR_DATA_TBL VALUES (2,0.25,0.15,0.53,0.36);
INSERT INTO PAL_ENET_MLR_DATA_TBL VALUES (3,0.35,0.16,0.33,0.35);
INSERT INTO PAL_ENET_MLR_DATA_TBL VALUES (4,0.45,0.17,3.33,0.37);
INSERT INTO PAL_ENET_MLR_DATA_TBL VALUES (5,0.55,0.18,0.53,0.38);
INSERT INTO PAL_ENET_MLR_DATA_TBL VALUES (6,0.65,0.19,0.33,0.39);
INSERT INTO PAL_ENET_MLR_DATA_TBL VALUES (7,0.75,0.19,0.33,0.31);
```

```
INSERT INTO PAL_ENET_MLR_DATA_TBL VALUES (8,0.85,0.11,0.68,0.32);
INSERT INTO PAL_ENET_MLR_DATA_TBL VALUES (9,0.95,0.12,10.33,0.33);
CALL
_SYS_AFL.PAL_LINEAR_REGRESSION(PAL_ENET_MLR_DATA_TBL,"#PAL_PARAMETER_TBL", ?, ?,
?,?,? );
```

### Expected Results

### Statistics

| | STAT_NAME | |
|---|---|---|
| 1 | timeout | FALSE |
| 2 | TEST_1_RMSE | 0.211884, 0.557461 |
| 3 | TEST_2_RMSE | 0.50738, 0.70951 |
| 4 | EVAL_RESULTS_1 | {"candidates":[{"TEST_RMSE":[[[0.3207136383291418,0.5967099998956490],[0.7884853453676841,0.2733523024651694]],[[0.251211 |
| 5 | EVAL_RESULTS_2 | E":[[[0.3896455597472315,0.7762781555120926],[1.137995031852180,0.2848692243018265]]],"TEST_RMSE.MEAN":"n/a","TEST_RMS |
| 6 | EVAL_RESULTS_3 | 2;ENET_LAMBDA=0.05;"},{"TEST_RMSE":[[[0.3834297289542122,0.7694490573331682],[1.095947486946810,0.2825402018408516]]] |
| 7 | R2 | 0.16256688352667747 |
| 8 | F | 0.18437508629504962 |
| 9 | AIC | 6.50795456659371 |
| 10 | BIC | 7.718294938569894 |
| 11 | MSE | 0.07204915634229406 |
| 12 | RMSE | 0.2684197391070449 |
| 13 | MAE | 0.195270045043907 |
| 14 | MAPE | 57.96114290039313 |
| 15 | EXPLAINED_VARIANCE | 0.1625668835266777 |
| 16 | MAX_ERROR | 0.40850445948569425 |

### Optimal Parameter

| | PARAM_NAME | INT_VALUE | DOUBLE_VALUE | STRING_VALUE |
|---|---|---|---|---|
| 1 | ENET_ALPHA | *NULL* | 0.5 | *NULL* |
| 2 | ENET_LAMBDA | *NULL* | 0.1 | *NULL* |

## Parameter Selection

Parameter selection will first generate a set of candidates. The candidates will be determined with a range or discrete set, plus a search method. Current supported search methods are grid search and random search. Both of them can be used over range or discrete set.

If the user just provides one candidate by providing `ENET_LAMBDA` and `ENET_ALPHA` parameters, then parameter selection reduces to model evaluation.

For each candidate of hyper parameter, parameter selection needs sample the provided data to separate it as training data and validating data. With candidate and training data, it will start a training process to obtain one model, then use the validating data to evaluate the model with the specified evaluation methods. Above process may repeat multiple times and average the evaluation metrics to get a final metric. By comparing the final evaluation metric for each candidate, it will elect the best candidate.

### Parameters to Define Candidate Set

Current algorithm hyper-parameter is a combination of `ENET_LAMBDA` and `ENET_ALPHA`.

The search space can be specified as range or discrete set. The range is defined with a start value and an end value and an optional step value in the middle.

- ENET_LAMBDA_VALUES

- ENET_ALPHA_VALUES
- ENET_LAMBDA_RANGE
- ENET_ALPHA_RANGE

Note: Users can only select one way for both `ENET_LAMBDA` and `ENET_ALPHA`, either both use VALUES suffix parameter or use RANGE suffix.

Following parameter defines the way to decide the actual candidates within the given range or discrete set:

- PARAM_SEARCH_STRATEGY
- RANDOM_SEARCH_TIMES

There are two search ways supported within search space: grid search and random search. Both can be applied to range or discrete set. When grid search applies to discrete set, it equals to cartesian product with all the given set. When it applies to range, it starts from the start value and iterates for each step value specified in RANGES parameter until the end value. Random search could also applies to discrete set and range, it uses parameter `RANDOM_SEARCH_TIMES` to define the final candidate size. Note random search may generate duplicate candidates.

**Parameters to Control Sampling Behaviour**

- RESAMPLING_METHOD
- FOLD_NUM

Current implementation supports two methods: cross validation and bootstrap. Cross validation needs another parameter `FOLD_NUM` to define the fold number.

**Parameters to Define Validation Metrics**

- EVALUATION_METRIC

Currently the only supported metric is `RMSE`.

For each candidate and sampling data set for training and validation set, parameter `REPEAT_TIMES` controls how many times to repeat.

If users need to get deterministic results between different runs with exactly the same result, parameter `SEED` can be used to reproduce the results.

For more details about this topic, refer to topic Model Evaluation and Parameter Selection [page 2227].

## Example for Model Evaluation and Parameter Selection

### Example: Model Evaluation

```
SET SCHEMA DM_PAL;
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE
    #PAL_PARAMETER_TBL
    ("PARAM_NAME" VARCHAR(256), "INT_VALUE" INTEGER, "DOUBLE_VALUE" DOUBLE,
"STRING_VALUE" VARCHAR(1000));
INSERT INTO #PAL_PARAMETER_TBL VALUES ('ALG', 6, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('ENET_LAMBDA', NULL, 0.003194, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('ENET_ALPHA', NULL, 0.95, NULL);
--model evaluation parameters
INSERT INTO #PAL_PARAMETER_TBL VALUES ('RESAMPLING_METHOD', NULL, NULL, 'cv');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('EVALUATION_METRIC', NULL, NULL, 'RMSE');
```

```
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FOLD_NUM', 2, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('REPEAT_TIMES', 2, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SEED', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PROGRESS_INDICATOR_ID', NULL, NULL,
'TEST');
DROP TABLE PAL_ENET_MLR_DATA_TBL;
CREATE COLUMN TABLE PAL_ENET_MLR_DATA_TBL ( "ID" INT,"Y" DOUBLE,"V1" DOUBLE,"V2"
DOUBLE,"V3" DOUBLE);
INSERT INTO PAL_ENET_MLR_DATA_TBL VALUES (0, 1.2, 0.1, 0.205, 0.9);
INSERT INTO PAL_ENET_MLR_DATA_TBL VALUES (1, 0.2, -1.705, -3.4, 1.7);
INSERT INTO PAL_ENET_MLR_DATA_TBL VALUES (2, 1.1, 0.4, 0.8, 0.5);
INSERT INTO PAL_ENET_MLR_DATA_TBL VALUES (3, 1.1, 0.1, 0.201, 0.8);
INSERT INTO PAL_ENET_MLR_DATA_TBL VALUES (4, 0.3, -0.306, -0.6, 0.2);
CALL
_SYS_AFL.PAL_LINEAR_REGRESSION(PAL_ENET_MLR_DATA_TBL,"#PAL_PARAMETER_TBL", ?, ?,
?,?,? );
```

### Expected Results

### Statistics

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | timeout | FALSE |
| 2 | TEST_1_RMSE | 0.789142, 0.403925 |
| 3 | TEST_2_RMSE | 0.640437, 0.129601 |
| 4 | TEST_RMSE.MEAN | 0.490776 |
| 5 | TEST_RMSE.VAR | 0.0831369 |
| 6 | EVAL_RESULTS_1 | {"candidates":[{"TEST_RMSE":[[0.7891422889141987,0.4039259718316649],[0.6404375320248410,0.1296014871415508]],"TEST_RM |
| 7 | R2 | 0.997562255203067 |
| 8 | F | 131.39648197843866 |
| 9 | AIC | -14.208217766226475 |
| 10 | BIC | -16.161028204055974 |
| 11 | MSE | 0.0023109820674924947 |
| 12 | RMSE | 0.04807267485269043 |
| 13 | MAE | 0.01837547229155846 |
| 14 | MAPE | 2.849370466948232 |
| 15 | EXPLAINED_VARIANCE | 0.997562255203067 |
| 16 | MAX_ERROR | 0.036113778513359174 |

### Parameter Selection

```
SET SCHEMA DM_PAL;
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE
    #PAL_PARAMETER_TBL
    ("PARAM_NAME" VARCHAR(256), "INT_VALUE" INTEGER, "DOUBLE_VALUE" DOUBLE,
"STRING_VALUE" VARCHAR(1000));
INSERT INTO #PAL_PARAMETER_TBL VALUES ('ALG', 6, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('HAS_ID', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('ENET_LAMBDA_VALUES', NULL, NULL,
'{0.01,0.001,0.05}');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('ENET_ALPHA_VALUES', NULL, NULL,
'{0.1,0.5,.03}');
--parameter selection parameters
INSERT INTO #PAL_PARAMETER_TBL VALUES ('RESAMPLING_METHOD', NULL, NULL, 'cv');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('EVALUATION_METRIC', NULL, NULL, 'RMSE');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FOLD_NUM', 2, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('REPEAT_TIMES', 2, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SEED', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PROGRESS_INDICATOR_ID', NULL, NULL,
'TEST');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PARAM_SEARCH_STRATEGY', NULL, NULL,
'grid');
DROP TABLE PAL_ENET_MLR_DATA_TBL;
CREATE COLUMN TABLE PAL_ENET_MLR_DATA_TBL ( "ID" INT,"Y" DOUBLE,"V1" DOUBLE,"V2"
DOUBLE,"V3" DOUBLE);
```

```
INSERT INTO PAL_ENET_MLR_DATA_TBL VALUES (0, 1.2, 0.1, 0.205, 0.9);
INSERT INTO PAL_ENET_MLR_DATA_TBL VALUES (1, 0.2, -1.705, -3.4, 1.7);
INSERT INTO PAL_ENET_MLR_DATA_TBL VALUES (2, 1.1, 0.4, 0.8, 0.5);
INSERT INTO PAL_ENET_MLR_DATA_TBL VALUES (3, 1.1, 0.1, 0.201, 0.8);
INSERT INTO PAL_ENET_MLR_DATA_TBL VALUES (4, 0.3, -0.306, -0.6, 0.2);
CALL
_SYS_AFL.PAL_LINEAR_REGRESSION(PAL_ENET_MLR_DATA_TBL,"#PAL_PARAMETER_TBL", ?, ?,
?,?,? );
```

### Expected Results

### Statistics

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | timeout | FALSE |
| 2 | TEST_1_RMSE | 0.798179, 0.0379027 |
| 3 | TEST_2_RMSE | 0.644155, 0.127269 |
| 4 | TEST_RMSE.MEAN | 0.401876 |
| 5 | TEST_RMSE.VAR | 0.141213 |
| 6 | EVAL_RESULTS_1 | {"candidates":[{"TEST_RMSE":[[0.7897891264068078,0.3742107331970970],[0.6403908972675434,0.1201982663352704]],"TEST_RM |
| 7 | EVAL_RESULTS_2 | 768585487459,"TEST_RMSE.VAR":0.1412136124845372,"parameters":"ENET_ALPHA=0.5;ENET_LAMBDA=0.001;"},{"TEST_RMSE":[[0. |
| 8 | R2 | 0.9980714214677238 |
| 9 | F | 171.0599852399282 |
| 10 | AIC | -15.379668409093185 |
| 11 | BIC | -17.332478846922683 |
| 12 | MSE | 0.0018282924485979034 |
| 13 | RMSE | 0.042758536558188044 |
| 14 | MAE | 0.015167009331687631 |
| 15 | MAPE | 2.2065611486073236 |
| 16 | EXPLAINED_VARIANCE | 0.9980714214677238 |

### Optimal Parameter

| | PARAM_NAME | INT_VALUE | DOUBLE_VALUE | STRING_VALUE |
|---|---|---|---|---|
| 1 | ENET_ALPHA | *NULL* | 0.5 | *NULL* |
| 2 | ENET_LAMBDA | *NULL* | 0.001 | *NULL* |

## 5.3.7 Online Linear Regression (Stateless)

Online linear regression is an online version of the linear regression and is used when the training data are obtained multiple rounds.

Additional data are obtained in each round of training. By making use of the current computed linear model and combining with the obtained data in each round, online linear regression adapts the linear model to make the prediction as precise as possible.

See Multiple Linear Regression [page 478] for linear regression definitions.

Before doing the online training, an empty model is initialized via user input parameters. In each round of online training, the intermediate model from the initialization or from the last round of online training is input to the online training algorithm, and in turn the algorithm will output the updated intermediate model after current round. In addition to the intermediate model, the algorithm also output the coefficients of linear model after current round of online training.

Use the obtained linear model with the prediction function in Multiple Linear Regression [page 478] to make the prediction.

# _SYS_AFL.PAL_INIT_ONLINE_LINEAR_REGRESSION

This procedure is used to initialize the linear model.

### Overview of All Tables

| Position | Table Name | Type |
|---|---|---|
| 1 | `<Parameter Table>` | IN |
| 2 | `<Intermediate Result Table>` | OUT |

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

The following parameter is optional. If it is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| ENET_LAMBDA | DOUBLE | 0 | Penalized weight. The value should be equal to or greater than 0. |
| ENET_ALPHA | DOUBLE | 0 | The elastic net mixing parameter. The value range is between 0 and 1 inclusively.<br>• 0: Ridge Penalty<br>• 1: Lasso Penalty |
| MAX_ITERATION | INTEGER | 1000 | Maximum iterative cycle. |
| THRESHOLD | DOUBLE | 1E-5 | Convergence threshold. |

### Output Tables

Intermediate Result Table

| Column | Data Type | Column Name | Description |
|---|---|---|---|
| 1st column | INTEGER | SEQUENCE | Sequence ID |
| 2nd column | NVARCHAR(5000) | INTERMEDIATE_MODEL | Intermediate result after initialization |

### Example

Example: Creating initial model for online linear regression

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_INTERMEDIATE_RESULT_TBL;
CREATE COLUMN TABLE PAL_INTERMEDIATE_RESULT_TBL ("SEQUENCE" INTEGER,
"INTERMEDIATE_MODEL" NVARCHAR(5000));
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL (
    "PARAM_NAME" NVARCHAR(256),
```

```
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR (1000)
);
INSERT INTO PAL_PARAMETER_TBL("PARAM_NAME", "INT_VALUE", "DOUBLE_VALUE",
"STRING_VALUE")
VALUES('ENET_LAMBDA', NULL, 0.1, NULL);
INSERT INTO PAL_PARAMETER_TBL("PARAM_NAME", "INT_VALUE", "DOUBLE_VALUE",
"STRING_VALUE")
VALUES('ENET_ALPHA', NULL, 0.5, NULL);
INSERT INTO PAL_PARAMETER_TBL("PARAM_NAME", "INT_VALUE", "DOUBLE_VALUE",
"STRING_VALUE")
VALUES('MAX_ITERATION', 1200, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL("PARAM_NAME", "INT_VALUE", "DOUBLE_VALUE",
"STRING_VALUE")
VALUES('THRESHOLD', NULL, 1E-6, NULL);
DO BEGIN
lt_para = SELECT * FROM PAL_PARAMETER_TBL;
CALL _SYS_AFL.PAL_INIT_ONLINE_LINEAR_REGRESSION
(:lt_para,lt_intermediate_result);
INSERT INTO PAL_INTERMEDIATE_RESULT_TBL
SELECT * FROM :lt_intermediate_result;
END;
SELECT * FROM PAL_INTERMEDIATE_RESULT_TBL;
```

Expected Result

| | SEQUENCE | INTERMEDIATE_MODEL |
|---|---|---|
| 1 | 0 | {"algorithm":"batch_algorithm","batch_algorithm":{"data":{"xtx":[],"xty":[]},"hyper_parameter":{"max_iteration":1200,"threshold":9 |

## _SYS_AFL.PAL_TRAIN_ONLINE_LINEAR_REGRESSION

This procedure is used to do online training each round.

### Overview of All Tables

| Position | Table Name | Type |
|---|---|---|
| 1 | `<DATA Table>` | IN |
| 2 | `<INTERMEDIATE_RESULT Table>` | IN |
| 3 | `<PARAMETER Table>` | IN |
| 4 | `<COEFFICIENT Table>` | OUT |
| 5 | `<INTERMEDIATE_RESULT_UPDATE Table>` | OUT |

### Input Table

Data Table

| Column | Data Type | Description |
|---|---|---|
| 1st column | INTEGER, VARCHAR, or NVARCHAR | ID |
| 2nd column | INTEGER, DOUBLE, DECIMAL(p, s) | Variable y |

| Column | Data Type | Description |
|---|---|---|
| Feature columns | INTEGER, DOUBLE, DECIMAL(p, s) | Variable Xm |

Intermediate_Result Table

| Column | Data Type | Description |
|---|---|---|
| 1st column | INTEGER | Sequence ID |
| 2nd column | VARCHAR or NVARCHAR | Intermediate Result from last training |

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| DEPENDENT_VARIABLE | VARCHAR or NVARCHAR | No default value | Column name in the data table used as dependent variable. If this parameter is not specified, the second column data will be processed as dependent variables. |
| THREAD_RATIO | DOUBLE | 1.0 | Specifies the ratio of total number of threads that can be used by this function. The value range is from 0 to 1, where 0 means only using 1 thread, and 1 means using at most all the currently available threads. Values outside the range will be ignored and this function heuristically determines the number of threads to use. |
| HAS_ID | INTEGER | 1 | Specifies whether the input data table has an ID column. If it exists, the ID column must be the first column.<br>• 0: No ID column<br>• 1: Has an ID column |

### Output Table

Coefficient Table

| Column | Data Type | Column Name | Description |
|---|---|---|---|
| 1st column | NVARCHAR(1000) | VARIABLE_NAME | Name of variable |

| Column | Data Type | Column Name | Description |
|---|---|---|---|
| 2nd column | DOUBLE | COEFFICIENT_VALUE | Value of coefficients |

INTERMEDIATE_RESULT_UPDATE Table

| Column | Data Type | Column Name | Description |
|---|---|---|---|
| 1st column | INTEGER | SEQUENCE | Sequence ID |
| 2nd column | NVARCHAR(5000) | INTERMEDIATE_MODEL | Intermediate result after current training |

### Example

Example: Fitting online linear regression model

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_LR_DATA_TBL;
CREATE COLUMN TABLE PAL_LR_DATA_TBL (
            "ID" INTEGER,
            "Y"  DOUBLE,
            "X1" DOUBLE,
            "X2" DOUBLE);
DROP TABLE PAL_LRONLINE_MODEL_TBL;
CREATE COLUMN TABLE PAL_LRONLINE_MODEL_TBL (
            "VARIABLE_NAME" NVARCHAR(1000),
            "COEFFICINET_VALUE" DOUBLE);
--the first time
TRUNCATE TABLE PAL_LR_DATA_TBL;
INSERT INTO PAL_LR_DATA_TBL VALUES(1, 130, 7, 26);
INSERT INTO PAL_LR_DATA_TBL VALUES(2, 124, 1, 29);
INSERT INTO PAL_LR_DATA_TBL VALUES(3, 262, 11, 56);
INSERT INTO PAL_LR_DATA_TBL VALUES(4, 162, 11, 31);
TRUNCATE TABLE PAL_PARAMETER_TBL;
INSERT INTO PAL_PARAMETER_TBL("PARAM_NAME", "INT_VALUE", "DOUBLE_VALUE",
"STRING_VALUE")
VALUES('DEPENDENT_VARIABLE', NULL, NULL, 'Y');
INSERT INTO PAL_PARAMETER_TBL("PARAM_NAME", "INT_VALUE", "DOUBLE_VALUE",
"STRING_VALUE")
VALUES('THREAD_RATIO', NULL, 0.5, NULL);
INSERT INTO PAL_PARAMETER_TBL("PARAM_NAME", "INT_VALUE", "DOUBLE_VALUE",
"STRING_VALUE")
VALUES('HAS_ID', 1, NULL, NULL);
TRUNCATE TABLE PAL_LRONLINE_MODEL_TBL;
DO BEGIN
lt_data = SELECT * FROM PAL_LR_DATA_TBL;
lt_intermediate_result = SELECT * FROM PAL_INTERMEDIATE_RESULT_TBL;
lt_para = SELECT * FROM PAL_PARAMETER_TBL;
CALL _SYS_AFL.PAL_TRAIN_ONLINE_LINEAR_REGRESSION
(:lt_data,:lt_intermediate_result,:lt_para,lt_model,lt_intermediate_result_update
);
INSERT INTO PAL_LRONLINE_MODEL_TBL
SELECT * FROM :lt_model;
TRUNCATE TABLE PAL_INTERMEDIATE_RESULT_TBL;
INSERT INTO PAL_INTERMEDIATE_RESULT_TBL
SELECT * FROM :lt_intermediate_result_update;
END;
SELECT * FROM PAL_LRONLINE_MODEL_TBL;
SELECT * FROM PAL_INTERMEDIATE_RESULT_TBL;
--the second time
TRUNCATE TABLE PAL_LR_DATA_TBL;
INSERT INTO PAL_LR_DATA_TBL VALUES(5, 234, 7, 52);
INSERT INTO PAL_LR_DATA_TBL VALUES(6, 258, 11, 55);
INSERT INTO PAL_LR_DATA_TBL VALUES(7, 298, 3, 71);
```

```
INSERT INTO PAL_LR_DATA_TBL VALUES(8, 132, 1, 31);
TRUNCATE TABLE PAL_LRONLINE_MODEL_TBL;
DO BEGIN
lt_data = SELECT * FROM PAL_LR_DATA_TBL;
lt_intermediate_result = SELECT * FROM PAL_INTERMEDIATE_RESULT_TBL;
lt_para = SELECT * FROM PAL_PARAMETER_TBL;
CALL _SYS_AFL.PAL_TRAIN_ONLINE_LINEAR_REGRESSION
(:lt_data,:lt_intermediate_result,:lt_para,lt_model,lt_intermediate_result_update
);
INSERT INTO PAL_LRONLINE_MODEL_TBL
SELECT * FROM :lt_model;
TRUNCATE TABLE PAL_INTERMEDIATE_RESULT_TBL;
INSERT INTO PAL_INTERMEDIATE_RESULT_TBL
SELECT * FROM :lt_intermediate_result_update;
END;
SELECT * FROM PAL_LRONLINE_MODEL_TBL;
SELECT * FROM PAL_INTERMEDIATE_RESULT_TBL;
--the third time
TRUNCATE TABLE PAL_LR_DATA_TBL;
INSERT INTO PAL_LR_DATA_TBL VALUES(9, 227, 2, 54);
INSERT INTO PAL_LR_DATA_TBL VALUES(10, 256, 21, 47);
INSERT INTO PAL_LR_DATA_TBL VALUES(11, 168, 1, 40);
INSERT INTO PAL_LR_DATA_TBL VALUES(12, 302, 11, 66);
INSERT INTO PAL_LR_DATA_TBL VALUES(13, 307, 10, 68);
TRUNCATE TABLE PAL_LRONLINE_MODEL_TBL;
DO BEGIN
lt_data = SELECT * FROM PAL_LR_DATA_TBL;
lt_intermediate_result = SELECT * FROM PAL_INTERMEDIATE_RESULT_TBL;
lt_para = SELECT * FROM PAL_PARAMETER_TBL;
CALL _SYS_AFL.PAL_TRAIN_ONLINE_LINEAR_REGRESSION
(:lt_data,:lt_intermediate_result,:lt_para,lt_model,lt_intermediate_result_update
);
INSERT INTO PAL_LRONLINE_MODEL_TBL
SELECT * FROM :lt_model;
TRUNCATE TABLE PAL_INTERMEDIATE_RESULT_TBL;
INSERT INTO PAL_INTERMEDIATE_RESULT_TBL
SELECT * FROM :lt_intermediate_result_update;
END;
SELECT * FROM PAL_LRONLINE_MODEL_TBL;
SELECT * FROM PAL_INTERMEDIATE_RESULT_TBL;
```

### Expected Results

The first time

COEFFICIENT

| | VARIABLE_NAME | COEFFICINET_VALUE |
|---|---|---|
| 1 | __PAL_INTERCEPT__ | 5.076244691026119 |
| 2 | X1 | 2.9872769514304642 |
| 3 | X2 | 4.000540375227728 |

INTERMEDIATE_RESULT

| | SEQUENCE | |
|---|---|---|
| 1 | 0 | {"algorithm":"batch_algorithm","batch_algorithm":{"data":{"xtx":[4.0,30.0,142.0,30.0,292.0,1168.0,142.0,1168.0,5614.0],"xty":[678 |

The second time

COEFFICIENT

| | VARIABLE_NAME | COEFFICINET_VALUE |
|---|---|---|
| 1 | __PAL_INTERCEPT__ | 5.09444370290214 |
| 2 | X1 | 2.9884192338348505 |
| 3 | X2 | 3.999563230857782 |

INTERMEDIATE_RESULT

| | SEQUENCE | |
|---|---|---|
| 1 | 0 | {"algorithm":"batch_algorithm","batch_algorithm":{"data":{"xtx":[8.0,52.0,351.0,52.0,472.0,2381.0,351.0,2381.0,17345.0],"xty":[16 |

The third time

COEFFICIENT

| | VARIABLE_NAME | COEFFICINET_VALUE |
|---|---|---|
| 1 | __PAL_INTERCEPT__ | 5.073337988966728 |
| 2 | X1 | 2.9941176016537763 |
| 3 | X2 | 3.9993886227220017 |

INTERMEDIATE_RESULT

| | SEQUENCE | |
|---|---|---|
| 1 | 0 | {"algorithm":"batch_algorithm","batch_algorithm":{"data":{"xtx":[13.0,97.0,626.0,97.0,1139.0,4922.0,626.0,4922.0,33050.0],"xty": |

## 5.3.8 Polynomial Regression

Polynomial regression is an approach to modeling the relationship between a scalar variable y and a variable denoted X. In polynomial regression, data is modeled using polynomial functions, and unknown model parameters are estimated from the data. Such models are called polynomial models.

In PAL, the implementation of polynomial regression is to transform to linear regression and solve it:

$y = \beta_0 + \beta_1 \times x + \beta_2 \times x^2 + \ldots + \beta_n \times x^n$

Where $\beta_0 \ldots \beta_n$ are parameters that need to be calculated.

Let $x = x_1$, $x^2 = x_2$, …, $x^n = x_n$, and then

$y = \beta_0 + \beta_1 \times x_1 + \beta_2 \times x_2 + \ldots + \beta_n \times x_n$

So, y and $x_1 \ldots x_n$ is a linear relationship and can be solved using the linear regression method.

The implementation also supports calculating the F value and $R^2$ to determine statistical significance.

### Prerequisites

- No missing or null data in the inputs.
- The data is numeric, not categorical.
- Given the structure as Y and X1...Xn, there are more than n+1 records available for analysis.

## _SYS_AFL.PAL_POLYNOMIAL_REGRESSION

This is a polynomial regression function.

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<COEFFICIENT table>` | OUT |
| 4 | `<PMML table>` | OUT |
| 5 | `<FITTED table>` | OUT |
| 6 | `<STATISTICS table>` | OUT |
| 7 | `<OPTIMAL PARAM table>` | OUT |

### Input Table

| Table | Column | Reference for Output Table | Data Type | Description |
|---|---|---|---|---|
| DATA | 1st column | ID | INTEGER, VARCHAR, or NVARCHAR | ID.<br>If this column is absent, the `HAS_ID` parameter must be set to 0. |
| | 2nd column | | INTEGER, DOUBLE, or DECIMAL(p, s) | Variable y |
| | 3rd column | | INTEGER, DOUBLE, or DECIMAL(p, s) | Variable X |

### Parameter Table

Mandatory Parameter

The following parameter is mandatory and must be given a value.

| Name | Data Type | Description |
| --- | --- | --- |
| POLYNOMIAL_NUM | INTEGER | This is a mandatory parameter to create a polynomial of degree POLYNOMIAL_NUM model.<br>**Note**: POLYNOMIAL_NUM replaces VARIABLE_NUM. |

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description |
| --- | --- | --- | --- |
| ALG | INTEGER | 1 | Specifies decomposition method:<br>• 0: LU Decomposition (Fast, but requires matrix invertible, can not ensure numerical stable)<br>• 1: QR Decomposition (Fast, does not require matrix invertible, numerical stable)<br>• 2: Singular Value Decomposition (Slow, but does not require matrix invertible, numerical stable)<br>• 5: Cholesky Decomposition (Very Fast, but requires matrix invertible, numerical stable) |
| ADJUSTED_R2 | INTEGER | 0 | • 0: Does not output adjusted R square<br>• 1: Outputs adjusted R square |

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| PMML_EXPORT | INTEGER | 0 | • 0: Does not export polynomial regression model in PMML.<br>• 1: Exports polynomial regression model in PMML in single row.<br>• 2: Exports polynomial regression model in PMML in several rows, and the minimum length of each row is 5000 characters. |
| SELECTED_FEATURES | VARCHAR | No default value | A string to specify the features that will be processed. The pattern is "$X_1,...,X_n$", where $X_i$ is the corresponding column name in the data table. If this parameter is not specified, all the features will be processed, and the third column data will be processed as independent variables. |
| DEPENDENT_VARIABLE | VARCHAR | No default value | Column name in the data table used as dependent variable. If this parameter is not specified, the second column data will be processed as dependent variables. |
| HAS_ID | INTEGER | 1 | Specifies whether the input data table has an ID column. If it exists, the ID column must be the first column.<br>• 0: No ID column<br>• 1: Has an ID column |

## Model Evaluation and Parameter Selection

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| RESAMPLING_METHOD | NVARCHAR | No default value | Specifies the resampling method for model evaluation or parameter selection.<br>• cv<br>• bootstrap<br>If no value is specified for this parameter, neither model evaluation nor parameter selection is activated. | |
| EVALUATION_METRIC | NVARCHAR | No default value | Specifies the evaluation metric for model evaluation or parameter selection.<br>• RMSE | |
| FOLD_NUM | INTEGER | No default value | Specifies the fold number for the cross validation method. | Mandatory and valid only when RESAMPLING_METHOD is set to cv or stratified_cv. |
| REPEAT_TIMES | INTEGER | 1 | Specifies the number of repeat times for resampling. | |
| PARAM_SEARCH_STRATEGY | NVARCHAR | No default value | Specifies the method to activate parameter selection.<br>• grid<br>• random | |
| RANDOM_SEARCH_TIMES | INTEGER | No default value | Specifies the number of times to randomly select candidate parameters for selection. | Mandatory and valid when PARAM_SEARCH_STRATEGY is set to random. |
| SEED | INTEGER | 0 | Specifies the seed for random generation. Use system time when 0 is specified. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| TIMEOUT | INTEGER | 0 | Specifies maximum running time for model evaluation or parameter selection, in seconds. No timeout when 0 is specified. | |
| PROGRESS_INDICATOR_ID | NVARCHAR | No default value | Sets an ID of progress indicator for model evaluation or parameter selection. No progress indicator is active if no value is provided. | |
| POLYNOMIAL_NUM_VALUES | NVARCHAR | No default value | Specifies values of POLYNOMIAL_NUM to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| POLYNOMIAL_NUM_RANGE | NVARCHAR | No default value | Specifies the range of POLYNOMIAL_NUM to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |

### Output Tables

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| COEFFICIENT | 1st column | NVARCHAR(1000) | VARIABLE_NAME | Name of variable |
| | 2nd column | DOUBLE | COEFFICIENT_VALUE | Values of the coefficients |
| PMML | 1st column | INTEGER | ROW_INDEX | ID |
| | 2nd column | NVARCHAR(5000) | MODEL_CONTENT | Polynomial regression model in PMML format |
| FITTED | 1st column | ID (in input table) data type | ID (in input table) column name | ID |
| | 2nd column | DOUBLE | VALUE | Value $Y_i$ |
| STATISTICS | 1st column | NVARCHAR(256) | STAT_NAME | Name |
| | 2nd column | NVARCHAR(1000) | STAT_VALUE | Value |
| OPTIMAL_PARAM | 1st column | NVARCHAR(256) | PARAM_NAME | |
| | 2nd column | INTEGER | INT_VALUE | |

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| | 3rd column | DOUBLE | DOUBLE_VALUE | |
| | 4th column | NVARCHAR(1000) | STRING_VALUE | |

### Examples

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

### Example 1

```
SET SCHEMA DM_PAL;
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE
    #PAL_PARAMETER_TBL
    ("PARAM_NAME" VARCHAR(256), "INT_VALUE" INTEGER, "DOUBLE_VALUE" DOUBLE,
"STRING_VALUE" VARCHAR(1000));
INSERT INTO #PAL_PARAMETER_TBL VALUES ('POLYNOMIAL_NUM',3,NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PMML_EXPORT',2,NULL,NULL);
DROP TABLE PAL_PR_DATA_TBL;
CREATE COLUMN TABLE PAL_PR_DATA_TBL ( "ID" INT,"Y" DOUBLE,"X1" DOUBLE);
INSERT INTO PAL_PR_DATA_TBL VALUES (0,5,1);
INSERT INTO PAL_PR_DATA_TBL VALUES (1,20,2);
INSERT INTO PAL_PR_DATA_TBL VALUES (2,43,3);
INSERT INTO PAL_PR_DATA_TBL VALUES (3,89,4);
INSERT INTO PAL_PR_DATA_TBL VALUES (4,166,5);
INSERT INTO PAL_PR_DATA_TBL VALUES (5,247,6);
INSERT INTO PAL_PR_DATA_TBL VALUES (6,403,7);
CALL _SYS_AFL.PAL_POLYNOMIAL_REGRESSION(PAL_PR_DATA_TBL,
"#PAL_PARAMETER_TBL", ?, ?, ?, ?,?);
```

### Expected Result

COEFFICIENT:

| | VARIABLE_NAME | COEFFICIENT_VALUE |
|---|---|---|
| 1 | __PAL_INTERCEPT__ | -10.999999999999696 |
| 2 | X1__PAL_DELIMIT__1 | 17.24999999999978 |
| 3 | X1__PAL_DELIMIT__2 | -3.4166666666666123 |
| 4 | X1__PAL_DELIMIT__3 | 1.3333333333333293 |

PMML:

| | ROW_INDEX | MODEL_CONTENT |
|---|---|---|
| 1 | 1 | <PMML version="4.0" xmlns="http://www.dmg.org/PMML-4_0" xmlns:xsi="http://www.... |

FITTED:

| | ID | VALUE |
|---|---|---|
| 1 | 0 | 4.1666666666668 |
| 2 | 1 | 20.50000000000005 |
| 3 | 2 | 46.00000000000003 |
| 4 | 3 | 88.6666666666667 |
| 5 | 4 | 156.50000000000... |
| 6 | 5 | 257.5000000000001 |
| 7 | 6 | 399.6666666666667 |

STATISTICS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | R2 | 0.9982396510008842 |
| 2 | F | 567.0691729323303 |
| 3 | MSE | 73.88888888888893 |
| 4 | RMSE | 8.595864638818421 |
| 5 | MAE | 3.999999999999998 |
| 6 | MAPE | 5.3312823227599875 |
| 7 | F_PVALUE | 0.00012531917755756645 |
| 8 | EXPLAINED_VARIANCE | 0.9982396510008842 |
| 9 | MAX_ERROR | 10.5 |

### Example 2: parameter selection

```
SET SCHEMA DM_PAL;
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE
    #PAL_PARAMETER_TBL
    ("PARAM_NAME" VARCHAR(256), "INT_VALUE" INTEGER, "DOUBLE_VALUE" DOUBLE,
"STRING_VALUE" VARCHAR(1000));
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PMML_EXPORT',2,NULL,NULL);
--parameters to be selected
INSERT INTO #PAL_PARAMETER_TBL VALUES ('POLYNOMIAL_NUM_VALUES',NULL,NULL,
'{1,3,2}');
--parameter selection parameters
INSERT INTO #PAL_PARAMETER_TBL VALUES ('RESAMPLING_METHOD', NULL, NULL, 'cv');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('EVALUATION_METRIC', NULL, NULL, 'RMSE');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FOLD_NUM', 2, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('REPEAT_TIMES', 2, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SEED', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PARAM_SEARCH_STRATEGY', NULL, NULL,
'grid');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PROGRESS_INDICATOR_ID', NULL, NULL,
'TEST');
DROP TABLE PAL_PR_DATA_TBL;
CREATE COLUMN TABLE PAL_PR_DATA_TBL ( "ID" INT,"Y" DOUBLE,"X1" DOUBLE);
INSERT INTO PAL_PR_DATA_TBL VALUES (0,5,1);
INSERT INTO PAL_PR_DATA_TBL VALUES (1,20,2);
INSERT INTO PAL_PR_DATA_TBL VALUES (2,43,3);
INSERT INTO PAL_PR_DATA_TBL VALUES (3,89,4);
INSERT INTO PAL_PR_DATA_TBL VALUES (4,166,5);
INSERT INTO PAL_PR_DATA_TBL VALUES (5,247,6);
INSERT INTO PAL_PR_DATA_TBL VALUES (6,403,7);
CALL _SYS_AFL.PAL_POLYNOMIAL_REGRESSION(PAL_PR_DATA_TBL,
"#PAL_PARAMETER_TBL", ?, ?, ?, ?,?);
```

### Expected Result

STATISTICS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | timeout | FALSE |
| 2 | TEST_1_RMSE | 80.2414, 74.3011 |
| 3 | TEST_2_RMSE | 14.0694, 103.983 |
| 4 | TEST_RMSE.MEAN | 68.149 |
| 5 | TEST_RMSE.VAR | 1464.27 |
| 6 | EVAL_RESULTS_1 | {"candidates":[{"TEST_RMS... |
| 7 | R2 | 0.9982396510008842 |
| 8 | F | 567.0691729323303 |
| 9 | MSE | 73.88888888888893 |
| 10 | RMSE | 8.595864638818421 |
| 11 | MAE | 3.999999999999998 |
| 12 | MAPE | 5.3312823227599875 |
| 13 | F_PVALUE | 0.00012531917755756645 |
| 14 | EXPLAINED_VARIANCE | 0.9982396510008842 |
| 15 | MAX_ERROR | 10.5 |

OPTIMAL PARAMETERS:

| | PARAM_NAME | INT_VALUE | DOUBLE_VALUE | STRING_VALUE |
|---|---|---|---|---|
| 1 | POLYNOMIAL_NUM | 3 | ? | ? |

## _SYS_AFL.PAL_POLYNOMIAL_REGRESSION_PREDICT

This function performs prediction with the polynomial regression result.

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<COEFFICIENT table>` | IN |
| 3 | `<PARAMETER table>` | IN |
| 4 | `<FITTED table>` | OUT |

### Input Tables

| Table | Column | Reference for Output Table | Data Type | Description |
|---|---|---|---|---|
| DATA | 1st column | ID | INTEGER, VARCHAR, or NVARCHAR | ID |

| Table | Column | Reference for Output Table | Data Type | Description |
|---|---|---|---|---|
| | 2nd column | | INTEGER, DOUBLE, or DECIMAL(p, s) | Variable X |
| COEFFICIENT | 1st column | | INTEGER, VARCHAR, or NVARCHAR | Variable name or ROW_INDEX for PMML model |
| | 2nd column | | INTEGER, DOUBLE, VARCHAR, NVARCHAR, or CLOB | Values of the coefficients or PMML model content |

The Coefficient Input Table accepts both model formats from the output tables Coefficient and PMML generated by the previous training procedure. To differentiate the two model formats, the predict procedure uses a combination of two column types to identify the model format. Once the model format is determined, the procedure parses the content. Ensure that the column types in the input table correspond to those in either the Coefficient or PMML Model Table, depending on the intended model format.

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| THREAD_RATIO | DOUBLE | 1.0 | Specifies the ratio of total number of threads that can be used by this function. The value range is from 0 to 1, where 0 means only using 1 thread, and 1 means using at most all the currently available threads. Values outside the range will be ignored and this function heuristically determines the number of threads to use. |

### Output Table

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| FITTED | 1st column | ID (in input table) data type | ID (in input table) column name | ID |
| | 2nd column | DOUBLE | VALUE | Value $Y_i$ |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA DM_PAL;
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE
    #PAL_PARAMETER_TBL
    ("PARAM_NAME" VARCHAR(256), "INT_VALUE" INTEGER, "DOUBLE_VALUE" DOUBLE,
"STRING_VALUE" VARCHAR(1000));
INSERT INTO #PAL_PARAMETER_TBL VALUES ('THREAD_RATIO',NULL,0.1,NULL);
DROP TABLE PAL_FPR_PREDICTDATA_TBL;
CREATE COLUMN TABLE PAL_FPR_PREDICTDATA_TBL ( "ID" INT,"X1" DOUBLE);
INSERT INTO PAL_FPR_PREDICTDATA_TBL VALUES (0,0.3);
INSERT INTO PAL_FPR_PREDICTDATA_TBL VALUES (1,4.0);
INSERT INTO PAL_FPR_PREDICTDATA_TBL VALUES (2,1.6);
INSERT INTO PAL_FPR_PREDICTDATA_TBL VALUES (3,0.45);
INSERT INTO PAL_FPR_PREDICTDATA_TBL VALUES (4,1.7);
DROP TABLE PAL_FPR_COEEFICIENT_TBL;
CREATE COLUMN TABLE PAL_FPR_COEEFICIENT_TBL ("VARIABLE_NAME" NVARCHAR(128),
"COEFFICIENT_VALUE" DOUBLE);
INSERT INTO PAL_FPR_COEEFICIENT_TBL VALUES ('__PAL_INTERCEPT__', 4.0);
INSERT INTO PAL_FPR_COEEFICIENT_TBL VALUES ('X1__PAL_DELIMIT__1', 3.0);
INSERT INTO PAL_FPR_COEEFICIENT_TBL VALUES ('X1__PAL_DELIMIT__2', 2.0);
INSERT INTO PAL_FPR_COEEFICIENT_TBL VALUES ('X1__PAL_DELIMIT__3', 1.0);
CALL _SYS_AFL.PAL_POLYNOMIAL_REGRESSION_PREDICT(PAL_FPR_PREDICTDATA_TBL,
PAL_FPR_COEEFICIENT_TBL, "#PAL_PARAMETER_TBL", ?);
```

### Expected Result

FITTED:

| | ID | VALUE |
|---|---|---|
| 1 | 0 | 5.107 |
| 2 | 1 | 112 |
| 3 | 2 | 18.016000000000002 |
| 4 | 3 | 5.846125 |
| 5 | 4 | 19.793 |

> **Note**
>
> The wrapper procedure generation approach to call PAL functions is still supported. For more information, refer to the older versions (prior to SAP HANA Platform 2.0 SPS 02) of this document.

## 5.4 Association Algorithms

This section describes the association algorithms that are provided by the Predictive Analysis Library.

### 5.4.1 Apriori

Apriori is a classic predictive analysis algorithm for finding association rules used in association analysis.

Association analysis uncovers the hidden patterns, correlations or casual structures among a set of items or objects. For example, association analysis enables you to understand what products and services customers tend to purchase at the same time. By analyzing the purchasing trends of your customers with association analysis, you can predict their future behavior.

Apriori is designed to operate on databases containing transactions. As is common in association rule mining, given a set of items, the algorithm attempts to find subsets which are common to at least a minimum number of the item sets. Apriori uses a "bottom up" approach, where frequent subsets are extended one item at a time, a step known as candidate generation, and groups of candidates are tested against the data. The algorithm terminates when no further successful extensions are found. Apriori uses breadth-first search and a tree structure to count candidate item sets efficiently. It generates candidate item sets of length `k` from item sets of length `k-1`, and then prunes the candidates which have an infrequent sub pattern. The candidate set contains all frequent `k`-length item sets. After that, it scans the transaction database to determine frequent item sets among the candidates.

The Apriori function in PAL uses vertical data format to store the transaction data in memory. The function can take VARCHAR/NVARCHAR or INTEGER transaction ID and item ID as input. It supports the output of confidence, support, and lift value, but does not limit the number of output rules. However, you can use SQL script to select the number of output rules, for example:

```
SELECT TOP 2000 FROM RULE_RESULTS where lift > 0.5
```

#### Prerequisites

- The input data does not contain null value.
- There are no duplicated items in each transaction.

#### _SYS_AFL.PAL_APRIORI

This function reads input transaction data and generates association rules by the Apriori algorithm.

**Overview of all Tables**

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<Result table>` | OUT |
| 4 | `<MODEL table>` | OUT |

### Input Table

| Table | Column | Data Type | Description |
|---|---|---|---|
| DATA | 1st column | INTEGER, VARCHAR, or NVARCHAR | Transaction ID |
| | 2nd column | INTEGER, VARCHAR, or NVARCHAR | Item ID |

### Parameter Table

Mandatory Parameters

The following parameters are mandatory and must be given a value.

| Name | Data Type | Description |
|---|---|---|
| MIN_SUPPORT | DOUBLE | User-specified minimum support (actual value). |
| MIN_CONFIDENCE | DOUBLE | User-specified minimum confidence (actual value). |

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| MIN_LIFT | DOUBLE | 0.0 | User-specified minimum lift. |
| MAX_CONSEQUENT | INTEGER | 100 | Maximum length of dependent items. |
| MAXITEMLENGTH | INTEGER | 5 | Total length of leading items and dependent items in the output. |

| Name | Data Type | Default Value | Description |
| --- | --- | --- | --- |
| UBIQUITOUS | DOUBLE | 1.0 | Ignores items whose support values are greater than the UBIQUITOUS value during the frequent items mining phase. |
| IS_USE_PREFIX_TREE | INTEGER | 0 | Indicates whether to use the prefix tree, which can save memory.<br>• 0: Does not use the prefix tree.<br>• 1: Uses the prefix tree. |
| LHS_RESTRICT | VARCHAR | No default value | Specifies that some items are only allowed on the left-hand side of the association rules. |
| RHS_RESTRICT | VARCHAR | No default value | Specifies that some items are only allowed on the right-hand side of the association rules. |

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| LHS_IS_COMPLEMENTARY_RHS | INTEGER | 0 | If you use RHS_RESTRICT to restrict some items to the right-hand side of the association rules, you can set this parameter to 1 to restrict the complementary items to the left-hand side.<br><br>For example, if you have 1000 items (i1, i2, ..., i1000) and want to restrict i1 and i2 to the right-hand side, and i3, i4, ..., i1000 to the left-hand side, you can set the parameters similar to the following:<br><br>`INSERT INTO PAL_CONTROL_TBL VALUES ('RHS_RESTRICT', NULL, NULL, 'i1');`<br><br>`INSERT INTO PAL_CONTROL_TBL VALUES ('RHS_RESTRICT', NULL, NULL, 'i2');`<br><br>`INSERT INTO PAL_CONTROL_TBL VALUES ('LHS_IS_COMPLEMENTARY_RHS', 1, NULL, NULL);` |
| RHS_IS_COMPLEMENTARY_LHS | INTEGER | 0 | If you use LHS_RESTRICT to restrict some items to the left-hand side of the association rules, you can set this parameter to 1 to restrict the complementary items to the right-hand side. |

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| THREAD_RATIO | DOUBLE | 0 | Specifies the ratio of total number of threads that can be used by this function. The value range is from 0 to 1, where 0 means only using 1 thread, and 1 means using at most all the currently available threads. Values outside the range will be ignored and this function heuristically determines the number of threads to use. |
| TIMEOUT | INTEGER | 3600 | Specifies the maximum run time in seconds. The algorithm will stop running when the specified timeout is reached. |
| PMML_EXPORT | INTEGER | 0 | • 0: Does not export Apriori model in PMML.<br>• 1: Exports Apriori model in PMML in single row.<br>• 2: Exports Apriori model in PMML in several rows, and the minimum length of each row is 5000 characters. |

### Output Tables

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| RESULT | 1st column | NVARCHAR(1000) | ANTECEDENT | Leading items |
| | 2nd column | NVARCHAR(1000) | CONSEQUENT | Dependent items |
| | 3rd column | DOUBLE | SUPPORT | Support value |
| | 4th column | DOUBLE | CONFIDENCE | Confidence value |
| | 5th column | DOUBLE | LIFT | Lift value |
| MODEL | 1st column | INTEGER | ROW_INDEX | ID |
| | 2nd column | NVARCHAR(5000) | MODEL_CONTENT | Apriori model in PMML format |

### Example

Assume that:

- DM_PAL is a schema belonging to USER1;
- USER1 has been assigned the AFL__SYS_AFL_AFLPAL_EXECUTE or AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION role.

```
SET SCHEMA DM_PAL ;
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME " VARCHAR(100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (100)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MIN_SUPPORT', null, 0.1, null);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MIN_CONFIDENCE', null, 0.3, null);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MIN_LIFT', null, 1.1, null);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MAX_CONSEQUENT', 1, null, null);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PMML_EXPORT', 1, null, null);
DROP TABLE PAL_APRIORI_TRANS_TBL;
CREATE COLUMN TABLE PAL_APRIORI_TRANS_TBL (
    "CUSTOMER" INTEGER,
    "ITEM" VARCHAR(20)
);
INSERT INTO PAL_APRIORI_TRANS_TBL VALUES (2, 'item2');
INSERT INTO PAL_APRIORI_TRANS_TBL VALUES (2, 'item3');
INSERT INTO PAL_APRIORI_TRANS_TBL VALUES (3, 'item1');
INSERT INTO PAL_APRIORI_TRANS_TBL VALUES (3, 'item2');
INSERT INTO PAL_APRIORI_TRANS_TBL VALUES (3, 'item4');
INSERT INTO PAL_APRIORI_TRANS_TBL VALUES (4, 'item1');
INSERT INTO PAL_APRIORI_TRANS_TBL VALUES (4, 'item3');
INSERT INTO PAL_APRIORI_TRANS_TBL VALUES (5, 'item2');
INSERT INTO PAL_APRIORI_TRANS_TBL VALUES (5, 'item3');
INSERT INTO PAL_APRIORI_TRANS_TBL VALUES (6, 'item1');
INSERT INTO PAL_APRIORI_TRANS_TBL VALUES (6, 'item3');
INSERT INTO PAL_APRIORI_TRANS_TBL VALUES (0, 'item1');
INSERT INTO PAL_APRIORI_TRANS_TBL VALUES (0, 'item2');
INSERT INTO PAL_APRIORI_TRANS_TBL VALUES (0, 'item5');
INSERT INTO PAL_APRIORI_TRANS_TBL VALUES (1, 'item2');
INSERT INTO PAL_APRIORI_TRANS_TBL VALUES (1, 'item4');
INSERT INTO PAL_APRIORI_TRANS_TBL VALUES (7, 'item1');
INSERT INTO PAL_APRIORI_TRANS_TBL VALUES (7, 'item2');
INSERT INTO PAL_APRIORI_TRANS_TBL VALUES (7, 'item3');
INSERT INTO PAL_APRIORI_TRANS_TBL VALUES (7, 'item5');
INSERT INTO PAL_APRIORI_TRANS_TBL VALUES (8, 'item1');
INSERT INTO PAL_APRIORI_TRANS_TBL VALUES (8, 'item2');
INSERT INTO PAL_APRIORI_TRANS_TBL VALUES (8, 'item3');
CALL _SYS_AFL.PAL_APRIORI(PAL_APRIORI_TRANS_TBL, #PAL_PARAMETER_TBL, ?, ?);
```

**Expected Result:**

RESULT:

| | ANTECEDENT | CONSEQUENT | SUPPORT | CONFIDENCE | LIFT |
|---|---|---|---|---|---|
| 1 | item5 | item2 | 0.2222222222222222 | 1 | 1.2857142857142856 |
| 2 | item1 | item5 | 0.2222222222222222 | 0.3333333333333333 | 1.5 |
| 3 | item5 | item1 | 0.2222222222222222 | 1 | 1.5 |
| 4 | item4 | item2 | 0.2222222222222222 | 1 | 1.2857142857142856 |
| 5 | item2&item1 | item5 | 0.2222222222222222 | 0.5 | 2.25 |
| 6 | item5&item1 | item2 | 0.2222222222222222 | 1 | 1.2857142857142856 |
| 7 | item5&item2 | item1 | 0.2222222222222222 | 1 | 1.5 |
| 8 | item5&item3 | item2 | 0.1111111111111111 | 1 | 1.2857142857142856 |
| 9 | item5&item3 | item1 | 0.1111111111111111 | 1 | 1.5 |
| 10 | item1&item4 | item2 | 0.1111111111111111 | 1 | 1.2857142857142856 |
| 11 | item2&item... | item5 | 0.1111111111111111 | 0.5 | 2.25 |
| 12 | item5&item... | item2 | 0.1111111111111111 | 1 | 1.2857142857142856 |
| 13 | item5&item... | item1 | 0.1111111111111111 | 1 | 1.5 |

MODEL:

| | ROW_INDEX | MODEL_CONTENT |
|---|---|---|
| 1 | 1 | <PMML version="4.0" xmlns="http://www.dmg.org/PMML-4_0" xmlns:xsi="http://www.... |

## _SYS_AFL.PAL_APRIORI_RELATIONAL

This function has the same logic with _SYS_AFL.PAL_APRIOR, but it splits the result table into three tables.

**Overview of all Tables**

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<ANTECEDENT table>` | OUT |
| 4 | `<CONSEQUENT table>` | OUT |
| 5 | `<STATISTICS table>` | OUT |
| 6 | `<PMML table>` | OUT |

**Input Table**

| Table | Column | Reference for Output Table | Data Type | Description |
|---|---|---|---|---|
| DATA | 1st column | | INTEGER, VARCHAR, or NVARCHAR | Transaction ID |
| | 2nd column | ITEM_ID | INTEGER, VARCHAR, or NVARCHAR | Item ID |

### Parameter Table

Mandatory Parameters

The following parameters are mandatory and must be given a value.

| Name | Data Type | Description |
|---|---|---|
| MIN_SUPPORT | DOUBLE | User-specified minimum support (actual value). |
| MIN_CONFIDENCE | DOUBLE | User-specified minimum confidence (actual value). |

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| MIN_LIFT | DOUBLE | 0.0 | User-specified minimum lift. |
| MAX_CONSEQUENT | INTEGER | 100 | Maximum length of dependent items. |
| MAXITEMLENGTH | INTEGER | 5 | Total length of leading items and dependent items in the output. |
| UBIQUITOUS | DOUBLE | 1.0 | Ignores items whose support values are greater than the UBIQUITOUS value during the frequent items mining phase. |

| Name | Data Type | Default Value | Description |
| --- | --- | --- | --- |
| IS_USE_PREFIX_TREE | INTEGER | 0 | Indicates whether to use the prefix tree, which can save memory.<br>• 0: Does not use the prefix tree.<br>• 1: Uses the prefix tree. |
| LHS_RESTRICT | VARCHAR | No default value | Specifies that some items are only allowed on the left-hand side of the association rules. |
| RHS_RESTRICT | VARCHAR | No default value | Specifies that some items are only allowed on the right-hand side of the association rules. |

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| LHS_IS_COMPLEMENTARY_RHS | INTEGER | 0 | If you use RHS_RESTRICT to restrict some items to the right-hand side of the association rules, you can set this parameter to 1 to restrict the complementary items to the left-hand side.<br><br>For example, if you have 1000 items (i1, i2, ..., i1000) and want to restrict i1 and i2 to the right-hand side, and i3, i4, ..., i1000 to the left-hand side, you can set the parameters similar to the following:<br><br>`INSERT INTO PAL_CONTROL_TBL VALUES ('RHS_RESTRICT', NULL, NULL, 'i1');`<br><br>`INSERT INTO PAL_CONTROL_TBL VALUES ('RHS_RESTRICT', NULL, NULL, 'i2');`<br><br>`INSERT INTO PAL_CONTROL_TBL VALUES ('LHS_IS_COMPLEMENTARY_RHS', 1, NULL, NULL);` |
| RHS_IS_COMPLEMENTARY_LHS | INTEGER | 0 | If you use LHS_RESTRICT to restrict some items to the left-hand side of the association rules, you can set this parameter to 1 to restrict the complementary items to the right-hand side. |

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| THREAD_RATIO | DOUBLE | 0 | Specifies the ratio of total number of threads that can be used by this function. The value range is from 0 to 1, where 0 means only using 1 thread, and 1 means using at most all the currently available threads. Values outside the range will be ignored and this function heuristically determines the number of threads to use. |
| TIMEOUT | INTEGER | 3600 | Specifies the maximum run time in seconds. The algorithm will stop running when the specified timeout is reached. |
| PMML_EXPORT | DOUBLE | 0 | • 0: Does not export the PMML model.<br>• 1: Exports Apriori model in PMML in single row.<br>• 2: Exports Apriori model in PMML in several rows, and the minimum length of each row is 5000 characters. |

### Output Tables

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| ANTECEDENT | 1st column | INTEGER | RULE_ID | ID |
| | 2nd column | ITEM_ID (in input table) data type | ANTECEDENT+ITEM_ID (in input table) column name | Antecedent item |
| CONSEQUENT | 1st column | INTEGER | RULE_ID | ID |
| | 2nd column | ITEM_ID (in input table) data type | CONSEQUENT+ITEM_ID (in input table) column name | Consequent item |

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| STATISTICS | 1st column | INTEGER | RULE_ID | ID |
| | 2nd column | DOUBLE | SUPPORT | Support |
| | 3rd column | DOUBLE | CONFIDENCE | Confidence |
| | 4th column | DOUBLE | LIFT | Lift |
| MODEL | 1st column | INTEGER | ROW_INDEX | ID |
| | 2nd column | NVARCHAR | MODEL_CONTENT | PMML model |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA DM_PAL ;
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME " VARCHAR(100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (100)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MIN_SUPPORT', null, 0.1, null);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MIN_CONFIDENCE', null, 0.3, null);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MIN_LIFT', null, 1.1, null);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MAX_CONSEQUENT', 1, null, null);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PMML_EXPORT', 1, null, null);
DROP TABLE PAL_APRIORI_TRANS_TBL;
CREATE COLUMN TABLE PAL_APRIORI_TRANS_TBL (
    "CUSTOMER" INTEGER,
    "ITEM" VARCHAR(20)
);
INSERT INTO PAL_APRIORI_TRANS_TBL VALUES (2, 'item2');
INSERT INTO PAL_APRIORI_TRANS_TBL VALUES (2, 'item3');
INSERT INTO PAL_APRIORI_TRANS_TBL VALUES (3, 'item1');
INSERT INTO PAL_APRIORI_TRANS_TBL VALUES (3, 'item2');
INSERT INTO PAL_APRIORI_TRANS_TBL VALUES (3, 'item4');
INSERT INTO PAL_APRIORI_TRANS_TBL VALUES (4, 'item1');
INSERT INTO PAL_APRIORI_TRANS_TBL VALUES (4, 'item3');
INSERT INTO PAL_APRIORI_TRANS_TBL VALUES (5, 'item2');
INSERT INTO PAL_APRIORI_TRANS_TBL VALUES (5, 'item3');
INSERT INTO PAL_APRIORI_TRANS_TBL VALUES (6, 'item1');
INSERT INTO PAL_APRIORI_TRANS_TBL VALUES (6, 'item3');
INSERT INTO PAL_APRIORI_TRANS_TBL VALUES (0, 'item1');
INSERT INTO PAL_APRIORI_TRANS_TBL VALUES (0, 'item2');
INSERT INTO PAL_APRIORI_TRANS_TBL VALUES (0, 'item5');
INSERT INTO PAL_APRIORI_TRANS_TBL VALUES (1, 'item2');
INSERT INTO PAL_APRIORI_TRANS_TBL VALUES (1, 'item4');
INSERT INTO PAL_APRIORI_TRANS_TBL VALUES (7, 'item1');
INSERT INTO PAL_APRIORI_TRANS_TBL VALUES (7, 'item2');
INSERT INTO PAL_APRIORI_TRANS_TBL VALUES (7, 'item3');
INSERT INTO PAL_APRIORI_TRANS_TBL VALUES (7, 'item5');
INSERT INTO PAL_APRIORI_TRANS_TBL VALUES (8, 'item1');
```

```
INSERT INTO PAL_APRIORI_TRANS_TBL VALUES (8, 'item2');
INSERT INTO PAL_APRIORI_TRANS_TBL VALUES (8, 'item3');
CALL _SYS_AFL.PAL_APRIORI_RELATIONAL(PAL_APRIORI_TRANS_TBL,
#PAL_PARAMETER_TBL, ?, ?, ?, ?);
```

**Expected Result:**

ANTECEDENT:

| | RULE_ID | ANTECEDENTITEM |
|---|---|---|
| 1 | 0 | item5 |
| 2 | 1 | item1 |
| 3 | 2 | item5 |
| 4 | 3 | item4 |
| 5 | 4 | item2 |
| 6 | 4 | item1 |
| 7 | 5 | item5 |
| 8 | 5 | item1 |
| 9 | 6 | item5 |
| 10 | 6 | item2 |
| 11 | 7 | item5 |
| 12 | 7 | item3 |
| 13 | 8 | item5 |
| 14 | 8 | item3 |
| 15 | 9 | item1 |
| 16 | 9 | item4 |
| 17 | 10 | item2 |
| 18 | 10 | item1 |
| 19 | 10 | item3 |
| 20 | 11 | item5 |
| 21 | 11 | item1 |
| 22 | 11 | item3 |
| 23 | 12 | item5 |
| 24 | 12 | item2 |
| 25 | 12 | item3 |

CONSEQUENT:

| | RULE_ID | CONSEQUENTITEM |
|---|---|---|
| 1 | 0 | item2 |
| 2 | 1 | item5 |
| 3 | 2 | item1 |
| 4 | 3 | item2 |
| 5 | 4 | item5 |
| 6 | 5 | item2 |
| 7 | 6 | item1 |
| 8 | 7 | item2 |
| 9 | 8 | item1 |
| 10 | 9 | item2 |
| 11 | 10 | item5 |
| 12 | 11 | item2 |
| 13 | 12 | item1 |

STATISTICS:

| | RULE_ID | SUPPORT | CONFIDENCE | LIFT |
|---|---|---|---|---|
| 1 | 0 | 0.2222222222222222 | 1 | 1.2857142857142856 |
| 2 | 1 | 0.2222222222222222 | 0.3333333333333333 | 1.5 |
| 3 | 2 | 0.2222222222222222 | 1 | 1.5 |
| 4 | 3 | 0.2222222222222222 | 1 | 1.2857142857142856 |
| 5 | 4 | 0.2222222222222222 | 0.5 | 2.25 |
| 6 | 5 | 0.2222222222222222 | 1 | 1.2857142857142856 |
| 7 | 6 | 0.2222222222222222 | 1 | 1.5 |
| 8 | 7 | 0.1111111111111111 | 1 | 1.2857142857142856 |
| 9 | 8 | 0.1111111111111111 | 1 | 1.5 |
| 10 | 9 | 0.1111111111111111 | 1 | 1.2857142857142856 |
| 11 | 10 | 0.1111111111111111 | 0.5 | 2.25 |
| 12 | 11 | 0.1111111111111111 | 1 | 1.2857142857142856 |
| 13 | 12 | 0.1111111111111111 | 1 | 1.5 |

MODEL:

| | ROW_INDEX | MODEL_CONTENT |
|---|---|---|
| 1 | 1 | <PMML version="4.0" xmlns="http://www.dmg.org/PMML-4_0" xmlns:xsi="http://www.... |

## _SYS_AFL.PAL_LITE_APRIORI

This is a light association rule mining algorithm to realize the Apriori algorithm. It only calculates two large item sets.

**Overview of all Tables**

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<RESULT table>` | OUT |
| 4 | `<MODEL table>` | OUT |

### Input Table

| Table | Column | Data Type | Description |
|---|---|---|---|
| DATA | 1st column | INTEGER, VARCHAR, or NVARCHAR | Transaction ID |
| | 2nd column | INTEGER, VARCHAR, or NVARCHAR | Item ID |

### Parameter Table

Mandatory Parameters

The following parameters are mandatory and must be given a value.

| Name | Data Type | Description |
|---|---|---|
| MIN_SUPPORT | DOUBLE | User-specified minimum support (actual value). |
| MIN_CONFIDENCE | DOUBLE | User-specified minimum confidence (actual value). |

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description |
| --- | --- | --- | --- |
| THREAD_RATIO | DOUBLE | 0 | Specifies the ratio of total number of threads that can be used by this function. The value range is from 0 to 1, where 0 means only using 1 thread, and 1 means using at most all the currently available threads. Values outside the range will be ignored and this function heuristically determines the number of threads to use. |
| SAMPLE_PROPORTION | DOUBLE | 1 | If you want to use the entire data, set the value to 1.<br><br>If you want to sample the source input data, specify a value as the sampling percentage. |
| IS_RECALCULATE | INTEGER | 1 | If you sample the input data, this parameter controls whether to use the remaining data or not.<br>• 1: Uses the remaining data to update the support, confidence, and lift.<br>• 0: Does not use the remaining data |
| TIMEOUT | INTEGER | 3600 | Specifies the maximum run time in seconds. The algorithm will stop running when the specified timeout is reached. |

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| PMML_EXPORT | INTEGER | 0 | • 0: Does not export liteApriori model in PMML.<br>• 1: Exports liteApriori model in PMML in single row.<br>• 2: Exports liteApriori model in PMML in several rows, and the minimum length of each row is 5000 characters. |

### Output Tables

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| RESULT | 1st column | NVARCHAR(1000) | ANTECEDENT | Leading items |
| | 2nd column | NVARCHAR(1000) | CONSEQUENT | Dependent items |
| | 3rd column | DOUBLE | SUPPORT | Support value |
| | 4th column | DOUBLE | CONFIDENCE | Confidence value |
| | 5th column | DOUBLE | LIFT | Lift value |
| MODEL | 1st column | INTEGER | ROW_INDEX | ID |
| | 2nd column | NVARCHAR(5000) | MODEL_CONTENT | liteApriori model in PMML format |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA DM_PAL ;
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME " VARCHAR(100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (100)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MIN_SUPPORT', null, 0.1, null);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MIN_CONFIDENCE', null, 0.3, null);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PMML_EXPORT', 1, null, null);
```

```
DROP TABLE PAL_APRIORI_TRANS_TBL;
CREATE COLUMN TABLE PAL_APRIORI_TRANS_TBL (
    "CUSTOMER" INTEGER,
    "ITEM" VARCHAR(20)
);
INSERT INTO PAL_APRIORI_TRANS_TBL VALUES (2, 'item2');
INSERT INTO PAL_APRIORI_TRANS_TBL VALUES (2, 'item3');
INSERT INTO PAL_APRIORI_TRANS_TBL VALUES (3, 'item1');
INSERT INTO PAL_APRIORI_TRANS_TBL VALUES (3, 'item2');
INSERT INTO PAL_APRIORI_TRANS_TBL VALUES (3, 'item4');
INSERT INTO PAL_APRIORI_TRANS_TBL VALUES (4, 'item1');
INSERT INTO PAL_APRIORI_TRANS_TBL VALUES (4, 'item3');
INSERT INTO PAL_APRIORI_TRANS_TBL VALUES (5, 'item2');
INSERT INTO PAL_APRIORI_TRANS_TBL VALUES (5, 'item3');
INSERT INTO PAL_APRIORI_TRANS_TBL VALUES (6, 'item1');
INSERT INTO PAL_APRIORI_TRANS_TBL VALUES (6, 'item3');
INSERT INTO PAL_APRIORI_TRANS_TBL VALUES (0, 'item1');
INSERT INTO PAL_APRIORI_TRANS_TBL VALUES (0, 'item2');
INSERT INTO PAL_APRIORI_TRANS_TBL VALUES (0, 'item5');
INSERT INTO PAL_APRIORI_TRANS_TBL VALUES (1, 'item2');
INSERT INTO PAL_APRIORI_TRANS_TBL VALUES (1, 'item4');
INSERT INTO PAL_APRIORI_TRANS_TBL VALUES (7, 'item1');
INSERT INTO PAL_APRIORI_TRANS_TBL VALUES (7, 'item2');
INSERT INTO PAL_APRIORI_TRANS_TBL VALUES (7, 'item3');
INSERT INTO PAL_APRIORI_TRANS_TBL VALUES (7, 'item5');
INSERT INTO PAL_APRIORI_TRANS_TBL VALUES (8, 'item1');
INSERT INTO PAL_APRIORI_TRANS_TBL VALUES (8, 'item2');
INSERT INTO PAL_APRIORI_TRANS_TBL VALUES (8, 'item3');
CALL _SYS_AFL.PAL_LITE_APRIORI(PAL_APRIORI_TRANS_TBL, #PAL_PARAMETER_TBL, ?, ?);
```

**Expected Result**

RESULT:

| | ANTECEDENT | CONSEQUENT | SUPPORT | CONFIDENCE | LIFT |
|---|---|---|---|---|---|
| 1 | item5 | item2 | 0.2222222222222222 | 1 | 1.2857142857142856 |
| 2 | item1 | item5 | 0.2222222222222222 | 0.3333333333333333 | 1.5 |
| 3 | item5 | item1 | 0.2222222222222222 | 1 | 1.5 |
| 4 | item5 | item3 | 0.1111111111111111 | 0.5 | 0.75 |
| 5 | item1 | item2 | 0.4444444444444444 | 0.6666666666666666 | 0.8571428571428571 |
| 6 | item2 | item1 | 0.4444444444444444 | 0.5714285714285714 | 0.8571428571428571 |
| 7 | item4 | item2 | 0.2222222222222222 | 1 | 1.2857142857142856 |
| 8 | item3 | item2 | 0.4444444444444444 | 0.6666666666666666 | 0.8571428571428571 |
| 9 | item2 | item3 | 0.4444444444444444 | 0.5714285714285714 | 0.8571428571428571 |
| 10 | item4 | item1 | 0.1111111111111111 | 0.5 | 0.75 |
| 11 | item3 | item1 | 0.4444444444444444 | 0.6666666666666666 | 1 |
| 12 | item1 | item3 | 0.4444444444444444 | 0.6666666666666666 | 1 |

MODEL:

| | ROW_INDEX | MODEL_CONTENT |
|---|---|---|
| 1 | 1 | <PMML version="4.0" xmlns="http://www.dmg.org/PMML-4_0" xmlns:xsi="http://www.... |

> **Note**
>
> The wrapper procedure generation approach to call PAL functions is still supported. For more information, refer to the older versions (prior to SAP HANA Platform 2.0 SPS 02) of this document.

## 5.4.2 FP-Growth

FP-Growth is an algorithm to find frequent patterns from transactions without generating a candidate itemset.

In PAL, the FP-Growth algorithm is extended to find association rules in three steps:

1. Converts the transactions into a compressed frequent pattern tree (FP-Tree);
2. Recursively finds frequent patterns from the FP-Tree;
3. Generates association rules based on the frequent patterns found in Step 2.

FP-Growth with relational output is also supported.

### Prerequisites

- The input data does not contain null value.
- There are no duplicated items in each transaction.

### _SYS_AFL.PAL_FPGROWTH

This function reads input transaction data and generates association rules by the FP-Growth algorithm.

**Overview of all Tables**

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<RESULT table>` | OUT |

**Input Table**

| Table | Column | Data Type | Description |
|---|---|---|---|
| DATA | 1st column | INTEGER, VARCHAR, or NVARCHAR | Transaction ID |

| Table | Column | Data Type | Description |
|---|---|---|---|
| | 2nd column | INTEGER, VARCHAR, or NVARCHAR | Item ID |

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| MIN_SUPPORT | DOUBLE | 0 | Minimum support.<br>Value range: between 0 and 1 |
| MIN_CONFIDENCE | DOUBLE | 0 | Minimum confidence.<br>Value range: between 0 and 1 |
| MIN_LIFT | DOUBLE | 0 | Minimum lift. |
| MAXITEMLENGTH | INTEGER | 10 | Maximum length of leading items and dependent items in the output. |
| MAX_CONSEQUENT | INTEGER | 10 | Maximum length of right-hand side. |
| UBIQUITOUS | DOUBLE | 1.0 | Ignores items whose support values are greater than the UBIQUITOUS value during the frequent items mining phase. |
| LHS_RESTRICT | VARCHAR | No default value | Specifies which items are allowed in the left-hand side of the association rule. |
| RHS_RESTRICT | VARCHAR | No default value | Specifies which items are allowed in the right-hand side of the association rule. |

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| LHS_IS_COMPLEMENTARY_RHS | INTEGER | 0 | If you use RHS_RESTRICT to restrict some items to the right-hand side of the association rules, you can set this parameter to **1** to restrict the consequent items to the left-hand side.<br><br>For example, if you have 1000 items (i1, i2, ..., i1000) and want to restrict i1 and i2 to the right-hand side, and i3, i4, ..., i1000 to the left-hand side, you can set the parameters similar to the following:<br><br>`INSERT INTO PAL_CONTROL_TBL VALUES ('RHS_RESTRICT', NULL, NULL, 'i1');`<br><br>`INSERT INTO PAL_CONTROL_TBL VALUES ('RHS_RESTRICT', NULL, NULL, 'i2');`<br><br>`INSERT INTO PAL_CONTROL_TBL VALUES ('LHS_IS_COMPLEMENTARY_RHS', 1, NULL, NULL);` |
| RHS_IS_COMPLEMENTARY_LHS | INTEGER | 0 | If you use LHS_RESTRICT to restrict some items to the left-hand side of the association rules, you can set this parameter to **1** to restrict the consequent items to the right-hand side. |

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| THREAD_RATIO | DOUBLE | 0 | Specifies the ratio of total number of threads that can be used by this function. The value range is from 0 to 1, where 0 means only using 1 thread, and 1 means using at most all the currently available threads. Values outside the range will be ignored and this function heuristically determines the number of threads to use. |
| TIMEOUT | INTEGER | 3600 | The function will automatically terminate if its running time is longer than the TIMEOUT value (unit: second). |

### Output Table

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| RESULT | 1st column | NVARCHAR(1000) | ANTECEDENT | Right-hand side items |
| | 2nd column | NVARCHAR(1000) | CONSEQUENT | Left-hand side items |
| | 3rd column | DOUBLE | SUPPORT | Support value |
| | 4th column | DOUBLE | CONFIDENCE | Confidence value |
| | 5th column | DOUBLE | LIFT | Lift value |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA DM_PAL;
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME " VARCHAR(100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (100)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MIN_SUPPORT', null, 0.2, null);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MIN_CONFIDENCE', null, 0.5, null);
```

```
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MIN_LIFT', null, 1, null);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MAXITEMLENGTH', 5, null, null);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MAX_CONSEQUENT', 1, null, null);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('LHS_RESTRICT', null, null, '1');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('LHS_RESTRICT', null, null, '2');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('LHS_RESTRICT', null, null, '3');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('TIMEOUT', 60, null, null);
DROP TABLE PAL_FPGROWTH_DATA_TBL;
CREATE COLUMN TABLE PAL_FPGROWTH_DATA_TBL (
    "TRANS" INTEGER,
    "ITEM" INTEGER
);
INSERT INTO PAL_FPGROWTH_DATA_TBL VALUES (1, 1);
INSERT INTO PAL_FPGROWTH_DATA_TBL VALUES (1, 2);
INSERT INTO PAL_FPGROWTH_DATA_TBL VALUES (2, 2);
INSERT INTO PAL_FPGROWTH_DATA_TBL VALUES (2, 3);
INSERT INTO PAL_FPGROWTH_DATA_TBL VALUES (2, 4);
INSERT INTO PAL_FPGROWTH_DATA_TBL VALUES (3, 1);
INSERT INTO PAL_FPGROWTH_DATA_TBL VALUES (3, 3);
INSERT INTO PAL_FPGROWTH_DATA_TBL VALUES (3, 4);
INSERT INTO PAL_FPGROWTH_DATA_TBL VALUES (3, 5);
INSERT INTO PAL_FPGROWTH_DATA_TBL VALUES (4, 1);
INSERT INTO PAL_FPGROWTH_DATA_TBL VALUES (4, 4);
INSERT INTO PAL_FPGROWTH_DATA_TBL VALUES (4, 5);
INSERT INTO PAL_FPGROWTH_DATA_TBL VALUES (5, 1);
INSERT INTO PAL_FPGROWTH_DATA_TBL VALUES (5, 2);
INSERT INTO PAL_FPGROWTH_DATA_TBL VALUES (6, 1);
INSERT INTO PAL_FPGROWTH_DATA_TBL VALUES (6, 2);
INSERT INTO PAL_FPGROWTH_DATA_TBL VALUES (6, 3);
INSERT INTO PAL_FPGROWTH_DATA_TBL VALUES (6, 4);
INSERT INTO PAL_FPGROWTH_DATA_TBL VALUES (7, 1);
INSERT INTO PAL_FPGROWTH_DATA_TBL VALUES (8, 1);
INSERT INTO PAL_FPGROWTH_DATA_TBL VALUES (8, 2);
INSERT INTO PAL_FPGROWTH_DATA_TBL VALUES (8, 3);
INSERT INTO PAL_FPGROWTH_DATA_TBL VALUES (9, 1);
INSERT INTO PAL_FPGROWTH_DATA_TBL VALUES (9, 2);
INSERT INTO PAL_FPGROWTH_DATA_TBL VALUES (9, 3);
INSERT INTO PAL_FPGROWTH_DATA_TBL VALUES (10, 2);
INSERT INTO PAL_FPGROWTH_DATA_TBL VALUES (10, 3);
INSERT INTO PAL_FPGROWTH_DATA_TBL VALUES (10, 5);
CALL _SYS_AFL.PAL_FPGROWTH(PAL_FPGROWTH_DATA_TBL, #PAL_PARAMETER_TBL, ?);
```

**Expected Result**

RESULT:

| | ANTECEDENT | CONSEQUENT | SUPPORT | CONFIDENCE | LIFT |
|---|---|---|---|---|---|
| 1 | 2 | 3 | 0.5 | 0.7142857142857143 | 1.1904761904761905 |
| 2 | 3 | 2 | 0.5 | 0.8333333333333334 | 1.1904761904761905 |
| 3 | 3 | 4 | 0.3 | 0.5 | 1.25 |
| 4 | 1&2 | 3 | 0.3 | 0.6 | 1 |
| 5 | 1&3 | 2 | 0.3 | 0.75 | 1.0714285714285714 |
| 6 | 1&3 | 4 | 0.2 | 0.5 | 1.25 |

## _SYS_AFL.PAL_FPGROWTH_RELATIONAL

FP-Growth with relational output uses the same algorithm as FP-Growth. The only difference is the output format. The relational output version of FP-Growth separates the result into three tables.

### Overview of all Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<ANTECEDENT table>` | OUT |
| 4 | `<CONSEQUENT table>` | OUT |
| 5 | `<STATISTICS table>` | OUT |

### Input Table

| Table | Column | Reference for Output Table | Data Type | Description |
|---|---|---|---|---|
| DATA | 1st column | | INTEGER, VARCHAR, or NVARCHAR | Transaction ID |
| | 2nd column | ITEM_ID | INTEGER, VARCHAR, or NVARCHAR | Item ID |

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| MIN_SUPPORT | DOUBLE | 0 | Minimum support.<br>Value range: between 0 and 1 |
| MIN_CONFIDENCE | DOUBLE | 0 | Minimum confidence.<br>Value range: between 0 and 1 |
| MIN_LIFT | DOUBLE | 0 | Minimum lift. |
| MAXITEMLENGTH | INTEGER | 10 | Maximum length of leading items and dependent items in the output. |

| Name | Data Type | Default Value | Description |
| --- | --- | --- | --- |
| MAX_CONSEQUENT | INTEGER | 10 | Maximum length of right-hand side. |
| UBIQUITOUS | DOUBLE | 1.0 | Ignores items whose support values are greater than the UBIQUITOUS value during the frequent items mining phase. |
| LHS_RESTRICT | VARCHAR | No default value | Specifies which items are allowed in the left-hand side of the association rule. |
| RHS_RESTRICT | VARCHAR | No default value | Specifies which items are allowed in the right-hand side of the association rule. |

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| LHS_IS_COMPLEMENTARY_RHS | INTEGER | 0 | If you use RHS_RESTRICT to restrict some items to the right-hand side of the association rules, you can set this parameter to **1** to restrict the consequent items to the left-hand side.<br><br>For example, if you have 1000 items (i1, i2, ..., i1000) and want to restrict i1 and i2 to the right-hand side, and i3, i4, ..., i1000 to the left-hand side, you can set the parameters similar to the following:<br><br>`INSERT INTO PAL_CONTROL_TBL VALUES ('RHS_RESTRICT', NULL, NULL, 'i1');`<br><br>`INSERT INTO PAL_CONTROL_TBL VALUES ('RHS_RESTRICT', NULL, NULL, 'i2');`<br><br>`INSERT INTO PAL_CONTROL_TBL VALUES ('LHS_IS_COMPLEMENTARY_RHS', 1, NULL, NULL);` |
| RHS_IS_COMPLEMENTARY_LHS | INTEGER | 0 | If you use LHS_RESTRICT to restrict some items to the left-hand side of the association rules, you can set this parameter to **1** to restrict the consequent items to the right-hand side. |

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| THREAD_RATIO | DOUBLE | 0 | Specifies the ratio of total number of threads that can be used by this function. The value range is from 0 to 1, where 0 means only using 1 thread, and 1 means using at most all the currently available threads. Values outside the range will be ignored and this function heuristically determines the number of threads to use. |
| TIMEOUT | INTEGER | 3600 | The function will automatically terminate if its running time is longer than the TIMEOUT value (unit: second). |

### Output Tables

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| ANTECEDENT | 1st column | INTEGER | RULE_ID | Rule ID |
| | 2nd column | ITEM_ID (in input table) data type | ANTECEDENT+ITEM_ID (in input table) column name | Item |
| CONSEQUENT | 1st column | INTEGER | RULE_ID | Rule ID |
| | 2nd column | ITEM_ID (in input table) data type | CONSEQUENT+ITEM_ID (in input table) column name | Item |
| STATISTICS | 1st column | INTEGER | RULE_ID | Rule ID |
| | 2nd column | DOUBLE | SUPPORT | Support |
| | 3rd column | DOUBLE | CONFIDENCE | Confidence |
| | 4th column | DOUBLE | LIFT | Lift |

### Example

Assume that:

- DM_PAL is a schema belonging to USER1;
- USER1 has been assigned the AFL__SYS_AFL_AFLPAL_EXECUTE or AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION role.

```
SET SCHEMA DM_PAL;
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME " VARCHAR(100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (100)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MIN_SUPPORT', null, 0.2, null);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MIN_CONFIDENCE', null, 0.5, null);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MIN_LIFT', null, 1, null);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MAXITEMLENGTH', 5, null, null);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MAX_CONSEQUENT', 1, null, null);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('LHS_RESTRICT', null, null, '1');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('LHS_RESTRICT', null, null, '2');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('LHS_RESTRICT', null, null, '3');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('TIMEOUT', 60, null, null);
DROP TABLE PAL_FPGROWTH_DATA_TBL;
CREATE COLUMN TABLE PAL_FPGROWTH_DATA_TBL (
    "TRANS" INTEGER,
    "ITEM" INTEGER
);
INSERT INTO PAL_FPGROWTH_DATA_TBL VALUES (1, 1);
INSERT INTO PAL_FPGROWTH_DATA_TBL VALUES (1, 2);
INSERT INTO PAL_FPGROWTH_DATA_TBL VALUES (2, 2);
INSERT INTO PAL_FPGROWTH_DATA_TBL VALUES (2, 3);
INSERT INTO PAL_FPGROWTH_DATA_TBL VALUES (2, 4);
INSERT INTO PAL_FPGROWTH_DATA_TBL VALUES (3, 1);
INSERT INTO PAL_FPGROWTH_DATA_TBL VALUES (3, 3);
INSERT INTO PAL_FPGROWTH_DATA_TBL VALUES (3, 4);
INSERT INTO PAL_FPGROWTH_DATA_TBL VALUES (3, 5);
INSERT INTO PAL_FPGROWTH_DATA_TBL VALUES (4, 1);
INSERT INTO PAL_FPGROWTH_DATA_TBL VALUES (4, 4);
INSERT INTO PAL_FPGROWTH_DATA_TBL VALUES (4, 5);
INSERT INTO PAL_FPGROWTH_DATA_TBL VALUES (5, 1);
INSERT INTO PAL_FPGROWTH_DATA_TBL VALUES (5, 2);
INSERT INTO PAL_FPGROWTH_DATA_TBL VALUES (6, 1);
INSERT INTO PAL_FPGROWTH_DATA_TBL VALUES (6, 2);
INSERT INTO PAL_FPGROWTH_DATA_TBL VALUES (6, 3);
INSERT INTO PAL_FPGROWTH_DATA_TBL VALUES (6, 4);
INSERT INTO PAL_FPGROWTH_DATA_TBL VALUES (7, 1);
INSERT INTO PAL_FPGROWTH_DATA_TBL VALUES (8, 1);
INSERT INTO PAL_FPGROWTH_DATA_TBL VALUES (8, 2);
INSERT INTO PAL_FPGROWTH_DATA_TBL VALUES (8, 3);
INSERT INTO PAL_FPGROWTH_DATA_TBL VALUES (9, 1);
INSERT INTO PAL_FPGROWTH_DATA_TBL VALUES (9, 2);
INSERT INTO PAL_FPGROWTH_DATA_TBL VALUES (9, 3);
INSERT INTO PAL_FPGROWTH_DATA_TBL VALUES (10, 2);
INSERT INTO PAL_FPGROWTH_DATA_TBL VALUES (10, 3);
INSERT INTO PAL_FPGROWTH_DATA_TBL VALUES (10, 5);
CALL _SYS_AFL.PAL_FPGROWTH_RELATIONAL(PAL_FPGROWTH_DATA_TBL,
#PAL_PARAMETER_TBL, ?,?,?);
```

### Expected Results

ANTECEDENT:

| | RULE_ID | ANTECEDENTITEM |
|---|---|---|
| 1 | 0 | 2 |
| 2 | 1 | 3 |
| 3 | 2 | 3 |
| 4 | 3 | 1 |
| 5 | 3 | 2 |
| 6 | 4 | 1 |
| 7 | 4 | 3 |
| 8 | 5 | 1 |
| 9 | 5 | 3 |

CONSEQUENT:

| | RULE_ID | CONSEQUENTITEM |
|---|---|---|
| 1 | 0 | 3 |
| 2 | 1 | 2 |
| 3 | 2 | 4 |
| 4 | 3 | 3 |
| 5 | 4 | 2 |
| 6 | 5 | 4 |

STATISTICS:

| | RULE_ID | SUPPORT | CONFIDENCE | LIFT |
|---|---|---|---|---|
| 1 | 0 | 0.5 | 0.7142857142857143 | 1.1904761904761905 |
| 2 | 1 | 0.5 | 0.8333333333333334 | 1.1904761904761905 |
| 3 | 2 | 0.3 | 0.5 | 1.25 |
| 4 | 3 | 0.3 | 0.6 | 1 |
| 5 | 4 | 0.3 | 0.75 | 1.0714285714285714 |
| 6 | 5 | 0.2 | 0.5 | 1.25 |

> **Note**
>
> The wrapper procedure generation approach to call PAL functions is still supported. For more information, refer to the older versions (prior to SAP HANA Platform 2.0 SPS 02) of this document.

## 5.4.3 K-Optimal Rule Discovery (KORD)

K-optimal rule discovery (KORD) follows the idea of generating association rules with respect to a well-defined measure, instead of first finding all frequent itemsets and then generating all possible rules.

The algorithm only calculates the top-k rules according to that measure. The size of the right hand side (RHS) of those rules is restricted to one. Furthermore, the KORD implementation generates only non-redundant rules.

The algorithm's search strategy is based on the so-called OPUS search. While the search space of all possible LHSs is traversed in a depth-first manner, the information about all qualified RHSs of the rules for a given LHS is propagated further to the deeper search levels. KORD does not build a real tree search structure; instead it traverses the LHSs in a specific order, which allows the pruning of the search space by simply not visiting those itemsets subsequently. In this way it is possible to use pruning rules which restrict the possible LHSs and RHSs at different rule generation stages.

## Prerequisites

- There are no duplicated items in each transaction.
- The input data does not contain null value. The algorithm will issue errors when encountering null values.

## _SYS_AFL.PAL_KORD

### Overview of all Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<ANTECEDENT table>` | OUT |
| 4 | `<CONSEQUENT table>` | OUT |
| 5 | `<STATISTICS table>` | OUT |

### Input Table

| Table | Column | Data Type | Description |
|---|---|---|---|
| DATA | 1st column | INTEGER, VARCHAR, or NVARCHAR | Transaction ID |
| | 2nd column | INTEGER, VARCHAR, or NVARCHAR | Item ID |

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| TOPK | INTEGER | 10 | Specifies the number (k) of top rules. | |
| MAX_ANTECEDENT | INTEGER | 4 | Specifies the maximum length of antecedent rules. | |
| MEASURE_TYPE | INTEGER | 0 | Specifies the measure that will be used to define the priority of the rules.<br>• 0: Leverage<br>• 1: Lift<br>• 2: Support<br>• 3: Confidence<br>Other values are treated as Lift. | |
| IS_USE_EPSILON | INTEGER | 0 | Controls whether to use epsilon to punish the length of rules:<br>• 0: Does not use epsilon<br>• 1: Uses epsilon | |
| MIN_SUPPORT | DOUBLE | 0.0 | Specifies the minimum support. | |
| MIN_CONFIDENCE | DOUBLE | 0.0 | Specifies the minimum confidence. | |
| MIN_COVERAGE | DOUBLE | The value of MIN_SUPPORT | Specifies the minimum coverage.<br>Default: T | |
| MIN_MEASURE | DOUBLE | 0.0 | Specifies the minimum measure value for leverage or lift, dependent on the MEASURE_TYPE setting. | |
| EPSILON | DOUBLE | 0.0 | Epsilon value. | Only valid when IS_USE_EPSILON is 1. |

## Output Tables

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| ANTECEDENT | 1st column | INTEGER | RULE_ID | ID |
| | 2nd column | NVARCHAR(1000) | ANTECEDENT_RULE | Antecedent items |
| CONSEQUENT | 1st column | INTEGER | RULE_ID | ID |
| | 2nd column | NVARCHAR(1000) | CONSEQUENT_RULE | Consequent items |
| STATISTICS | 1st column | INTEGER | RULE_ID | ID |
| | 2nd column | DOUBLE | SUPPORT | Support |
| | 3rd column | DOUBLE | CONFIDENCE | Confidence |
| | 4th column | DOUBLE | LIFT | Lift |
| | 5th column | DOUBLE | LEVERAGE | Leverage |
| | 6th column | DOUBLE | MEASURE | Measure value |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA DM_PAL;
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
       "PARAM_NAME " VARCHAR(100),
       "INT_VALUE" INTEGER,
       "DOUBLE_VALUE" DOUBLE,
       "STRING_VALUE" VARCHAR (100)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('TOPK', 5, null, null);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MEASURE_TYPE', 1, null, null);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MIN_SUPPORT', null, 0.1, null);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MIN_CONFIDENCE', null, 0.2, null);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('IS_USE_EPSILON', 0, null, null);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('EPSILON', null, 0.1, null);
DROP TABLE PAL_KORD_DATA_TBL;
CREATE COLUMN TABLE PAL_KORD_DATA_TBL (
"CUSTOMER" INTEGER,
"ITEM" VARCHAR(20)
);
INSERT INTO PAL_KORD_DATA_TBL VALUES (2, 'item2');
INSERT INTO PAL_KORD_DATA_TBL VALUES (2, 'item3');
INSERT INTO PAL_KORD_DATA_TBL VALUES (3, 'item1');
INSERT INTO PAL_KORD_DATA_TBL VALUES (3, 'item2');
INSERT INTO PAL_KORD_DATA_TBL VALUES (3, 'item4');
INSERT INTO PAL_KORD_DATA_TBL VALUES (4, 'item1');
INSERT INTO PAL_KORD_DATA_TBL VALUES (4, 'item3');
INSERT INTO PAL_KORD_DATA_TBL VALUES (5, 'item2');
INSERT INTO PAL_KORD_DATA_TBL VALUES (5, 'item3');
INSERT INTO PAL_KORD_DATA_TBL VALUES (6, 'item1');
INSERT INTO PAL_KORD_DATA_TBL VALUES (6, 'item3');
```

```
INSERT INTO PAL_KORD_DATA_TBL VALUES (0, 'item1');
INSERT INTO PAL_KORD_DATA_TBL VALUES (0, 'item2');
INSERT INTO PAL_KORD_DATA_TBL VALUES (0, 'item5');
INSERT INTO PAL_KORD_DATA_TBL VALUES (1, 'item2');
INSERT INTO PAL_KORD_DATA_TBL VALUES (1, 'item4');
INSERT INTO PAL_KORD_DATA_TBL VALUES (7, 'item1');
INSERT INTO PAL_KORD_DATA_TBL VALUES (7, 'item2');
INSERT INTO PAL_KORD_DATA_TBL VALUES (7, 'item3');
INSERT INTO PAL_KORD_DATA_TBL VALUES (7, 'item5');
INSERT INTO PAL_KORD_DATA_TBL VALUES (8, 'item1');
INSERT INTO PAL_KORD_DATA_TBL VALUES (8, 'item2');
INSERT INTO PAL_KORD_DATA_TBL VALUES (8, 'item3');
CALL _SYS_AFL.PAL_KORD(PAL_KORD_DATA_TBL, #PAL_PARAMETER_TBL, ?,?,?);
```

**Expected Result**

ANTECEDENT:

| | RULE_ID | ANTECEDENT_RULE |
|---|---|---|
| 1 | 0 | item2 |
| 2 | 1 | item1 |
| 3 | 2 | item2 |
| 4 | 2 | item1 |
| 5 | 3 | item5 |
| 6 | 4 | item2 |

CONSEQUENT:

| | RULE_ID | CONSEQUENT_RULE |
|---|---|---|
| 1 | 0 | item5 |
| 2 | 1 | item5 |
| 3 | 2 | item5 |
| 4 | 3 | item1 |
| 5 | 4 | item4 |

STATISTICS:

| | RULE_ID | SUPPORT | CONFIDENCE | LIFT | LEVERAGE | MEASURE |
|---|---|---|---|---|---|---|
| 1 | 0 | 0.2222222222222222 | 0.2857142857142857 | 1.2857142857142858 | 0.04938271604938271 | 1.2857142857142858 |
| 2 | 1 | 0.2222222222222222 | 0.3333333333333333 | 1.5 | 0.07407407407407407 | 1.5 |
| 3 | 2 | 0.2222222222222222 | 0.5 | 2.25 | 0.12345679012345678 | 2.25 |
| 4 | 3 | 0.2222222222222222 | 1 | 1.5 | 0.07407407407407407 | 1.5 |
| 5 | 4 | 0.2222222222222222 | 0.2857142857142857 | 1.2857142857142858 | 0.04938271604938271 | 1.2857142857142858 |

> **Note**
>
> The wrapper procedure generation approach to call PAL functions is still supported. For more information, refer to the older versions (prior to SAP HANA Platform 2.0 SPS 02) of this document.

## 5.4.4 Sequential Pattern Mining

The sequential pattern mining algorithm searches for frequent patterns in sequence databases.

A sequence database consists of ordered elements or events. For example, a customer first buys bread, then eggs and cheese, and then milk. This forms a sequence consisting of three ordered events. We consider an event or a subsequent event is frequent if its support, which is the number of sequences that contain this event or subsequence, is greater than a certain value. This algorithm finds patterns in input sequences satisfying user defined minimum support.

### Prerequisites

- The input data does not contain null value.
- There are no duplicated items in each transaction.

### _SYS_AFL.PAL_SPM

This function reads input transaction data and generates association rules.

#### Overview of all Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<RESULT table>` | OUT |

#### Input Tables

| Table | Column | Data Type | Description |
|---|---|---|---|
| DATA | 1st column | INTEGER, VARCHAR, or NVARCHAR | Customer ID |
| | 2nd column | INTEGER or TIMESTAMP | Transaction ID |
| | 3rd column | INTEGER, VARCHAR, or NVARCHAR | Item/Event ID |

#### Parameter Table

Mandatory Parameters

The following parameter is mandatory and must be given a value.

| Name | Data Type | Description |
|---|---|---|
| MIN_SUPPORT | DOUBLE | Specifies the minimum support (actual value). |

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| UBIQUITOUS | DOUBLE | 1.0 | Ignores the items whose support values are greater than the specified value during the frequent items mining phase. |
| MIN_EVENT_SIZE | INTEGER | 1 | Minimum number of items in an event. |
| MAX_EVENT_SIZE | INTEGER | 10 | Maximum number of items in an event. |
| MIN_EVENT_LENGTH | INTEGER | 1 | Minimum length of events in the output. |
| MAX_EVENT_LENGTH | INTEGER | 10 | Maximum length of events in the output. |
| ITEM_RESTRICT | VARCHAR | No default value | Specifies which items are allowed in the association rule. |
| MIN_GAP | INTEGER | No default value | The minimum time difference between consecutive events of a sequence.<br>If the data type of the input transaction ID is timestamp, the unit of this parameter is second. |
| CALCULATE_LIFT | INTEGER | 0 | • 0: Only calculates lift values for the cases that the last event contains only one item.<br>• 1: Calculates lift values for all applicable cases. This will take extra time. |

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| TIMEOUT | INTEGER | 3600 | The function will automatically terminate if the run time of the mining process exceeds the specified time (in second). |

### Output Table

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| RESULT | 1st column | NVARCHAR(5000) | PATTERN | This column displays frequent patterns under a specific format. Items inside one pair of brackets represent one event, and items in the brackets have no order (for example, {apple, milk} and {milk, apple} are the same, which means a customer purchased apple and milk together in one transaction.). The order of a certain event is represented by its position in the frequent pattern sequence. For example, "{apple, milk}, {cheese}" means that that a customer first purchases apple and milk in one transaction, then cheese in a later transaction. |
| | 2nd column | DOUBLE | SUPPORT | Support value. The number of sequences that contain the pattern. |

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| | 3rd column | DOUBLE | CONFIDENCE | Confidence value. The confidence of the rule that the last event is the postrule and all other events are the prerules. |
| | 4th column | DOUBLE | LIFT | Lift value. The lift of the rule that the last event is the postrule and all other events are the prerule. |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA DM_PAL;
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME " VARCHAR(100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (100)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MIN_SUPPORT', NULL, 0.5,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('CALCULATE_LIFT', 1, NULL, NULL);
DROP TABLE PAL_SPM_DATA_TBL;
CREATE COLUMN TABLE PAL_SPM_DATA_TBL ("CUSTID" VARCHAR(100), "TRANSID" INT,
"ITEMS" VARCHAR(100));
INSERT INTO PAL_SPM_DATA_TBL VALUES ('A',1,'Apple');
INSERT INTO PAL_SPM_DATA_TBL VALUES ('A',1,'Blueberry');
INSERT INTO PAL_SPM_DATA_TBL VALUES ('A',2,'Apple');
INSERT INTO PAL_SPM_DATA_TBL VALUES ('A',2,'Cherry');
INSERT INTO PAL_SPM_DATA_TBL VALUES ('A',3,'Dessert');
INSERT INTO PAL_SPM_DATA_TBL VALUES ('B',1,'Cherry');
INSERT INTO PAL_SPM_DATA_TBL VALUES ('B',1,'Blueberry');
INSERT INTO PAL_SPM_DATA_TBL VALUES ('B',1,'Apple');
INSERT INTO PAL_SPM_DATA_TBL VALUES ('B',2,'Dessert');
INSERT INTO PAL_SPM_DATA_TBL VALUES ('B',3,'Blueberry');
INSERT INTO PAL_SPM_DATA_TBL VALUES ('C',1,'Apple');
INSERT INTO PAL_SPM_DATA_TBL VALUES ('C',2,'Blueberry');
INSERT INTO PAL_SPM_DATA_TBL VALUES ('C',3,'Dessert');
CALL _SYS_AFL.PAL_SPM(PAL_SPM_DATA_TBL, #PAL_PARAMETER_TBL, ?);
```

### Expected Result

RESULT:

| | PATTERN | SUPPORT | CONFIDENCE | LIFT |
|---|---|---|---|---|
| 1 | {Apple} | 1 | 0 | 0 |
| 2 | {Apple},{Blueberry} | 0.6666666666666666 | 0.6666666666666666 | 0.6666666666666666 |
| 3 | {Apple},{Dessert} | 1 | 1 | 1 |
| 4 | {Apple,Blueberry} | 0.6666666666666666 | 0 | 0 |
| 5 | {Apple,Blueberry... | 0.6666666666666666 | 1 | 1 |
| 6 | {Apple,Cherry} | 0.6666666666666666 | 0 | 0 |
| 7 | {Apple,Cherry},{... | 0.6666666666666666 | 1 | 1 |
| 8 | {Blueberry} | 1 | 0 | 0 |
| 9 | {Blueberry},{Dess... | 1 | 1 | 1 |
| 10 | {Cherry} | 0.6666666666666666 | 0 | 0 |
| 11 | {Cherry},{Dessert} | 0.6666666666666666 | 1 | 1 |
| 12 | {Dessert} | 1 | 0 | 0 |

## _SYS_AFL.PAL_SPM_RELATIONAL

SPM with relational output uses the same algorithm as SPM. The only difference is the output format. The relational output version of SPM separates the result into two tables.

### Overview of all Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<PATTERN table>` | OUT |
| 4 | `<STATISTICS table>` | OUT |

### Input Tables

| Table | Column | Data Type | Description |
|---|---|---|---|
| DATA | 1st column | INTEGER, VARCHAR, or NVARCHAR | Customer ID |
| | 2nd column | INTEGER or TIMESTAMP | Transaction ID |
| | 3rd column | INTEGER, VARCHAR, or NVARCHAR | Item/Event ID |

### Parameter Table

Mandatory Parameters

The following parameter is mandatory and must be given a value.

| Name | Data Type | Description |
|---|---|---|
| MIN_SUPPORT | DOUBLE | Specifies the minimum support (actual value). |

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| UBIQUITOUS | DOUBLE | 1.0 | Ignores the items whose support values are greater than the specified value during the frequent items mining phase. |
| MIN_EVENT_SIZE | INTEGER | 1 | Minimum number of items in an event. |
| MAX_EVENT_SIZE | INTEGER | 10 | Maximum number of items in an event. |
| MIN_EVENT_LENGTH | INTEGER | 1 | Minimum length of events in the output. |
| MAX_EVENT_LENGTH | INTEGER | 10 | Maximum length of events in the output. |
| ITEM_RESTRICT | VARCHAR | No default value | Specifies which items are allowed in the association rule. |
| MIN_GAP | INTEGER | No default value | The minimum time difference between consecutive events of a sequence.<br>If the data type of the input transaction ID is timestamp, the unit of this parameter is second. |

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| CALCULATE_LIFT | INTEGER | 0 | • 0: Only calculates lift values for the cases that the last event contains only one item.<br>• 1: Calculates lift values for all applicable cases. This will take extra time. |
| TIMEOUT | INTEGER | 3600 | The function will automatically terminate if the run time of the mining process exceeds the specified time (in second). |

## Output Tables

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| PATTERN | 1st column | INTEGER | PATTERN_ID | Frequent pattern ID |
| | 2nd column | INTEGER | EVENT_ID | Transaction ID |
| | 3rd column | NVARCHAR(5000) | ITEM | Item |
| STATISTICS | 1st column | INTEGER | PATTERN_ID | Frequent pattern ID, which should correspond to the frequent pattern ID in the PATTERN table. |
| | 2nd column | DOUBLE | SUPPORT | Support value. |
| | 3rd column | DOUBLE | CONFIDENCE | Confidence value. The confidence of the rule that the last event is the postrule and all other events are the prerules. |
| | 4th column | DOUBLE | LIFT | Lift value. The lift of the rule that the last event is the postrule and all other events are the prerule. |

## Example

Assume that:

- DM_PAL is a schema belonging to USER1;
- USER1 has been assigned the AFL__SYS_AFL_AFLPAL_EXECUTE or AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION role.

```
SET SCHEMA DM_PAL;
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME " VARCHAR(100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (100)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MIN_SUPPORT', NULL, 0.5,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('CALCULATE_LIFT', 1, NULL, NULL);
DROP TABLE PAL_SPM_DATA_TBL;
CREATE COLUMN TABLE PAL_SPM_DATA_TBL ("CUSTID" VARCHAR(100), "TRANSID" INT,
"ITEMS" VARCHAR(100));
INSERT INTO PAL_SPM_DATA_TBL VALUES ('A',1,'Apple');
INSERT INTO PAL_SPM_DATA_TBL VALUES ('A',1,'Blueberry');
INSERT INTO PAL_SPM_DATA_TBL VALUES ('A',2,'Apple');
INSERT INTO PAL_SPM_DATA_TBL VALUES ('A',2,'Cherry');
INSERT INTO PAL_SPM_DATA_TBL VALUES ('A',3,'Dessert');
INSERT INTO PAL_SPM_DATA_TBL VALUES ('B',1,'Cherry');
INSERT INTO PAL_SPM_DATA_TBL VALUES ('B',1,'Blueberry');
INSERT INTO PAL_SPM_DATA_TBL VALUES ('B',1,'Apple');
INSERT INTO PAL_SPM_DATA_TBL VALUES ('B',2,'Dessert');
INSERT INTO PAL_SPM_DATA_TBL VALUES ('B',3,'Blueberry');
INSERT INTO PAL_SPM_DATA_TBL VALUES ('C',1,'Apple');
INSERT INTO PAL_SPM_DATA_TBL VALUES ('C',2,'Blueberry');
INSERT INTO PAL_SPM_DATA_TBL VALUES ('C',3,'Dessert');
CALL _SYS_AFL.PAL_SPM_RELATIONAL(PAL_SPM_DATA_TBL, #PAL_PARAMETER_TBL, ?,?);
```

**Expected Result**

PATTERN:

| | PATTERN_ID | EVENT_ID | ITEM |
|---|---|---|---|
| 1 | 1 | 1 | {Apple} |
| 2 | 2 | 1 | {Apple} |
| 3 | 2 | 2 | {Blueberry} |
| 4 | 3 | 1 | {Apple} |
| 5 | 3 | 2 | {Dessert} |
| 6 | 4 | 1 | {Apple,Blueberry} |
| 7 | 5 | 1 | {Apple,Blueberry} |
| 8 | 5 | 2 | {Dessert} |
| 9 | 6 | 1 | {Apple,Cherry} |
| 10 | 7 | 1 | {Apple,Cherry} |
| 11 | 7 | 2 | {Dessert} |
| 12 | 8 | 1 | {Blueberry} |
| 13 | 9 | 1 | {Blueberry} |
| 14 | 9 | 2 | {Dessert} |
| 15 | 10 | 1 | {Cherry} |
| 16 | 11 | 1 | {Cherry} |
| 17 | 11 | 2 | {Dessert} |
| 18 | 12 | 1 | {Dessert} |

STATISTICS:

| | PATTERN_ID | SUPPORT | CONFIDENCE | LIFT |
|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 |
| 2 | 2 | 0.6666666666666666 | 0.6666666666666666 | 0.6666666666666666 |
| 3 | 3 | 1 | 1 | 1 |
| 4 | 4 | 0.6666666666666666 | 0 | 0 |
| 5 | 5 | 0.6666666666666666 | 1 | 1 |
| 6 | 6 | 0.6666666666666666 | 0 | 0 |
| 7 | 7 | 0.6666666666666666 | 1 | 1 |
| 8 | 8 | 1 | 0 | 0 |
| 9 | 9 | 1 | 1 | 1 |
| 10 | 10 | 0.6666666666666666 | 0 | 0 |
| 11 | 11 | 0.6666666666666666 | 1 | 1 |
| 12 | 12 | 1 | 0 | 0 |

ⓘ Note

The wrapper procedure generation approach to call PAL functions is still supported. For more information, refer to the older versions (prior to SAP HANA Platform 2.0 SPS 02) of this document.

## 5.5 Time Series Algorithms

This section describes the time series algorithms that are provided by the Predictive Analysis Library.

Financial market data or economic data usually comes with time stamps. Predicting the future values, such as stock value for tomorrow, is of great interest in many business scenarios. Quantity over time is called time series, and predicting the future value based on existing time series is also known as forecasting. The time series algorithms in PAL include three smoothing based time series models: single, double, and triple exponential smoothing. These models can be used to smooth the existing time series and forecast. In these algorithms, let $x_t$ be the observed values for the $t$-th time period, and $T$ be the total number of time periods.

### 5.5.1 Additive Model Time Series Analysis

Additive model time series analysis uses an additive model to forecast time series data. It handles data with strong seasonal effects and is robust to shift in the historical trend.

Additive model time series analysis – also called as Prophet – uses a decomposable time series model with three main components: trend, seasonality, and holidays or events. The formula is as follows:

$$y(t) = g(t) + s(t) + h(t) + \varepsilon(t)$$

where

- `g(t)` is the trend function which models non-periodic changes.
- `s(t)` represents the periodic changes.
- `h(t)` represents the effect of holidays.
- `ε(t)` is the error term.

**Trend Model**

Prophet implements two trend models:

- Non-linear saturating growth model
- Piecewise linear model

The mathematics form for non-linear saturating growth model is:

$$g(t) = \frac{C}{1 + \exp(-k(t-m))}$$

with `C` representing the carrying capacity, `k` representing the growth rate, and `m` representing an offset parameter.

The growth rate for prophet is not constant. Assume there are `S` change points at times $s_j$, $j = 1,2,\ldots,S$. Define a vector of rate adjustments $\delta \in R^S$, where $\delta_j$ is the change in rate that occurs at time $s_j$, then the rate at any time `t` is then the base rate `k`, plus all of the adjustments up to that point: $k + \sum_{j:t>s_j}\delta_j$ . Defining a vector $a(t) \in \{0, 1\}^S$ such as

$$a_j(t) = \begin{cases} 1, & \text{if } t > s_j \\ 0, & \text{otherwise} \end{cases}$$

The rate at time $t$ is then $k + a(t)^T\delta$, then the corresponding offset adjustment at change point $j$ is:

$$\gamma_j = (s_j - m - \sum_{l<j} \gamma_l)(1 - \frac{k + \sum_{l<j} \delta_l}{k + \sum_{l \le j} \delta_l})$$

The piecewise logistic growth model is then

$$g(t) = \frac{C(t)}{1 + \exp\left(-(k + a(t)^T\delta)\left(t - (m + a(t)^T\gamma)\right)\right)}$$

For linear growth trend, its basic form is

$$g(t) = (k + a(t)^T\delta)t + (m + a(t)^T\gamma)$$

where $\gamma_j$ is set to $-s_j\delta_j$.

**Seasonality:**

Use Fourier series to approximately smooth seasonal effect with:

$$s(t) = \sum_{n=1}^{N}\left(a_n\cos\left(\frac{2\pi nt}{p}\right) + b_n\sin\left(\frac{2\pi nt}{p}\right)\right)$$

$P$ is the regular period, $N$ is the Fourier order. $\beta = [a_1, b_1, \ldots, a_N, b_N]^T$, for example, when $N = 10$:

$$X(t) = \left[\cos\left(\frac{2\pi(1)t}{365.25}\right), \ldots, \sin\left(\frac{2\pi(10)t}{365.25}\right)\right]$$

The seasonal component is then,

$$s(t) = X(t)\beta$$

**Holidays or Events:**

For each holiday $i$, $D_i$ is the set of past and future dates for that holiday. Define an indicator function representing whether time $t$ is happening during holiday $i$, and assign each holiday a parameter $k_i$ which is the corresponding change in the forecast.

$$Z(t) = [1(t \in D_1), \ldots, 1(t \in D_L)]$$

and replacing $h(t)$ with $Z(t)k$.

Stan's L-BFGS is used for model fitting to find a maximum posteriori estimate:

```
Model {
    // Priors
k ~ normal(0,5);
m ~ normal(0,5);
ε ~ normal(0,0.5);
δ ~ double_exponential(0,τ);
β ~ normal(0,σ);
// Logistic likelihood
y ~ normal(C./(1+exp(-(k+A*δ).*(t-(m+A*γ))))+X*β,ε);
    // Linear likelihood
y ~ normal((k+A*δ).*t+(m+A*γ)+X*β,σ);
}
```

# _SYS_AFL.PAL_ADDITIVE_MODEL_ANALYSIS

This function generates the prophet model.

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<HOLIDAY table>` | IN |
| 3 | `<PARAMETER table>` | IN |
| 4 | `<MODEL table>` | OUT |

### Input Table

| Table | Column | Data Type | Description |
|---|---|---|---|
| DATA | 1st column | TIMESTAMP, SECONDDATE, or DATE | Time stamp |
| | 2nd column | INTEGER, DOUBLE, or DECIMAL(p, s) | Raw data |
| | Other columns | INTEGER, DOUBLE, DECIMAL(p, s), VARCHAR, or NVARCHAR | Other regressors' data |
| HOLIDAY | 1st column | TIMESTAMP, SECONDDATE, DATE | Time stamp |
| | 2nd column | VARCHAR, NVARCHAR | Holiday name |
| | 3rd column | INTEGER | Optional, lower window of holiday, which is less than 0 |
| | 4th column | INTEGER | Optional, upper window of holiday, which is greater than 0 |

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| GROWTH | VARCHAR, NVARCHAR | `linear` | Specify a trend:<br>• `linear`: linear trend<br>• `logistic`: logistic trend | |
| CAP | DOUBLE | None | Specify the carrying capacity for logistic growth. | Mandatory and valid only when the value of GROWTH is `logistic`. |
| SEASONALITY_MODE | VARCHAR, NVARCHAR | `additive` | Specify a seasonality mode:<br>• `additive`<br>• `multiplicative` | |
| NUM_CHANGE-POINTS | INTEGER | 25 | Number of potential change points to include. | Not used if CHANGE_POINT is provided. |
| CHANGE-POINT_RANGE | DOUBLE | 0.8 | Proportion of history in which trend change points are estimated. | Not used if CHANGE_POINT is provided. |
| SEASONALITY | VARCHAR, NVARCHAR | None | Add seasonality to model in a json format, including NAME, PERIOD, FOURIER_ORDER, PRIOR_SCALE, and MODE elements. For example:<br>`'{ "NAME": "MONTHLY", "PERIOD":30, "FOURIER_ORDER":5 }'` | A seasonality parameter must include NAME, PERIOD, and FOURIER_ORDER elements. PRIOR_SCALE and MODE elements are optional. FOURIER_ORDER determines how quickly the seasonality can change. PRIOR_SCALE controls the amount of regularization. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| REGRESSOR | VARCHAR, NVARCHAR | None | Specify the regressor in a json format, including PRIOR_SCALE, STANDARDIZE, and MODE elements. For example: `'{ "NAME": "X1", "PRIOR_SCALE" :4, "MODE": "additive" }'` | PRIOR_SCALE controls for the amount of regularization; STANDARDIZE specifies whether or not the regressor is standardized. |
| CHANGE_POINT | VARCHAR, NVARCHAR | None | Specify change point location in a timestamp format, such as '2019-01-01 00:00:00'. | |
| YEARLY_SEASONALITY | INTEGER | -1 | Fit yearly seasonality:<br>• -1: auto<br>• 0: false<br>• 1: true | |
| WEEKLY_SEASONALITY | INTEGER | -1 | Fit weekly seasonality:<br>• -1: auto<br>• 0: false<br>• 1: true | |
| DAILY_SEASONALITY | INTEGER | -1 | Fit daily seasonality:<br>• -1: auto<br>• 0: false<br>• 1: true | |
| SEASONALITY_PRIOR_SCALE | DOUBLE | 10 | Parameter modulating the strength of the seasonality model. | |
| HOLIDAYS_PRIOR_SCALE | DOUBLE | 10 | Parameter modulating the strength of the holiday components model. | |
| CHANGEPOINT_PRIOR_SCALE | DOUBLE | 0.05 | Parameter modulating the flexibility of the automatic change point selection. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| CATEGORICAL_VARIABLE | VARCHAR or NVARCHAR | Detected from input data | Indicate which variables are treated as categorical. The default behavior is:<br>• string: categorical<br>• integer and double: numerical | Valid only for integer or string column. |

### Output Tables

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| MODEL | 1st column | INTEGER | ROW_INDEX | ID |
| | 2nd column | NCLOB | MODEL_CONTENT | Value |

### Examples

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL;
CREATE COLUMN TABLE PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL("ts" DATE, "y" double);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2007-12-10',9.590761139);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2007-12-11',8.519590316);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2007-12-12',8.183676583);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2007-12-13',8.072467369);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2007-12-14',7.893572074);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2007-12-15',7.783640596);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2007-12-16',8.414052432);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2007-12-17',8.829226355);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2007-12-18',8.382518288);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2007-12-19',8.069655307);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2007-12-20',7.879291485);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2007-12-21',7.761744985);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2007-12-22',7.529406458);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL VALUES('2007-12-23',8.38526052);
```

```
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2007-12-24',8.620110725);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2007-12-25',7.852439085);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2007-12-26',7.853993087);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2007-12-27',8.051978079);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2007-12-28',7.926602599);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2007-12-29',7.838343316);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2007-12-30',9.703144581);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2007-12-31',9.385972941);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-01-01',8.293799609);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL VALUES('2008-01-02',8.43468077);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-01-03',8.262042844);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-01-04',8.106816039);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-01-05',7.950149888);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-01-06',9.509259076);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-01-07',8.846784667);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-01-08',8.430545385);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-01-09',8.248267447);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL VALUES('2008-01-10',8.28172399);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-01-11',8.292798858);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-01-12',8.199189359);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-01-13',9.996522419);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-01-14',10.12707101);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-01-15',8.933796044);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-01-16',8.566173814);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-01-17',8.547722396);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-01-18',8.399760095);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-01-19',8.223090551);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-01-20',8.838986793);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-01-21',10.89720218);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-01-22',9.444938073);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-01-23',8.923324744);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-01-24',8.543445563);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-01-25',8.495560891);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-01-26',8.417372856);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-01-27',8.572627898);
```

```
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-01-28',8.736489351);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-01-29',8.634086943);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-01-30',8.673512946);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-02-01',8.824236617);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-02-02',8.537975731);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-02-03',9.698061122);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-02-04',12.09745684);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-02-05',10.63527836);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-02-06',9.691716588);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-02-07',9.315600883);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-02-08',8.970813341);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-02-09',8.589141691);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-02-10',8.617400452);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-02-11',8.616314282);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-02-12',8.215547412);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-02-13',8.064950892);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL VALUES('2008-02-14',8.11342664);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-02-15',7.799343398);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL VALUES('2008-02-16',7.62754439);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-02-17',7.555905094);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL VALUES('2008-02-18',7.71154898);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-02-19',7.788626066);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-02-20',7.708410667);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-02-21',7.768533301);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-02-22',7.695303135);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-02-23',7.378383713);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-02-24',7.910590612);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-02-25',7.626570206);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-02-26',7.573531263);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-02-27',7.567862605);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-02-29',7.552237288);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-03-02',7.336936914);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL VALUES('2008-03-05',8.12474302);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-03-06',7.887584032);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-03-07',7.819636302);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-03-08',7.383989458);
```

```
DROP TABLE PAL_ADDITIVE_MODEL_ANALYSIS_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_ADDITIVE_MODEL_ANALYSIS_PARAMETER_TBL ("NAME" VARCHAR
(50), "INT_VALUE" INTEGER, "DOUBLE_VALUE" DOUBLE, "STRING_VALUE" VARCHAR (100));
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_PARAMETER_TBL VALUES ('GROWTH', NULL,
NULL, 'linear');
DROP TABLE PAL_ADDITIVE_MODEL_ANALYSIS_MODEL;
CREATE COLUMN TABLE PAL_ADDITIVE_MODEL_ANALYSIS_MODEL ("ROW_INDEX" INTEGER,
"MODEL_CONTENT" NCLOB);
DROP TABLE PAL_ADDITIVE_MODEL_ANALYSIS_HOLIDAY;
CREATE COLUMN TABLE PAL_ADDITIVE_MODEL_ANALYSIS_HOLIDAY ("ts" timestamp,
"holiday" NVARCHAR(50));
DO BEGIN
  lt_data = SELECT * FROM PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL;
  lt_holiday = SELECT * FROM PAL_ADDITIVE_MODEL_ANALYSIS_HOLIDAY;
  lt_param = SELECT * FROM PAL_ADDITIVE_MODEL_ANALYSIS_PARAMETER_TBL;
  CALL _SYS_AFL.PAL_ADDITIVE_MODEL_ANALYSIS (:lt_data, :lt_holiday, :lt_param,
lt_model);
  INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_MODEL SELECT * FROM :lt_model;
END;
SELECT * FROM PAL_ADDITIVE_MODEL_ANALYSIS_MODEL;
```

### Expected Result

| | ROW_INDEX | MODEL_CONTENT |
|---|---|---|
| 1 | 0 | {"FLOOR":0.0,"GROWTH":"linear","SEASONALITY_MODE":"additive","beta":[[-0.02142726977552888,-0.02903846... |

## _SYS_AFL.PAL_ADDITIVE_MODEL_PREDICT

This function makes time series forecast based on the estimated PROPHET model.

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<MODEL table>` | IN |
| 3 | `<PARAMETER table>` | IN |
| 4 | `<RESULT table>` | OUT |

### Input Table

| Table | Column | Reference for Output Table | Data Type | Description |
|---|---|---|---|---|
| Data | 1st column | ID | TIMESTAMP, SECOND-DATE, or DATE | Time stamp |
| | 2nd column | | DOUBLE or DECIMAL(p, s) | Placeholder column |

| Table | Column | Reference for Output Table | Data Type | Description |
|---|---|---|---|---|
| | Other columns | | INTEGER, DOUBLE, DECIMAL(p, s), VARCHAR, or NVARCHAR | Other regressors |
| Model | 1st column | | INTEGER | ID |
| | 2nd column | | NCLOB | Model content |

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| CAP | DOUBLE | None | Specify the carrying capacity for logistic growth. | Mandatory and valid only when GROWTH is logistic. |
| INTERVAL_WIDTH | DOUBLE | 0.8 | Width of the uncertainty intervals provided for the forecast. | |
| UNCERTAINTY_SAMPLES | INTEGER | 1000 | Number of simulated draws used to estimate uncertainty intervals. | |

### Output Tables

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| RESULT | 1st column | ID (in input table) column type | ID (in input table) column name | Time stamp |
| | 2nd column | DOUBLE | YHAT | Predicted value |
| | 3rd column | DOUBLE | YHAT_LOWER | Lower bound of the uncertainty interval. |
| | 4th column | DOUBLE | YHAT_UPPER | Upper bound of the uncertainty interval. |

### Examples

Assume that:

- DM_PAL is a schema belonging to USER1;
- USER1 has been assigned the AFL__SYS_AFL_AFLPAL_EXECUTE or AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION role.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL;
CREATE COLUMN TABLE PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL("ts" DATE, "y" double);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES('2008-03-09', 0);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES('2008-03-10', 0);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES('2008-03-11', 0);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES('2008-03-12', 0);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES('2008-03-13', 0);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES('2008-03-14', 0);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES('2008-03-15', 0);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES('2008-03-16', 0);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES('2008-03-17', 0);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES('2008-03-18', 0);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES('2008-03-19', 0);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES('2008-03-20', 0);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES('2008-03-21', 0);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES('2008-03-22', 0);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES('2008-03-23', 0);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES('2008-03-24', 0);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES('2008-03-25', 0);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES('2008-03-26', 0);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES('2008-03-27', 0);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES('2008-03-28', 0);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES('2008-03-29', 0);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES('2008-03-30', 0);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES('2008-03-31', 0);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES('2008-04-01', 0);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES('2008-04-02', 0);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES('2008-04-03', 0);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES('2008-04-04', 0);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES('2008-04-05', 0);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES('2008-04-06', 0);
DROP TABLE PAL_ADDITIVE_MODEL_PREDICT_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_ADDITIVE_MODEL_PREDICT_PARAMETER_TBL("NAME" VARCHAR
(50), "INT_VALUE" INTEGER, "DOUBLE_VALUE" DOUBLE, "STRING_VALUE" VARCHAR (100));
DROP TABLE PAL_ADDITIVE_MODEL_PREDICT_RESULT;
CREATE COLUMN TABLE PAL_ADDITIVE_MODEL_PREDICT_RESULT ("ts" date, "yhat" double,
"yhat_lower" double, "yhat_upper" double);
DO BEGIN
  lt_pdata = SELECT * FROM PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL;
  lt_model = SELECT * FROM PAL_ADDITIVE_MODEL_ANALYSIS_MODEL;
  lt_param = SELECT * FROM PAL_ADDITIVE_MODEL_PREDICT_PARAMETER_TBL;
  CALL _SYS_AFL.PAL_ADDITIVE_MODEL_PREDICT(:lt_pdata, :lt_model, :lt_param,
lt_result);
  INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_RESULT SELECT * FROM :lt_result;
END;
SELECT * FROM PAL_ADDITIVE_MODEL_PREDICT_RESULT;
```

### Expected Result

| | ts | yhat | yhat_lower | yhat_upper |
|---|---|---|---|---|
| 1 | Mar 9, 2008 | 7.680082030588704 | 6.968088075221833 | 8.44375750905527 |
| 2 | Mar 10, 2008 | 8.151029922391375 | 7.376879379545207 | 8.904750465464048 |
| 3 | Mar 11, 2008 | 7.413845101966806 | 6.599017075882863 | 8.216964447617759 |
| 4 | Mar 12, 2008 | 7.20229377978114 | 6.40033087311625 | 7.9031610984468... |
| 5 | Mar 13, 2008 | 7.0906874536511... | 6.284769939677005 | 7.823624683361853 |
| 6 | Mar 14, 2008 | 6.936619616300364 | 6.185901887879514 | 7.674196690771886 |
| 7 | Mar 15, 2008 | 6.688536194576434 | 5.874311255313974 | 7.440130388765489 |
| 8 | Mar 16, 2008 | 7.385135348604989 | 6.627590266131613 | 8.197219121531553 |
| 9 | Mar 17, 2008 | 7.8560832404072... | 7.067215631692029 | 8.647069503401866 |
| 10 | Mar 18, 2008 | 7.118898419983736 | 6.368819676376356 | 7.881993680601273 |
| 11 | Mar 19, 2008 | 6.907347097796976 | 6.112773704969173 | 7.637393094938448 |
| 12 | Mar 20, 2008 | 6.795740771666682 | 5.988101113215993 | 7.557145008950169 |
| 13 | Mar 21, 2008 | 6.641672934315742 | 5.892489628506409 | 7.400826984417631 |
| 14 | Mar 22, 2008 | 6.3935895125926... | 5.629493732818583 | 7.15393523608846 |
| 15 | Mar 23, 2008 | 7.09018866662022 | 6.276704424211055 | 7.889090876060668 |
| 16 | Mar 24, 2008 | 7.561136558423635 | 6.787211007877979 | 8.327099695262657 |
| 17 | Mar 25, 2008 | 6.823951737999295 | 6.0831892364625... | 7.603377025263359 |
| 18 | Mar 26, 2008 | 6.612400415812788 | 5.812211157700925 | 7.413889321531278 |
| 19 | Mar 27, 2008 | 6.500794089682563 | 5.669751566652495 | 7.292564445147262 |
| 20 | Mar 28, 2008 | 6.346726252331958 | 5.564409040576096 | 7.1406644104459... |
| 21 | Mar 29, 2008 | 6.098642830607479 | 5.3402797766236... | 6.89299425640731 |
| 22 | Mar 30, 2008 | 6.7952419846358... | 6.074905042711254 | 7.580932496921763 |

## _SYS_AFL.PAL_ADDITIVE_MODEL_EXPLAIN

This function adds decomposition of forecast compared with _SYS_AFL.PAL_ADDITIVE_MODEL_PREDICT. Forecast values are divided into trend, seasonality and holidays or events. Seasonality part can be further decomposed into yearly, weekly, daily or customized seasonality. If there are extra regressors, the contributions of each regressor will be listed in exogenous column.

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | <DATA table> | IN |
| 2 | <MODEL table> | IN |
| 3 | <PARAMETER table> | IN |
| 4 | <RESULT table> | OUT |
| 5 | <DECOMPOSITION table> | OUT |

### Input table

| Table | Column | Reference for Output Table | Data Type | Description |
|---|---|---|---|---|
| DATA | 1st column | ID | TIMESTAMP, SECOND-DATE, or DATE | Time stamp |
| | 2nd column | | DOUBLE, or DECIMAL (p,s) | Placeholder column |
| | Other columns | | INTEGER, DOUBLE, DECIMAL (p, s), VAR-CHAR, or NVARCHAR | Other regressors |
| MODEL | 1st column | | INTEGER | ID |

| Table | Column | Reference for Output Table | Data Type | Description |
|---|---|---|---|---|
| | 2nd column | | NCLOB | Model content |

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| CAP | DOUBLE | None | Specify the carrying capacity for logistic growth. | Mandatory and valid only when GROWTH is logistic. |
| EXPLAIN_SEASONALITY | INTEGER | 0 | Specify whether or not seasonal component will be decomposed. | |
| EXPLAIN_HOLIDAY | INTEGER | 0 | Specify whether or not holiday component will be decomposed. | |

### Output Table

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| RESULT | 1st column | ID (in input table) column type | ID (in input table) column name | Time stamp |
| | 2nd column | DOUBLE | YHAT | Predicted value |
| | 3rd column | DOUBLE | YHAT_LOWER | Lower bound of the uncertainty interval. |
| | 4th column | DOUBLE | YHAT_UPPER | Upper bound of the uncertainty interval. |

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| DECOMPOSITION | 1st column | ID (in input table) column type | ID (in input table) column name | Time stamp |
| | 2nd column | DOUBLE | TREND | The trend component |
| | 3rd column | NCLOB | SEASONAL | The seasonal component |
| | 4th column | NCLOB | HOLIDAY | The holiday component |
| | 5th column | NCLOB | EXOGENOUS | The contributions of extra regressors to forecast values. |

### Example

Assume that:

- DM_PAL is a schema belonging to USER1.
- USER1 has been assigned the AFL__SYS_AFL_AFLPAL_EXECUTE or AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION role.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL;
CREATE COLUMN TABLE PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL("ts" DATE, "y" DOUBLE);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES('2008-03-09', 0);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES('2008-03-10', 0);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES('2008-03-11', 0);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES('2008-03-12', 0);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES('2008-03-13', 0);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES('2008-03-14', 0);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES('2008-03-15', 0);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES('2008-03-16', 0);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES('2008-03-17', 0);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES('2008-03-18', 0);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES('2008-03-19', 0);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES('2008-03-20', 0);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES('2008-03-21', 0);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES('2008-03-22', 0);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES('2008-03-23', 0);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES('2008-03-24', 0);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES('2008-03-25', 0);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES('2008-03-26', 0);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES('2008-03-27', 0);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES('2008-03-28', 0);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES('2008-03-29', 0);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES('2008-03-30', 0);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES('2008-03-31', 0);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES('2008-04-01', 0);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES('2008-04-02', 0);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES('2008-04-03', 0);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES('2008-04-04', 0);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES('2008-04-05', 0);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES('2008-04-06', 0);
DROP TABLE PAL_ADDITIVE_MODEL_PREDICT_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_ADDITIVE_MODEL_PREDICT_PARAMETER_TBL("NAME" VARCHAR
(50), "INT_VALUE" INTEGER, "DOUBLE_VALUE" DOUBLE, "STRING_VALUE" VARCHAR (100));
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_PARAMETER_TBL VALUES
('EXPLAIN_SEASONALITY', 1, NULL, NULL);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_PARAMETER_TBL VALUES ('EXPLAIN_HOLIDAY',
1, NULL, NULL);
DROP TABLE PAL_ADDITIVE_MODEL_PREDICT_RESULT;
CREATE COLUMN TABLE PAL_ADDITIVE_MODEL_PREDICT_RESULT ("ts" DATE, "yhat" DOUBLE,
"yhat_lower" DOUBLE, "yhat_upper" DOUBLE);
DROP TABLE PAL_ADDITIVE_MODEL_PREDICT_DECOMP;
CREATE COLUMN TABLE PAL_ADDITIVE_MODEL_PREDICT_DECOMP ("ts" DATE, "trend"
DOUBLE, "seasonal" NCLOB, "holiday" NCLOB, "exogenous" NCLOB);
DO BEGIN
  lt_pdata = SELECT * FROM PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL;
  lt_model = SELECT * FROM PAL_ADDITIVE_MODEL_ANALYSIS_MODEL;
  lt_param = SELECT * FROM PAL_ADDITIVE_MODEL_PREDICT_PARAMETER_TBL;
  CALL _SYS_AFL.PAL_ADDITIVE_MODEL_EXPLAIN(:lt_pdata, :lt_model, :lt_param,
lt_result, lt_decomp);
  INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_RESULT SELECT * FROM :lt_result;
  INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DECOMP SELECT * FROM :lt_decomp;
END;
SELECT * FROM PAL_ADDITIVE_MODEL_PREDICT_RESULT;
SELECT * FROM PAL_ADDITIVE_MODEL_PREDICT_DECOMP;
```

### Expected Result

DECOMPOSITION table:

| | ts | trend | seasonal | holiday | exogenous |
|---|---|---|---|---|---|
| 1 | Mar 9, 2008 | 7.440477964912988 | {"weekly":0.24470008765156388} | {} | {} |
| 2 | Mar 10, 2008 | 7.398442832546005 | {"weekly":0.7570396521042914} | {} | {} |
| 3 | Mar 11, 2008 | 7.356407700179022 | {"weekly":0.06219457505223375} | {} | {} |
| 4 | Mar 12, 2008 | 7.314372567812039 | {"weekly":-0.10685903160351618} | {} | {} |
| 5 | Mar 13, 2008 | 7.272337435445057 | {"weekly":-0.17589025172677659} | {} | {} |
| 6 | Mar 14, 2008 | 7.230302303078075 | {"weekly":-0.287257313974045} | {} | {} |
| 7 | Mar 15, 2008 | 7.188267170711093 | {"weekly":-0.4939277175030615} | {} | {} |
| 8 | Mar 16, 2008 | 7.1462320383441... | {"weekly":0.24470008765204183} | {} | {} |
| 9 | Mar 17, 2008 | 7.104196905977127 | {"weekly":0.7570396521043107} | {} | {} |
| 10 | Mar 18, 2008 | 7.062161773610144 | {"weekly":0.06219457505335663} | {} | {} |
| 11 | Mar 19, 2008 | 7.020126641243162 | {"weekly":-0.10685903160348485} | {} | {} |
| 12 | Mar 20, 2008 | 6.9780915088761... | {"weekly":-0.17589025172700116} | {} | {} |
| 13 | Mar 21, 2008 | 6.936056376509197 | {"weekly":-0.2872573139744751} | {} | {} |
| 14 | Mar 22, 2008 | 6.894021244142215 | {"weekly":-0.4939277175026465} | {} | {} |
| 15 | Mar 23, 2008 | 6.851986111775232 | {"weekly":0.244700087651465} | {} | {} |
| 16 | Mar 24, 2008 | 6.809950979408249 | {"weekly":0.7570396521049357} | {} | {} |

## _SYS_AFL.PAL_MASSIVE_ADDITIVE_MODEL_ANALYSIS

This function provides massive mode for _SYS_AFL.PAL_ADDITIVE_MODEL_ANALYSIS. Massive mode can process multiple datasets and parameters in parallel by specifying an additional Group_ID column.

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<HOLIDAY table>` | IN |
| 3 | `<PARAMETER table>` | IN |
| 4 | `<MODEL table>` | OUT |
| 5 | `<ErrorMsgTBL table>` | OUT |

### Input table

| Table | Column | Data Type | Description |
|---|---|---|---|
| DATA | 1st column | INTEGER, VRACHAR or NVARCHAR | Group ID |
| | 2nd column | TIMESTAMP, SECONDDATE, or DATE | Time stamp |

| Table | Column | Data Type | Description |
|---|---|---|---|
| | 3rd column | INTEGER, DOUBLE, or DECIMAL(p, s) | Raw data |
| | Other columns | INTEGER, DOUBLE, DECIMAL(p, s), VARCHAR, or NVARCHAR | Other regressors' data |
| HOLIDAY | 1st column | INTEGER, VRACHAR or NVARCHAR | Group ID |
| | 2nd column | TIMESTAMP, SECONDDATE, DATE | Time stamp |
| | 3rd column | VARCHAR, NVARCHAR | Holiday name |
| | 4th column | INTEGER | Optional, lower window of holiday, which is less than 0 |
| | 5th column | INTEGER | Optional, upper window of holiday, which is greater than 0 |
| PARAMETER | 1st column | INTEGER, VRACHAR or NVARCHAR | Group ID |
| | 2nd column | VARCHAR or NVARCHAR | Parameter name |
| | 3rd column | INTEGER | Integer value |
| | 4th column | DOUBLE | Double value |
| | 5th column | VARCHAR or NVARCHAR | String value |

### Parameter Table

See _SYS_AFL.PAL_ADDITIVE_MODEL_ANALYSIS.

### Output Table

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| MODEL | 1st column | Group ID (in input table) column type | GROUP_ID | Group ID |
| | 2nd column | INTEGER | ROW_INDEX | ID |
| | 3rd column | NCLOB | MODEL_CONTENT | Value |

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| ERRORMsgTBL | 1st column | Group ID (in input table) column type | GROUP_ID | Group ID |
| | 2nd column | VARCHAR, NVARCHAR | ERROR_TIMESTAMP | Error time stamp |
| | 3rd column | INTEGER | ERRORCODE | Error code |
| | 4th column | VARCHAR, NVARCHAR | MESSAGE | Error message |

### Example

Assume that:

- DM_PAL is a schema belonging to USER1;
- USER1 has been assigned the AFL__SYS_AFL_AFLPAL_EXECUTE or AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION role.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_MASSIVE_ADDITIVE_MODEL_ANALYSIS_DATA_TBL;
CREATE COLUMN TABLE PAL_MASSIVE_ADDITIVE_MODEL_ANALYSIS_DATA_TBL("GroupID"
INTEGER, "ts" DATE, "y" DOUBLE);
INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES(0,'2008-02-03',9.698061122);
INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES(0,'2008-02-04',12.09745684);
INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES(0,'2008-02-05',10.63527836);
INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES(0,'2008-02-06',9.691716588);
INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES(0,'2008-02-07',9.315600883);
INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES(0,'2008-02-08',8.970813341);
INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES(0,'2008-02-09',8.589141691);
INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES(0,'2008-02-10',8.617400452);
INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES(0,'2008-02-11',8.616314282);
INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES(0,'2008-02-12',8.215547412);
INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES(0,'2008-02-13',8.064950892);
INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES(0,'2008-02-14',8.11342664);
INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES(0,'2008-02-15',7.799343398);
INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES(0,'2008-02-16',7.62754439);
INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES(0,'2008-02-17',7.555905094);
INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES(0,'2008-02-18',7.71154898);
INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES(0,'2008-02-19',7.788626066);
INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES(0,'2008-02-20',7.708410667);
INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES(0,'2008-02-21',7.768533301);
INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES(0,'2008-02-22',7.695303135);
INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES(0,'2008-02-23',7.378383713);
INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES(0,'2008-02-24',7.910590612);
INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES(0,'2008-02-25',7.626570206);
INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES(0,'2008-02-26',7.573531263);
INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES(0,'2008-02-27',7.567862605);
INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES(0,'2008-02-29',7.552237288);
INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES(0,'2008-03-02',7.336936914);
```

```
INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES(0,'2008-03-05',8.12474302);
INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES(0,'2008-03-06',7.887584032);
INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES(0,'2008-03-07',7.819636302);
INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES(0,'2008-03-08',7.383989458);
INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES(1,'2008-02-03',9.698061122);
INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES(1,'2008-02-04',12.09745684);
INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES(1,'2008-02-05',10.63527836);
INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES(1,'2008-02-06',9.691716588);
INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES(1,'2008-02-07',9.315600883);
INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES(1,'2008-02-08',8.970813341);
INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES(1,'2008-02-09',8.589141691);
INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES(1,'2008-02-10',8.617400452);
INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES(1,'2008-02-11',8.616314282);
INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES(1,'2008-02-12',8.215547412);
INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES(1,'2008-02-13',8.064950892);
INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES(1,'2008-02-14',8.11342664);
INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES(1,'2008-02-15',7.799343398);
INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES(1,'2008-02-16',7.62754439);
INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES(1,'2008-02-17',7.555905094);
INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES(1,'2008-02-18',7.71154898);
INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES(1,'2008-02-19',7.788626066);
INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES(1,'2008-02-20',7.708410667);
INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES(1,'2008-02-21',7.768533301);
INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES(1,'2008-02-22',7.695303135);
INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES(1,'2008-02-23',7.378383713);
INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES(1,'2008-02-24',7.910590612);
INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES(1,'2008-02-25',7.626570206);
INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES(1,'2008-02-26',7.573531263);
INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES(1,'2008-02-27',7.567862605);
INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES(1,'2008-02-29',7.552237288);
INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES(1,'2008-03-02',7.336936914);
INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES(1,'2008-03-05',8.12474302);
INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES(1,'2008-03-06',7.887584032);
INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES(1,'2008-03-07',7.819636302);
```

```
INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES(1,'2008-03-08',7.383989458);
DROP TABLE PAL_MASSIVE_ADDITIVE_MODEL_ANALYSIS_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_MASSIVE_ADDITIVE_MODEL_ANALYSIS_PARAMETER_TBL ("GroupID"
INTEGER, "NAME" VARCHAR(50), "INT_VALUE" INTEGER, "DOUBLE_VALUE" DOUBLE,
"STRING_VALUE" VARCHAR(100));
INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_ANALYSIS_PARAMETER_TBL VALUES (0,
'SEASONALITY_MODE', NULL, NULL, 'additive');
INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_ANALYSIS_PARAMETER_TBL VALUES (1,
'SEASONALITY_MODE', NULL, NULL, 'multiplicative');
DROP TABLE PAL_MASSIVE_ADDITIVE_MODEL_ANALYSIS_HOLIDAY;
CREATE COLUMN TABLE PAL_MASSIVE_ADDITIVE_MODEL_ANALYSIS_HOLIDAY ("GroupID"
INTEGER, "ts" timestamp, "holiday" NVARCHAR(50));
DROP TABLE PAL_MASSIVE_ADDITIVE_MODEL_ANALYSIS_MODEL;
CREATE COLUMN TABLE PAL_MASSIVE_ADDITIVE_MODEL_ANALYSIS_MODEL ("GroupID"
INTEGER, "ROW_INDEX" INTEGER, "MODEL_CONTENT" NCLOB);
DROP TABLE PAL_MASSIVE_ERRORMSG_TBL;
CREATE COLUMN TABLE PAL_MASSIVE_ERRORMSG_TBL ("GroupID" INTEGER,
"ERROR_TIMESTAMP" NVARCHAR(50), "ERRORCODE" INTEGER, "MASSAGE" NVARCHAR(100));
DO BEGIN
  lt_data = SELECT * FROM PAL_MASSIVE_ADDITIVE_MODEL_ANALYSIS_DATA_TBL;
  lt_holiday = SELECT * FROM PAL_MASSIVE_ADDITIVE_MODEL_ANALYSIS_HOLIDAY;
  lt_param = SELECT * FROM PAL_MASSIVE_ADDITIVE_MODEL_ANALYSIS_PARAMETER_TBL;
  CALL _SYS_AFL.PAL_MASSIVE_ADDITIVE_MODEL_ANALYSIS
(:lt_data, :lt_holiday, :lt_param, lt_model, lt_errmsg);
  INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_ANALYSIS_MODEL SELECT * FROM :lt_model;
  INSERT INTO PAL_MASSIVE_ERRORMSG_TBL SELECT * FROM :lt_errmsg;
END;
SELECT * FROM PAL_MASSIVE_ADDITIVE_MODEL_ANALYSIS_MODEL;
SELECT * FROM PAL_MASSIVE_ERRORMSG_TBL;
```

### Expected Result

MODEL table:

| | GroupID | ROW_INDEX | MODEL_CONTENT |
|---|---|---|---|
| 1 | 0 | 0 | {"GROWTH":"linear","FLOOR":0.0,"SEASONALITY_MODE":"additive","start":"... |
| 2 | 1 | 0 | {"GROWTH":"linear","FLOOR":0.0,"SEASONALITY_MODE":"multiplicative","st... |

## _SYS_AFL.PAL_MASSIVE_ADDITIVE_MODEL_PREDICT

This function provides massive mode for _SYS_AFL.PAL_ADDITIVE_MODEL_EXPLAIN. Massive mode can process multiple datasets and parameters in parallel by specifying an additional Group_ID column.

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<MODEL table>` | IN |
| 3 | `<PARAMETER table>` | IN |
| 4 | `<RESULT table>` | OUT |
| 5 | `<DECOMPOSITION table>` | OUT |

| Position | Table Name | Parameter Type |
|---|---|---|
| 6 | `<ErrorMsgTBL table>` | OUT |

### Input table

| Table | Column | Reference for Output Table | Data Type | Description |
|---|---|---|---|---|
| DATA | 1st column | | INTEGER, VRACHAR or NVARCHAR | Group ID |
| | 2nd column | ID | TIMESTAMP, SECOND-DATE, or DATE | Time stamp |
| | 3rd column | | DOUBLE, or DECI-MAL(p,s) | Placeholder column |
| | Other columns | | INTEGER, DOUBLE, DECIMAL(p, s), VAR-CHAR, or NVARCHAR | Other regressors |
| MODEL | 1st column | | INTEGER, VRACHAR or NVARCHAR | Group ID |
| | 2nd column | | INTEGER | ID |
| | 3rd column | | NCLOB | Model content |
| PARAMETER | 1st column | | INTEGER, VRACHAR or NVARCHAR | Group ID |
| | 2nd column | | VARCHAR or NVARCHAR | Parameter name |
| | 3rd column | | INTEGER | Integer value |
| | 4th column | | DOUBLE | Double value |
| | 5th column | | VARCHAR or NVARCHAR | String value |

### Parameter Table

See _SYS_AFL.PAL_ADDITIVE_MODEL_EXPLAIN.

### Output Table

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| RESULT | 1st column | Group ID (in input ta-ble) column type | GROUP_ID | Group ID |
| | 2nd column | ID (in input table) col-umn type | ID (in input table) col-umn name | Time stamp |
| | 3rd column | DOUBLE | YHAT | Predicted value |
| | 4th column | DOUBLE | YHAT_LOWER | Lower bound of the un-certainty interval |

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| | 5th column | DOUBLE | YHAT_UPPER | Upper bound of the uncertainty interval |

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| DECOMPOSITION | 1st column | Group ID (in input table) column type | GROUP_ID | Group ID |
| | 2nd column | ID (in input table) column type | ID (in input table) column name | Time stamp |
| | 3rd column | DOUBLE | TREND | The trend component |
| | 4th column | NCLOB | SEASONAL | The seasonal component |
| | 5th column | NCLOB | HOLIDAY | The holiday component |
| | 6th Column | NCLOB | EXOGENOUS | The contributions of extra regressors to forecast values |

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| ERRORMsgTBL | 1st column | Group ID (in input table) column type | GROUP_ID | Group ID |
| | 2nd column | VARCHAR, NVARCHAR | ERROR_TIMESTAMP | Error time stamp |
| | 3rd column | INTEGER | ERRORCODE | Error code |
| | 4th column | VARCHAR, NVARCHAR | MESSAGE | Error message |

### Example

Assume that:

- DM_PAL is a schema belonging to USER1;
- USER1 has been assigned the AFL__SYS_AFL_AFLPAL_EXECUTE or AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION role.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_MASSIVE_ADDITIVE_MODEL_PREDICT_DATA_TBL;
CREATE COLUMN TABLE PAL_MASSIVE_ADDITIVE_MODEL_PREDICT_DATA_TBL("GroupID"
INTEGER, "ts" DATE, "y" DOUBLE);
INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES(0,'2008-03-09',
0);
INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES(0,'2008-03-10',
0);
INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES(0,'2008-03-11',
0);
INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES(0,'2008-03-12',
0);
INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES(0,'2008-03-13',
0);
INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES(0,'2008-03-14',
0);
```

```
INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES(0,'2008-03-15',
0);
INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES(0,'2008-03-16',
0);
INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES(0,'2008-03-17',
0);
INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES(0,'2008-03-18',
0);
INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES(1,'2008-03-09',
0);
INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES(1,'2008-03-10',
0);
INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES(1,'2008-03-11',
0);
INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES(1,'2008-03-12',
0);
INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES(1,'2008-03-13',
0);
INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES(1,'2008-03-14',
0);
INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES(1,'2008-03-15',
0);
INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES(1,'2008-03-16',
0);
INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES(1,'2008-03-17',
0);
INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES(1,'2008-03-18',
0);
INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES(1,'2008-03-19',
0);
DROP TABLE PAL_MASSIVE_ADDITIVE_MODEL_PREDICT_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_MASSIVE_ADDITIVE_MODEL_PREDICT_PARAMETER_TBL ("GroupID"
INTEGER, "NAME" VARCHAR(50), "INT_VALUE" INTEGER, "DOUBLE_VALUE" DOUBLE,
"STRING_VALUE" VARCHAR(100));
INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_PREDICT_PARAMETER_TBL VALUES (0,
'EXPLAIN_SEASONALITY', 0, NULL, NULL);
INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_PREDICT_PARAMETER_TBL VALUES (1,
'EXPLAIN_SEASONALITY', 1, NULL, NULL);
DROP TABLE PAL_MASSIVE_ADDITIVE_MODEL_PREDICT_RESULT;
CREATE COLUMN TABLE PAL_MASSIVE_ADDITIVE_MODEL_PREDICT_RESULT ("GroupID"
INTEGER, "ts" DATE, "yhat" DOUBLE, "yhat_lower" DOUBLE, "yhat_upper" DOUBLE);
DROP TABLE PAL_MASSIVE_ADDITIVE_MODEL_PREDICT_DECOMP;
CREATE COLUMN TABLE PAL_MASSIVE_ADDITIVE_MODEL_PREDICT_DECOMP ("GroupID"
INTEGER, "ts" DATE, "trend" DOUBLE, "seasonal" NCLOB, "holiday" NCLOB,
"exogenous" NCLOB);
DROP TABLE PAL_MASSIVE_ERRORMSG_TBL;
CREATE COLUMN TABLE PAL_MASSIVE_ERRORMSG_TBL ("GroupID" INTEGER,
"ERROR_TIMESTAMP" NVARCHAR(50), "ERRORCODE" INTEGER, "MASSAGE" NVARCHAR(100));
DO BEGIN
  lt_pdata = SELECT * FROM PAL_MASSIVE_ADDITIVE_MODEL_PREDICT_DATA_TBL;
  lt_model = SELECT * FROM PAL_MASSIVE_ADDITIVE_MODEL_ANALYSIS_MODEL;
  lt_param = SELECT * FROM PAL_MASSIVE_ADDITIVE_MODEL_PREDICT_PARAMETER_TBL;
  CALL
_SYS_AFL.PAL_MASSIVE_ADDITIVE_MODEL_PREDICT(:lt_pdata, :lt_model, :lt_param,
lt_result, lt_decomp, lt_errmsg);
  INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_PREDICT_RESULT SELECT * FROM :lt_result;
  INSERT INTO PAL_MASSIVE_ADDITIVE_MODEL_PREDICT_DECOMP SELECT * FROM :lt_decomp;
  INSERT INTO PAL_MASSIVE_ERRORMSG_TBL SELECT * FROM :lt_errmsg;
END;
SELECT * FROM PAL_MASSIVE_ADDITIVE_MODEL_PREDICT_RESULT;
SELECT * FROM PAL_MASSIVE_ADDITIVE_MODEL_PREDICT_DECOMP;
SELECT * FROM PAL_MASSIVE_ERRORMSG_TBL;
```

### Expected Result

RESULT table:

| | GroupID | ts | yhat | yhat_lower | yhat_upper |
|---|---|---|---|---|---|
| 1 | 0 | Mar 9, 2008 | 7.496156576322274 | 6.946512866300335 | 7.999520102236311 |
| 2 | 0 | Mar 10, 2008 | 8.213313718062468 | 7.672419222264816 | 8.7389569212497 |
| 3 | 0 | Mar 11, 2008 | 7.858537142888998 | 7.335136337567891 | 8.352760709705347 |
| 4 | 0 | Mar 12, 2008 | 7.760384324726053 | 7.23083970120614 | 8.248489546264262 |
| 5 | 0 | Mar 13, 2008 | 7.757540287690886 | 7.253426928396245 | 8.274076288585906 |
| 6 | 0 | Mar 14, 2008 | 7.6436253657405... | 7.091636476440785 | 8.196372497058146 |
| 7 | 0 | Mar 15, 2008 | 7.4069900656345... | 6.860876074619777 | 7.950315777470149 |
| 8 | 0 | Mar 16, 2008 | 7.532388284113138 | 7.007749069055368 | 8.08638727386506 |
| 9 | 0 | Mar 17, 2008 | 8.249545425853162 | 7.734199705371561 | 8.796828630507118 |
| 10 | 0 | Mar 18, 2008 | 7.894768850679773 | 7.351444512092524 | 8.481300299653487 |
| 11 | 1 | Mar 9, 2008 | 7.512597349512471 | 7.09242906186976 | 7.938037147299898 |
| 12 | 1 | Mar 10, 2008 | 8.28442634957331 | 7.871806733744034 | 8.703364290483927 |
| 13 | 1 | Mar 11, 2008 | 7.928944343831213 | 7.486818458815916 | 8.37993896578833 |
| 14 | 1 | Mar 12, 2008 | 7.814850632434616 | 7.377748512806855 | 8.248966528011424 |
| 15 | 1 | Mar 13, 2008 | 7.817957589113324 | 7.369646519917282 | 8.271773126363097 |

DECOMPOSITION table:

| | GroupID | ts | trend | seasonal | holiday | exogenous |
|---|---|---|---|---|---|---|
| 1 | 0 | Mar 9, 2008 | 7.718264622527421 | {"seasonalities":-0.22210804620514677} | {} | {} |
| 2 | 0 | Mar 10, 2008 | 7.723440580783211 | {"seasonalities":0.4898731372792571} | {} | {} |
| 3 | 0 | Mar 11, 2008 | 7.728616539038999 | {"seasonalities":0.12992060384999869} | {} | {} |
| 4 | 0 | Mar 12, 2008 | 7.733792497294789 | {"seasonalities":0.02659182743126505... | {} | {} |
| 5 | 0 | Mar 13, 2008 | 7.738968455550577 | {"seasonalities":0.01857183214030899} | {} | {} |
| 6 | 0 | Mar 14, 2008 | 7.744144413806366 | {"seasonalities":-0.1005190480657729} | {} | {} |
| 7 | 0 | Mar 15, 2008 | 7.749320372062154 | {"seasonalities":-0.34233030642758685} | {} | {} |
| 8 | 0 | Mar 16, 2008 | 7.754496330317942 | {"seasonalities":-0.22210804620480407} | {} | {} |
| 9 | 0 | Mar 17, 2008 | 7.759672288573732 | {"seasonalities":0.48987313727942996} | {} | {} |
| 10 | 0 | Mar 18, 2008 | 7.76484824682952 | {"seasonalities":0.12992060385025309} | {} | {} |
| 11 | 1 | Mar 9, 2008 | 7.769244102007118 | {"weekly":-0.03303368373100087} | {} | {} |
| 12 | 1 | Mar 10, 2008 | 7.778040744211441 | {"weekly":0.06510451950752908} | {} | {} |
| 13 | 1 | Mar 11, 2008 | 7.786837386415764 | {"weekly":0.018249637222854697} | {} | {} |
| 14 | 1 | Mar 12, 2008 | 7.7956340286200... | {"weekly":0.0024650469408874565} | {} | {} |
| 15 | 1 | Mar 13, 2008 | 7.8044306708244... | {"weekly":0.0017332357553618778} | {} | {} |

## 5.5.1.1 Customized Seasonalities

The algorithm uses a partial Fourier sum to estimate periodic effects.

By default yearly seasonality is fitted if the time series is more than a year of data long. Similarly, weekly seasonality is fitted if the time series is longer than one week. And daily seasonality will only be considered for a sub-daily time series.

The number of terms in the partial sum determines how quickly the seasonality can change. For reference, the default number of Fourier terms for yearly and weekly seasonalities are 10 and 3, respectively. Increasing the Fourier order allows the seasonality to fit higher-frequency changes, but can also lead to overfitting. In most cases, the default values are appropriate.

In addition to three auto-detected seasonalities, user can add other seasonalities (monthly, quarterly, hourly, etc.) with parameter SEASONALITY. It accepts a string in json format contains a name, the period of the seasonality in days, the Fourier order, and optional elements prior scale and mode for the seasonality.

## Relevant Parameters

- SEASONALITY
  Add customized seasonality to model in a json format, including NAME, PERIOD, FOURIER_ORDER, PRIOR_SCALE, and MODE elements. The PRIOR_SCALE and MODE are optional.
- YEARLY_SEASONALITY
- WEEKLY_SEASONALITY
- DAILY_SEASONALITY
  These parameters control whether yearly, weekly, and daily seasonalities are fitted. Options are -1: auto, 0: false, and 1: ture.
- SEASONALITY_MODE
  Options are `additive` and `multiplicative`. By default the algorithm fits additive seasonalities, meaning the effect of the seasonality is added to the trend to get the forecast. If the seasonal fluctuations grow with the trend, this is multiplicative seasonality. Any added seasonalities will by default use SEASONALITY_MODE's setting, but can be overridden by specifying MODE as an element when adding the seasonality. Note that it is pretty unlikely to have a mix of additive and multiplicative seasonalities.
- SEASONALITY_PRIOR_SCALE
  This parameter adjusts the extent to which the seasonality model will fit the data. A large value allows the seasonality to fit large fluctuations, a small value shrinks the impact of the seasonality. By default this parameter is 10, which provides little regularization. If the seasonalities are overfitting, adjusting their prior scale will smooth them.

## Example

As an example, we fit the same data from the main document, but replace the weekly seasonality with monthly seasonality. The monthly seasonality then will appear in the decomposition of result.

**Example**

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL;
CREATE COLUMN TABLE PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL("ts" DATE, "y" double);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2007-12-10',9.590761139);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2007-12-11',8.519590316);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2007-12-12',8.183676583);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2007-12-13',8.072467369);
```

```
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2007-12-14',7.893572074);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2007-12-15',7.783640596);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2007-12-16',8.414052432);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2007-12-17',8.829226355);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2007-12-18',8.382518288);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2007-12-19',8.069655307);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2007-12-20',7.879291485);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2007-12-21',7.761744985);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2007-12-22',7.529406458);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL VALUES('2007-12-23',8.38526052);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2007-12-24',8.620110725);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2007-12-25',7.852439085);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2007-12-26',7.853993087);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2007-12-27',8.051978079);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2007-12-28',7.926602599);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2007-12-29',7.838343316);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2007-12-30',9.703144581);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2007-12-31',9.385972941);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-01-01',8.293799609);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL VALUES('2008-01-02',8.43468077);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-01-03',8.262042844);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-01-04',8.106816039);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-01-05',7.950149888);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-01-06',9.509259076);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-01-07',8.846784667);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-01-08',8.430545385);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-01-09',8.248267447);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL VALUES('2008-01-10',8.28172399);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-01-11',8.292798858);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-01-12',8.199189359);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-01-13',9.996522419);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-01-14',10.12707101);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-01-15',8.933796044);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-01-16',8.566173814);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-01-17',8.547722396);
```

```
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-01-18',8.399760095);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-01-19',8.223090551);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-01-20',8.838986793);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-01-21',10.89720218);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-01-22',9.444938073);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-01-23',8.923324744);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-01-24',8.543445563);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-01-25',8.495560891);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-01-26',8.417372856);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-01-27',8.572627898);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-01-28',8.736489351);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-01-29',8.634086943);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-01-30',8.673512946);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-02-01',8.824236617);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-02-02',8.537975731);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-02-03',9.698061122);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-02-04',12.09745684);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-02-05',10.63527836);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-02-06',9.691716588);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-02-07',9.315600883);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-02-08',8.970813341);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-02-09',8.589141691);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-02-10',8.617400452);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-02-11',8.616314282);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-02-12',8.215547412);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-02-13',8.064950892);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL VALUES('2008-02-14',8.11342664);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-02-15',7.799343398);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL VALUES('2008-02-16',7.62754439);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-02-17',7.555905094);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL VALUES('2008-02-18',7.71154898);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-02-19',7.788626066);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-02-20',7.708410667);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-02-21',7.768533301);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-02-22',7.695303135);
```

```
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-02-23',7.378383713);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-02-24',7.910590612);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-02-25',7.626570206);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-02-26',7.573531263);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-02-27',7.567862605);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-02-29',7.552237288);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-03-02',7.336936914);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL VALUES('2008-03-05',8.12474302);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-03-06',7.887584032);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-03-07',7.819636302);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL
VALUES('2008-03-08',7.383989458);
DROP TABLE PAL_ADDITIVE_MODEL_ANALYSIS_HOLIDAY;
CREATE COLUMN TABLE PAL_ADDITIVE_MODEL_ANALYSIS_HOLIDAY ("ts" timestamp,
"holiday" NVARCHAR(50));
DROP TABLE PAL_ADDITIVE_MODEL_ANALYSIS_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_ADDITIVE_MODEL_ANALYSIS_PARAMETER_TBL ("NAME" VARCHAR
(50), "INT_VALUE" INTEGER, "DOUBLE_VALUE" DOUBLE, "STRING_VALUE" VARCHAR (100));
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_PARAMETER_TBL VALUES ('GROWTH', NULL,
NULL, 'linear');
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_PARAMETER_TBL VALUES
('WEEKLY_SEASONALITY', 0, NULL, NULL);
INSERT INTO PAL_ADDITIVE_MODEL_ANALYSIS_PARAMETER_TBL VALUES ('SEASONALITY',
NULL, NULL, '{ "NAME": "MONTHLY", "PERIOD":30, "FOURIER_ORDER":5 }');
DROP TABLE PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL;
CREATE COLUMN TABLE PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL("ts" DATE, "y" DOUBLE);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES('2008-03-09', 0);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES('2008-03-10', 0);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES('2008-03-11', 0);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES('2008-03-12', 0);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES('2008-03-13', 0);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES('2008-03-14', 0);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES('2008-03-15', 0);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES('2008-03-16', 0);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES('2008-03-17', 0);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES('2008-03-18', 0);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES('2008-03-19', 0);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES('2008-03-20', 0);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES('2008-03-21', 0);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES('2008-03-22', 0);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES('2008-03-23', 0);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES('2008-03-24', 0);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES('2008-03-25', 0);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES('2008-03-26', 0);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES('2008-03-27', 0);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES('2008-03-28', 0);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES('2008-03-29', 0);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES('2008-03-30', 0);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES('2008-03-31', 0);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES('2008-04-01', 0);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES('2008-04-02', 0);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES('2008-04-03', 0);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES('2008-04-04', 0);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES('2008-04-05', 0);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL VALUES('2008-04-06', 0);
DROP TABLE PAL_ADDITIVE_MODEL_PREDICT_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_ADDITIVE_MODEL_PREDICT_PARAMETER_TBL("NAME" VARCHAR
(50), "INT_VALUE" INTEGER, "DOUBLE_VALUE" DOUBLE, "STRING_VALUE" VARCHAR (100));
```

```
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_PARAMETER_TBL VALUES
('EXPLAIN_SEASONALITY', 1, NULL, NULL);
INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_PARAMETER_TBL VALUES ('EXPLAIN_HOLIDAY',
1, NULL, NULL);
DROP TABLE PAL_ADDITIVE_MODEL_PREDICT_DECOMP;
CREATE COLUMN TABLE PAL_ADDITIVE_MODEL_PREDICT_DECOMP ("ts" DATE, "trend"
DOUBLE, "seasonal" NCLOB, "holiday" NCLOB, "exogenous" NCLOB);
DO BEGIN
  lt_data = SELECT * FROM PAL_ADDITIVE_MODEL_ANALYSIS_DATA_TBL;
  lt_holiday = SELECT * FROM PAL_ADDITIVE_MODEL_ANALYSIS_HOLIDAY;
  lt_param = SELECT * FROM PAL_ADDITIVE_MODEL_ANALYSIS_PARAMETER_TBL;
  CALL _SYS_AFL.PAL_ADDITIVE_MODEL_ANALYSIS (:lt_data, :lt_holiday, :lt_param,
lt_model);
  lt_pdata = SELECT * FROM PAL_ADDITIVE_MODEL_PREDICT_DATA_TBL;
  lt_pparam = SELECT * FROM PAL_ADDITIVE_MODEL_PREDICT_PARAMETER_TBL;
  CALL _SYS_AFL.PAL_ADDITIVE_MODEL_EXPLAIN(:lt_pdata, :lt_model, :lt_pparam,
lt_result, lt_decomp);
  INSERT INTO PAL_ADDITIVE_MODEL_PREDICT_DECOMP SELECT * FROM :lt_decomp;
END;
SELECT * FROM PAL_ADDITIVE_MODEL_PREDICT_DECOMP;
```

**Expected Result**

| | ts | trend | seasonal | holiday | exogenous |
|---|---|---|---|---|---|
| 1 | Mar 9, 2008 | 7.178767473566212 | {"MONTHLY":0.07113698371810685} | {} | {} |
| 2 | Mar 10, 2008 | 7.1291324829916825 | {"MONTHLY":-0.11831428899667505} | {} | {} |
| 3 | Mar 11, 2008 | 7.079497492417153 | {"MONTHLY":-0.12868087381938846} | {} | {} |
| 4 | Mar 12, 2008 | 7.029862501842625 | {"MONTHLY":-0.0343709868589206} | {} | {} |
| 5 | Mar 13, 2008 | 6.980227511268096 | {"MONTHLY":0.06863828002983939} | {} | {} |
| 6 | Mar 14, 2008 | 6.930592520693566 | {"MONTHLY":0.10828035662136962} | {} | {} |
| 7 | Mar 15, 2008 | 6.880957530119037 | {"MONTHLY":0.04616084345776677} | {} | {} |
| 8 | Mar 16, 2008 | 6.831322539544507 | {"MONTHLY":-0.11708611453009177} | {} | {} |
| 9 | Mar 17, 2008 | 6.781687548969977 | {"MONTHLY":-0.32615977958123545} | {} | {} |
| 10 | Mar 18, 2008 | 6.732052558395448 | {"MONTHLY":-0.4737375645476394} | {} | {} |
| 11 | Mar 19, 2008 | 6.682417567820919 | {"MONTHLY":-0.45510794064236045} | {} | {} |
| 12 | Mar 20, 2008 | 6.63278257724639 | {"MONTHLY":-0.2555468639448672} | {} | {} |
| 13 | Mar 21, 2008 | 6.583147586671861 | {"MONTHLY":0.00559234623675375... | {} | {} |
| 14 | Mar 22, 2008 | 6.533512596097332 | {"MONTHLY":0.13765419644572236} | {} | {} |
| 15 | Mar 23, 2008 | 6.483877605522801 | {"MONTHLY":0.02232929881361553... | {} | {} |
| 16 | Mar 24, 2008 | 6.434242614948273 | {"MONTHLY":-0.275906987544626} | {} | {} |

## 5.5.2 ARIMA

The auto regressive integrated moving average (ARIMA) algorithm is famous in econometrics, statistics and time series analysis.

A series $y_t$ can be expressed as an ARIMA model, that is,

$\varphi(B)(1-B)^d y_t=\theta(B)\varepsilon_t$

where $\varepsilon_t \sim WN(0,\sigma^2)$, is white noise. And

$\varphi(B)=1-\varphi_1 B-...-\varphi_p B^p$

$\theta(B)=1+\theta_1 B+...+\theta_q B^q$

where $B^k y_t=y_{t-k}$ is the backshift notation, $\varphi_i$ and $\theta_j$ are some constants. $\varphi(B)$ is called AR part, $(1-B)^d$ is the $d$-degree integrated part, and $\theta(B)$ is the MA part. Such an ARIMA model can be written as ARIMA (p, d, q). If the integrated part is absent, it yields to $\varphi(B)y_t=\theta(B)\varepsilon_t$, which is the so-called ARMA (p, q) model. Taking the seasonality factor into account, a seasonal ARIMA (SARIMA) model is obtained,

$\varphi(B)\Phi(B^s)(1-B)^d(1-B^s)^D y_t=\theta(B)\Theta(B^s)\varepsilon_t$

where $B^s y_t=y_{t-s}$ is the seasonal backshift notation with period $s$. Still, $y_t$ is usually determined by some covariates, or regressors, say $x_t$, which is an $r\times 1$ vector. An SARIMAX model is generated.

$\varphi(B)\Phi(B^s)\{(1-B)^d(1-B^s)^D y_t-\mu-x^T\beta\}=\theta(B)\Theta(B^s)\varepsilon_t$

where $\mu$ is a scalar constant, like intercept, and $\beta$ is an $r$-dimensional vector, which are the coefficients to be determined. In conclusion, the general model can be notated as SARIMAX (p, d, q) (P, D, Q)s.

A basis procedure to estimate an SARIMAX model is as follows:

1. Difference the series, and get SARMAX model.
2. Preliminary estimation, to get initial $\mu,\beta$ (least square), and ARMA parameters (Yule-Walker, Burg, Innovations, Hannan-Rissanen, and so on.).
3. Do some transformation to guarantee stability, invertibility, and causality.
4. Numeric optimization for $\mu,\beta,\varphi,\theta$.
5. Inverse transformation to original space.

In PAL, two methods are provided for numeric optimization: the conditional sum of squares (CSS) and maximum likelihood estimation (MLE), the former of which is faster but much less precise. Therefore, MLE is highly recommended. MLE, in detail, employs Kalman filter to calculate the likelihood function. Both of the two methods are realized with the BFGS algorithm, and the gradient is calculated via finite-difference method.

As for forecast based on the SARIMAX model, two approaches are provided. The first is to calculate future series straightforwardly via the SARIMAX formula, and the second is to apply innovations algorithm, which requires more original information to be stored.

As for the explanation part in ARIMA_EXPLAIN, forecast values are divided into two main parts, the signal and the external. Signal part comes from ARIMA model itself, which can be further decomposed into trend, seasonal, transitory and irregular components. External part, on the other hand, captures the Shapely Value of each exogenous data by LinearSHAP.

Note that for the sake of interoperability, the ARIMA model output table still takes key-value style, but has some adaptation of the key names. The new key names are listed as follows:

| Key name | Description |
|---|---|
| p | Number of AR parameters |
| AR | Coefficients of AR parameters |
| d | Degree of first differencing |
| q | Number of MA parameters |
| MA | Coefficients of MA parameters |

| Key name | Description |
|---|---|
| s | The seasonal period |
| P | Number of SAR parameters |
| SAR | Coefficients of SAR parameters |
| D | Degree of seasonal differencing |
| Q | Number of SMA parameters |
| SMA | Coefficients of SMA parameters |
| mu | The constant part |
| regressor | The covariates' names |
| beta | The coefficients of covariates |
| sigma^2 | The variance |
| log-likelihood | Log likelihood of the fitted series |
| AIC | Akaike information criterion |
| AICc | Akaike information criterion correction |
| BIC | Bayesian information criterion |
| dy(n-p:n-1)_ aux | An indicator to the number of rows dy(n-p:n-1) occupies |
| dy(n-p:n-1)_i | ith row of last p elements of the differenced series |
| dy_aux | An indicator to the number of rows dy occupies |
| dy_i | ith row of the whole differenced series |
| y(n-d:n-1)_aux | An indicator to the number of rows y(n-d:n-1)_occupies |
| y(n-d:n-1)_i | ith row of last d elements of original series |
| epsilon(n-q:n-1)_aux | An indicator to the number of rows epsilon(n-q:n-1)_occupies |
| epsilon(n-q:n-1)_i | ith row of last q elements of $\varepsilon_t$ |
| bgdata_i | ith row of background data for ARIMA_EXPLAIN |

To ensure backward compatibility, the forecast algorithm can parse the old ARIMA model generated by the previous SAP HANA PAL releases as well.

## Prerequisite

- The input data must not have missing value.
- The ID column must not have any identical value.
- For ARIMAX, the valid covariates' name in training must also be present in forecast.

## _SYS_AFL.PAL_ARIMA

This function generates ARIMA models with given orders.

### Overview of all Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<MODEL table>` | OUT |
| 4 | `<FIT table>` | OUT |

### Input Table

| Table | Column | Reference for Output Table | Data Type | Description |
|---|---|---|---|---|
| DATA | 1st column | ID | INTEGER | Time stamp.<br>No identical value is allowed. |
| | 2nd column | | INTEGER, DOUBLE, or DECIMAL(p, s) | Raw data |
| | (For ARIMAX only) Other columns | | INTEGER, DOUBLE, DECIMAL (p, s), VARCHAR, or NVARCHAR | External (Intervention) data |

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| P | INTEGER | 0 | Value of the auto regression order | |
| D | INTEGER | 0 | Value of the differentiation order | |
| Q | INTEGER | 0 | Value of the moving average order | |
| SEASONAL_PERIOD | INTEGER | 0 | Value of the seasonal period | |
| SEASONAL_P | INTEGER | 0 | Value of the auto regression order for seasonal part | Valid only when SEASONAL_PERIOD is larger than 1. |
| SEASONAL_D | INTEGER | 0 | Value of the differentiation order for seasonal part | Valid only when SEASONAL_PERIOD is larger than 1. |
| SEASONAL_Q | INTEGER | 0 | Value of the moving average order for seasonal part | Valid only when SEASONAL_PERIOD is larger than 1. |
| METHOD | INTEGER | 2 | The object function for numeric optimization:<br>• 0: Uses the conditional sum of squares (CSS).<br>• 1: Uses the maximized likelihood estimation (MLE).<br>• 2: Firstly uses CSS to approximate starting values, and then uses MLE to fit (CSS-MLE). | |
| INCLUDE_MEAN | INTEGER | 0 when D + SEASONAL_D is 1;<br>1 when D + SEASONAL_D is 0 | Controls whether the ARIMA model includes a constant part:<br>• 0: Excludes<br>• 1: Includes | Valid only when D + SEASONAL_D is not larger than 1. That is, the constant part will be excluded in all other cases. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| FORECAST_METHOD | INTEGER | 1 | Store information for the subsequent forecast method.<br>• 0: Formula forecast<br>• 1: Innovations algorithm | |
| OUTPUT_FITTED | INTEGER | 1 | If output fitted and residuals<br>• 0: not output<br>• 1: output | |
| THREAD_RATIO | DOUBLE | -1 | The ratio of available threads.<br>• 0: single thread<br>• 0~1: percentage<br>• Others: heuristically determined | |
| DEPENDENT_VARIABLE | VARCHAR or NVARCHAR | The second column's name | The dependent variable, i.e. time series. | Valid only for ARIMAX; cannot be the first column name (time stamp column). |
| EXCLUDED_FEATURE | VARCHAR or NVARCHAR | No default value | Excludes the indicated feature column. | Valid only for ARIMAX; cannot be the first column name (time stamp column) and the column name of dependent variable. |
| CATEGORICAL_VARIABLE | VARCHAR or NVARCHAR | Detected from input data | Indicate which variables are treated as categorical. The default behavior is:<br>• string: categorical<br>• integer and double: continuous | Valid only for integer or string column. |
| SOLVER | INTEGER | 1 | Optimization solver:<br>• 0: BFGS<br>• 1: L-BFGS<br>• 2: L-BFGS-B | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| BACKGROUND_SIZE | INTEGER | 0 | Number of background data rendered in MODEL table for ARIMA_EXPLAIN | |

### Output Tables

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| MODEL | 1st column | NVARCHAR(100) | KEY | Name |
| | 2nd column | NVARCHAR(5000) | VALUE | Value |
| FIT | 1st column | ID (in input table) column name | ID (in input table) column name | Time stamp |
| | 2nd column | DOUBLE | FITTED | Fitted values |
| | 3rd column | DOUBLE | RESIDUALS | Residuals values |

> **Note**
>
> The first d + s * D values of FITTED and RESIDUALS are absent.

### Examples

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

### Example 1: ARIMA

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_ARIMA_DATA_TBL;
CREATE COLUMN TABLE PAL_ARIMA_DATA_TBL ("TIMESTAMP" INTEGER, "Y" DOUBLE);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (1, -0.636126431 );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (2, 3.092508651  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (3, -0.73733556  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (4, -3.142190983 );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (5, 2.088819813  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (6, 3.179302734  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (7, -0.871376102 );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (8, -3.475633275 );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (9, 1.779244219  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (10, 3.609159416 );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (11, 0.082170143 );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (12, -4.42439631 );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (13, 0.499210261 );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (14, 4.514017351 );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (15, -0.320607187);
```

```
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (16, -3.70219307 );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (17, 0.100228116 );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (18, 4.553625233 );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (19, 0.261489853 );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (20, -4.474116429);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (21, -0.372574233);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (22, 2.872305281 );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (23, 1.289850031 );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (24, -3.662763983);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (25, -0.168962933);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (26, 4.018728154 );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (27, -1.306247869);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (28, -2.182690245);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (29, -0.845114493);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (30, 0.99806763  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (31, -0.641201109);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (32, -2.640777923);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (33, 1.493840358 );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (34, 4.326449202 );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (35, -0.653797151);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (36, -4.165384227);
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL ( "NAME" VARCHAR (50),"INT_VALUE"
INTEGER,"DOUBLE_VALUE" DOUBLE,"STRING_VALUE" VARCHAR (100));
INSERT INTO PAL_PARAMETER_TBL VALUES ('D', 0, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('Q', 1, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('SEASONAL_PERIOD', 4, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('SEASONAL_P', 1, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('METHOD', 1, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 1.0, NULL);
DROP TABLE PAL_ARIMA_MODEL_TBL;  -- for the following forecast
CREATE COLUMN TABLE PAL_ARIMA_MODEL_TBL ("KEY" NVARCHAR(100), "VALUE"
NVARCHAR(5000));
do begin
    lt_data = select * from PAL_ARIMA_DATA_TBL;
    lt_control = select * from PAL_PARAMETER_TBL;
    _SYS_AFL.PAL_ARIMA(:lt_data, :lt_control, lt_model, lt_fit);
    INSERT INTO PAL_ARIMA_MODEL_TBL SELECT * FROM :lt_model;
    SELECT * FROM PAL_ARIMA_MODEL_TBL;
    SELECT* FROM :lt_fit;
end;
```

### Expected Result

MODEL:

| | KEY | VALUE |
|---|---|---|
| 1 | p | 0 |
| 2 | AR | |
| 3 | d | 0 |
| 4 | q | 1 |
| 5 | MA | -0.141073 |
| 6 | s | 4 |
| 7 | P | 1 |
| 8 | SAR | 0.869755 |
| 9 | D | 0 |
| 10 | Q | 0 |
| 11 | SMA | |
| 12 | mu | 0.0233736 |
| 13 | sigm... | 1.69634 |
| 14 | log-l... | -63.4291 |
| 15 | AIC | 134.858 |
| 16 | AICc | 136.148 |
| 17 | BIC | 141.192 |
| 18 | dy(n... | |
| 19 | dy_aux | 0 |
| 20 | dy_0 | -0.636126;3.0925;-0... |
| 21 | y(n-... | |
| 22 | epsil... | 0 |
| 23 | epsil... | -1.81085 |

FIT:

| | TIMESTAMP | FITTED | RESIDUALS |
|---|---|---|---|
| 1 | 1 | 0.02337363393571401 | -0.659500064935714 |
| 2 | 2 | 0.1145959096328033 | 2.977912741367197 |
| 3 | 3 | -0.3965667978216644 | -0.3407687621783... |
| 4 | 4 | 0.10108234304353525 | -3.243273326043535 |
| 5 | 5 | -0.4370271728760655 | 2.52584698587606... |
| 6 | 6 | 2.3416997005921427 | 0.83760303340785... |
| 7 | 7 | -0.7563859661945617 | -0.1149901358054... |
| 8 | 8 | -2.713671648833266 | -0.7619616261667... |
| 9 | 9 | 1.927299099731048 | -0.148054880731048 |
| 10 | 10 | 2.7891468657530516 | 0.82001255024694... |
| 11 | 11 | -0.8705216109242832 | 0.95269175392428... |
| 12 | 12 | -3.1543060598366255 | -1.2700902501633... |
| 13 | 13 | 1.7297273421632222 | -1.2305170811632... |
| 14 | 14 | 3.3157234216186033 | 1.19829392938139... |
| 15 | 15 | -0.09453487192880... | -0.2260723150711... |
| 16 | 16 | -3.8132059656136232 | 0.11101289561362... |
| 17 | 17 | 0.42157421494315916 | -0.3213460989431... |
| 18 | 18 | 3.9744689700312614 | 0.57915626296873... |
| 19 | 19 | -0.3575089689968078 | 0.61899882199680... |
| 20 | 20 | -3.3042825744440973 | -1.169833854555903 |
| 21 | 21 | 0.2552503819632394 | -0.6278246149632... |
| 22 | 22 | 4.052154030824162 | -1.1798487498241... |
| 23 | 23 | 0.39692149598922855 | 0.89292853501077... |
| 24 | 24 | -4.014311280590275 | 0.35154729759027... |
| 25 | 25 | -0.37059808018197... | 0.20163514718197... |
| 26 | 26 | 2.472802273513204 | 1.545925880486796 |
| 27 | 27 | 0.90680980972932 | -2.21305767872932 |
| 28 | 28 | -2.870461793965125 | 0.687771548965125 |

### Example 2: ARIMAX

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_ARIMA_DATA_TBL;
CREATE COLUMN TABLE PAL_ARIMA_DATA_TBL (
    "TIMESTAMP" INTEGER,
    "Y" DOUBLE,
    "X" DOUBLE
    );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(1, 1.2, 0.8);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(2, 1.34845613096197, 1.2);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(3, 1.32261090809898, 1.34845613096197);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(4, 1.38095306748554, 1.32261090809898);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(5, 1.54066648969168, 1.38095306748554);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(6, 1.50920806756785, 1.54066648969168);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(7, 1.48461408893443, 1.50920806756785);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(8, 1.43784887380224, 1.48461408893443);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(9, 1.64251548718992, 1.43784887380224);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(10, 1.74292337447476, 1.64251548718992);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(11, 1.91137546943257, 1.74292337447476);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(12, 2.07735796176367, 1.91137546943257);
```

```
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(13, 2.01741246166924, 2.07735796176367);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(14, 1.87176938196573, 2.01741246166924);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(15, 1.83354723357744, 1.87176938196573);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(16, 1.66104978144571, 1.83354723357744);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(17, 1.65115984070812, 1.66104978144571);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(18, 1.69470966154593, 1.65115984070812);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(19, 1.70459802935728, 1.69470966154593);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(20, 1.61246059980916, 1.70459802935728);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(21, 1.53949706614636, 1.61246059980916);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(22, 1.59231354902055, 1.53949706614636);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(23, 1.81741927705578, 1.59231354902055);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(24, 1.80224252773564, 1.81741927705578);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(25, 1.81881576781466, 1.80224252773564);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(26, 1.78089755157948, 1.81881576781466);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(27, 1.61473635574416, 1.78089755157948);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(28, 1.42002147867225, 1.61473635574416);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(29, 1.49971641345022, 1.42002147867225);
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL ("NAME" VARCHAR (50),"INT_VALUE"
INTEGER,"DOUBLE_VALUE" DOUBLE,"STRING_VALUE" VARCHAR (100));
INSERT INTO PAL_PARAMETER_TBL VALUES ('P', 1, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('Q', 1, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('D', 0, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('METHOD', 1, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('DEPENDENT_VARIABLE', NULL, NULL, 'Y');
DROP TABLE PAL_ARIMA_MODEL_TBL;  -- for the forecast followed
CREATE COLUMN TABLE PAL_ARIMA_MODEL_TBL ("KEY" NVARCHAR(100), "VALUE"
NVARCHAR(5000));
DROP TABLE PAL_ARIMA_MODEL_TBL;  -- for the following forecast
CREATE COLUMN TABLE PAL_ARIMA_MODEL_TBL ("KEY" NVARCHAR(100), "VALUE"
NVARCHAR(5000));
do begin
    lt_data = select * from PAL_ARIMA_DATA_TBL;
    lt_control = select * from PAL_PARAMETER_TBL;
    _SYS_AFL.PAL_ARIMA(:lt_data, :lt_control, lt_model, lt_fit);
    INSERT INTO PAL_ARIMA_MODEL_TBL SELECT * FROM :lt_model;
    SELECT * FROM PAL_ARIMA_MODEL_TBL;
    SELECT* FROM :lt_fit;
end;
```

**Expected Result**

MODEL:

| | KEY | VALUE |
|---|---|---|
| 1 | p | 1 |
| 2 | AR | 0.302207 |
| 3 | d | 0 |
| 4 | q | 1 |
| 5 | MA | 0.291575 |
| 6 | mu | 0.732733 |
| 7 | regressor | X |
| 8 | beta | 0.562037 |
| 9 | sigma^2 | 0.00874419 |
| 10 | log-likelihood | 27.3948 |
| 11 | AIC | -44.7896 |
| 12 | AICc | -42.1809 |
| 13 | BIC | -37.9531 |
| 14 | dy(n-p:n-1)_aux | |
| 15 | dy_aux | 0 |
| 16 | dy_0 | 0.75037;0.674011;0.564728;... |
| 17 | x(n-d:n-1)_aux | |
| 18 | y(n-d:n-1)_aux | |
| 19 | epsilon(n-q:n-1)_aux | 0 |
| 20 | epsilon(n-q:n-1)_0 | 0.0788643 |

FIT:

| | TIMESTAMP | FITTED | RESIDUALS |
|---|---|---|---|
| 1 | 1 | 1.1823630950401558 | 0.017636904959844157 |
| 2 | 2 | 1.4162128404184604 | -0.06775670945649033 |
| 3 | 3 | 1.45357203158135 | -0.13096112348237007 |
| 4 | 4 | 1.387207623215808 | -0.0062545557302681... |
| 5 | 5 | 1.478305828440324 | 0.06236066125135585 |
| 6 | 6 | 1.6264337161635836 | -0.11722564859573355 |
| 7 | 7 | 1.5197556877881384 | -0.03514159885370849 |
| 8 | 8 | 1.5277773835726225 | -0.08992850977038251 |
| 9 | 9 | 1.4755636191687576 | 0.16695186802116235 |
| 10 | 10 | 1.7352887795937637 | 0.007634594880996395 |
| 11 | 11 | 1.7408497744700782 | 0.1705256949624917 |
| 12 | 12 | 1.9168741505246254 | 0.16048381123904482 |
| 13 | 13 | 2.0287837664677473 | -0.0113713047985074... |
| 14 | 14 | 1.89867504344609 | -0.0269056614803600... |
| 15 | 15 | 1.7784562071041177 | 0.05509102647332238 |
| 16 | 16 | 1.7940688775736529 | -0.1330190961279429 |
| 17 | 17 | 1.5966327142158172 | 0.054527126492302916 |
| 18 | 18 | 1.6720682449909847 | 0.022641416554945268 |
| 19 | 19 | 1.7020886871988727 | 0.002509342158407247 |
| 20 | 20 | 1.697367622024235 | -0.08490702221507515 |
| 21 | 21 | 1.5905703293274271 | -0.0510732631810671... |
| 22 | 22 | 1.553026786876798 | 0.039286762143752034 |
| 23 | 23 | 1.6374128951632905 | 0.18000638189248963 |
| 24 | 24 | 1.8640187077447454 | -0.06177618000910548 |
| 25 | 25 | 1.7421698324956858 | 0.07664593531897412 |
| 26 | 26 | 1.7994314326949454 | -0.01853388111546537 |
| 27 | 27 | 1.7360937385398374 | -0.12135738279567725 |
| 28 | 28 | 1.5689494972231397 | -0.14892801855088972 |
| 29 | 29 | 1.4208520761195238 | 0.0788643373306964 |

## _SYS_AFL.PAL_ARIMA_FORECAST

This function makes time series forecast based on the estimated ARIMA model.

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |

| Position | Table Name | Parameter Type |
|---|---|---|
| 2 | `<MODEL table>` | IN |
| 3 | `<PARAMETER table>` | IN |
| 4 | `<FORECAST table>` | OUT |

> **Note**
>
> If the first input table (data table) is absent, it means that no covariate presents, and only ARIMA forecast is carried out. Otherwise, ARIMAX forecast is carried out with the number of forecasters being the length of the data table.

### Input Tables

| Table | Column | Reference for Output Table | Data Type | Description |
|---|---|---|---|---|
| DATA | 1st column | ID | INTEGER | Time stamp |
| | Other columns | | INTEGER, DOUBLE, or DECIMAL (p, s) | External (Intervention) data |
| MODEL | 1st column | | VARCHAR or NVARCHAR | Model key |
| | 2nd column | | VARCHAR or NVARCHAR | Model value |

### Parameter Table

Mandatory Parameters

None.

Optional Parameter

The following parameter is optional. If it is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| FORECAST_METHOD | INTEGER | 1 | Specifies the forecast method:<br>• 0: Formula method<br>• 1: Innovations algorithm<br>• 2: Truncation algorithm | if FORECAST_METHOD in PAL_ARIMA is set to 0, the formula method is adopted. Truncation algorithm is much faster than innovations algorithm when the AR representation of ARIMA model can be truncated to finite order. |
| FORECAST_LENGTH | INTEGER | 1 | Number of points to forecast. | Valid only when the first input table is absent. In ARIMAX, the forecast length is the same as the length of the input data. |

### Output Table

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| RESULT | 1st column | INTEGER | ID (in input table) column name | Time stamp |
| | 2nd column | DOUBLE | FORECAST | Forecast value |
| | 3rd column | DOUBLE | SE | Standard error |
| | 4th column | DOUBLE | LO80 | Low 80% value |
| | 5th column | DOUBLE | HI80 | High 80% value |
| | 6th column | DOUBLE | LO95 | Low 95% value |
| | 7th column | DOUBLE | HI95 | High 95% value |

### Examples

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

#### Example 1: ARIMA forecast

```
SET SCHEMA DM_PAL;
```

```
DROP TABLE PAL_ARIMA_DATA_TBL;
CREATE COLUMN TABLE PAL_ARIMA_DATA_TBL ("TIMESTAMP" INTEGER, "Y" DOUBLE);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL ( "NAME" VARCHAR
(50),"INT_VALUE" INTEGER,"DOUBLE_VALUE" DOUBLE,"STRING_VALUE" VARCHAR (100));
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FORECAST_METHOD', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FORECAST_LENGTH', 10, NULL, NULL);
CALL _SYS_AFL.PAL_ARIMA_FORECAST (PAL_ARIMA_DATA_TBL, PAL_ARIMA_MODEL_TBL,
"#PAL_PARAMETER_TBL", ?);
```

### Expected Result

RESULT:

| | TIMESTAMP | FORECAST | SE | LO80 | HI80 | LO95 | HI95 |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 1.5577828082727392 | 1.3024361788586802 | -0.1113568816159602 5 | 3.2269224981614384 | -0.9949452145878352 | 4.110510831133314 |
| 2 | 1 | 3.7659871167319996 | 1.315332606390884 | 2.0803199843465494 | 5.45165424911745 | 1.1879825601796972 | 6.343991673284302 |
| 3 | 2 | -0.565598915202999 | 1.315332606390884 | -2.25126604758845 | 1.12006821718245 | -3.143603471755302 | 2.012405641349302 |
| 4 | 3 | -3.619815787367999 | 1.315332606390884 | -5.30548291975345 | -1.934148654982549 | -6.197820343920302 | -1.04181123081569 |
| 5 | 4 | 1.357933680941256 | 1.7358964725800452 | -0.8667079152866461 | 3.582575277169158 | -2.0443609130426195 | 4.760228274925132 |
| 6 | 5 | 3.2785304192452402 | 1.7432369362365585 | 1.0444816371374062 | 5.512579201353074 | -0.13815121924871 | 6.695212057739191 |
| 7 | 6 | -0.488888189960385 | 1.7432369362365585 | -2.722936972068219 | 1.7451605921474491 | -3.905569828454335 | 2.927793448533565 |
| 8 | 7 | -3.145308585610254 | 1.7432369362365585 | -5.379357367718089 | -0.911259803502420 | -6.561990224104205 | 0.271373052883695 |
| 9 | 8 | 1.1841139031990622 | 2.002401035222135 | -1.3820671482919353 | 3.7502949546900597 | -2.7405200393990543 | 5.108747845797179 |
| 10 | 9 | 2.8545625193226436 | 2.007219239377763 | 0.2822066886595924 | 5.426918349985694 | -1.0795149299651543 | 6.788639968610441 |

### Example 2: ARIMAX forecast

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_ARIMA_DATA_TBL;
CREATE COLUMN TABLE PAL_ARIMA_DATA_TBL ("TIMESTAMP" INTEGER, "X" DOUBLE);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(1, 0.8);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(2, 1.2);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(3, 1.34845613096197);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(4, 1.32261090809898);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(5, 1.38095306748554);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(6, 1.54066648969168);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(7, 1.50920806756785);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(8, 1.48461408893443);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(9, 1.43784887380224);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(10, 1.64251548718992);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL ( "NAME" VARCHAR
(50),"INT_VALUE" INTEGER,"DOUBLE_VALUE" DOUBLE,"STRING_VALUE" VARCHAR (100));
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FORECAST_METHOD', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FORECAST_LENGTH', 5, NULL, NULL); --
takes not effect here
CALL _SYS_AFL.PAL_ARIMA_FORECAST (PAL_ARIMA_DATA_TBL, PAL_ARIMA_MODEL_TBL,
"#PAL_PARAMETER_TBL", ?);
```

### Expected Result

RESULT:

| | TIMESTAMP | FORECAST | SE | LO80 | HI80 | LO95 | HI95 |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 1.1959520991296921 | 0.0935103737560705 | 1.0761136926218524 | 1.3157905056375319 | 1.0126751329412491 | 1.3792290653181352 |
| 2 | 1 | 1.4112842417634868 | 0.10875288884942748 | 1.2719117595527254 | 1.5506567239742481 | 1.1981324947226075 | 1.624435988804366 |
| 3 | 2 | 1.4918563548062909 | 0.11003980386482552 | 1.350834624083716 | 1.6328780855288658 | 1.276182300664172 | 1.7075304089484098 |
| 4 | 3 | 1.4764643409976097 | 0.11015658737468524 | 1.3352929461344072 | 1.6176357358608122 | 1.260561395380372 | 1.6923672866148474 |
| 5 | 4 | 1.5089930691914968 | 0.11016724693469745 | 1.3678080135478414 | 1.6501781248351521 | 1.2930692312203793 | 1.7249169071626143 |
| 6 | 5 | 1.5986788270306331 | 0.11016822041100977 | 1.4574925238264627 | 1.7398651302348036 | 1.3827530810809887 | 1.8146045729802776 |
| 7 | 6 | 1.5809741268219155 | 0.11016830931726818 | 1.4397877096797518 | 1.7221605439640792 | 1.3650482066192053 | 1.7969000470246257 |
| 8 | 7 | 1.567144177194721 | 0.11016831743699057 | 1.4259577496467108 | 1.7083306047427311 | 1.3512182410776472 | 1.7830701133117948 |
| 9 | 8 | 1.5408582129374562 | 0.11016831817855724 | 1.3996717844390898 | 1.6820446414358226 | 1.3249322753669384 | 1.756784150507974 |
| 10 | 9 | 1.6558877625960542 | 0.11016831824628384 | 1.5147013340108926 | 1.7970741911812158 | 1.4399618248927948 | 1.8718137002993136 |

## MASSIVE INTERFACE

When there exist tons of data to be analyzed by ARIMA, it is troublesome to call the functions above again and again, and the MASSIVE interface is used to tackle that obstacle by placing all the time series in a single `<DATA table>` and identifying each series by a new column named "Group ID". Afterwards, ARIMA is applied to each time series automatically inside this MASSIVE interface. Particularly, after the incorporation of "Group ID", `<PARAMETER table>` in the MASSIVE interface is extended into 5 columns as shown below.

| Table | Column | Data Type | Description |
|---|---|---|---|
| PARAMETER | 1st column | INTEGER, VRACHAR, or NVARCHAR | Group ID. |
| | 2nd column | VARCHAR or NVARCHAR | Parameter name. |
| | 3rd column | INTEGER | Integer value. |
| | 4th column | DOUBLE | Double value. |
| | 5th column | VARCHAR or NVARCHAR | String value. |

## _SYS_AFL.PAL_MASSIVE_ARIMA

This function is the MASSIVE version of _SYS_AFL.PAL_ARIMA, by which multiple time series can be analyzed in a single call.

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<MODEL table>` | OUT |
| 4 | `<FIT table>` | OUT |
| 5 | `<ERROR_MSG table>` | OUT |

### Input Table

| Table | Column | Data Type | Description |
|---|---|---|---|
| DATA | 1st column | INTEGER, VRACHAR or NVARCHAR | Group ID. |
| | 2nd column | INTEGER | Time stamp. Reference for Output Table is "ID" |
| | 3rd column | INTEGER, DOUBLE, or DECIMAL(p,s) | Raw data. |

| Table | Column | Data Type | Description |
|---|---|---|---|
| | (For ARIMAX only) Other columns | INTEGER, DOUBLE, DECIMAL (p,s), VARCHAR, or NVARCHAR | External (Intervention) data. |

### Parameter Table

Same as _SYS_AFL.PAL_ARIMA.

### Output Table

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| MODEL | 1st column | VARCHAR | GROUP_ID | Group ID. |
| | 2nd column | VARCHAR or NVARCHAR | ROW_INDEX | Model key. |
| | 3rd column | VARCHAR or NVARCHAR | MODEL_CONTENT | Model value. |
| FIT | 1st column | VARCHAR | GROUP_ID | Group ID. |
| | 2nd column | INTEGER | ID (in input table) column name | Time stamp. |
| | 3rd column | DOUBLE | FITTED | Fitted values. |
| | 4th column | DOUBLE | RESIDUALS | Residuals values. |
| ERROR_MSG | 1st column | VARCHAR | GROUP_ID | Group ID. |
| | 2nd column | VARCHAR | ERROR_TIMESTAMP | Time stamp when the error occured. |
| | 3rd column | INTEGER | ERROR_CODE | Error code |
| | 4th column | VARCHAR | ERROR_MESSAGE | Error message. |

### Examples

Assume that:

- DM_PAL is a schema belonging to USER1;
- USER1 has been assigned the AFL__SYS_AFL_AFLPAL_EXECUTE or AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION role.

### Example 1: MASSIVE ARIMA

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_MASSIVE_ARIMA_DATA_TBL;
CREATE COLUMN TABLE PAL_MASSIVE_ARIMA_DATA_TBL (
    GROUP_ID  NVARCHAR(40),
    TIMESTAMP INTEGER,
    Y         DOUBLE
);
DROP TABLE PAL_MASSIVE_ARIMA_PARAM_TBL;
CREATE COLUMN TABLE PAL_MASSIVE_ARIMA_PARAM_TBL (
    GROUP_ID   NVARCHAR(40),
    NAME       NVARCHAR(100),
    INTARGS    INTEGER,
    DOUBLEARGS DOUBLE,
    STRINGARGS NVARCHAR(100)
```

```
);
DROP TABLE PAL_MASSIVE_ARIMA_MODEL_TBL;
CREATE COLUMN TABLE PAL_MASSIVE_ARIMA_MODEL_TBL (
    GROUP_ID NVARCHAR(40),
    KEY      NVARCHAR(100),
    VALUE    NVARCHAR(5000)
);
DROP TABLE PAL_MASSIVE_ARIMA_FIT_TBL;
CREATE COLUMN TABLE PAL_MASSIVE_ARIMA_FIT_TBL (
    GROUP_ID   NVARCHAR(40),
    TIMESTAMP INTEGER,
    FITTED    DOUBLE,
    RESIDUAL  DOUBLE
);
DROP TABLE PAL_MASSIVE_ARIMA_ERROR_TBL;
CREATE COLUMN TABLE PAL_MASSIVE_ARIMA_ERROR_TBL (
    GROUP_ID        NVARCHAR(40),
    ERROR_TIMESTAMP NVARCHAR(100),
    ERRORCODE       INTEGER,
    ERROR_MESSAGE   NVARCHAR(200)
);
INSERT INTO PAL_MASSIVE_ARIMA_DATA_TBL VALUES ('GROUP_A', 1, -0.636126431 );
INSERT INTO PAL_MASSIVE_ARIMA_DATA_TBL VALUES ('GROUP_A', 2, 3.092508651  );
INSERT INTO PAL_MASSIVE_ARIMA_DATA_TBL VALUES ('GROUP_A', 3, -0.73733556  );
INSERT INTO PAL_MASSIVE_ARIMA_DATA_TBL VALUES ('GROUP_A', 4, -3.142190983 );
INSERT INTO PAL_MASSIVE_ARIMA_DATA_TBL VALUES ('GROUP_A', 5, 2.088819813  );
INSERT INTO PAL_MASSIVE_ARIMA_DATA_TBL VALUES ('GROUP_A', 6, 3.179302734  );
INSERT INTO PAL_MASSIVE_ARIMA_DATA_TBL VALUES ('GROUP_A', 7, -0.871376102 );
INSERT INTO PAL_MASSIVE_ARIMA_DATA_TBL VALUES ('GROUP_A', 8, -3.475633275 );
INSERT INTO PAL_MASSIVE_ARIMA_DATA_TBL VALUES ('GROUP_A', 9, 1.779244219  );
INSERT INTO PAL_MASSIVE_ARIMA_DATA_TBL VALUES ('GROUP_A', 10, 3.609159416 );
INSERT INTO PAL_MASSIVE_ARIMA_DATA_TBL VALUES ('GROUP_A', 11, 0.082170143 );
INSERT INTO PAL_MASSIVE_ARIMA_DATA_TBL VALUES ('GROUP_A', 12, -4.42439631 );
INSERT INTO PAL_MASSIVE_ARIMA_DATA_TBL VALUES ('GROUP_A', 13, 0.499210261 );
INSERT INTO PAL_MASSIVE_ARIMA_DATA_TBL VALUES ('GROUP_A', 14, 4.514017351 );
INSERT INTO PAL_MASSIVE_ARIMA_DATA_TBL VALUES ('GROUP_A', 15, -0.320607187);
INSERT INTO PAL_MASSIVE_ARIMA_DATA_TBL VALUES ('GROUP_A', 16, -3.70219307 );
INSERT INTO PAL_MASSIVE_ARIMA_DATA_TBL VALUES ('GROUP_B', 1, -0.636126431 );
INSERT INTO PAL_MASSIVE_ARIMA_DATA_TBL VALUES ('GROUP_B', 2, 3.092508651  );
INSERT INTO PAL_MASSIVE_ARIMA_DATA_TBL VALUES ('GROUP_B', 3, -0.73733556  );
INSERT INTO PAL_MASSIVE_ARIMA_DATA_TBL VALUES ('GROUP_B', 4, -3.142190983 );
INSERT INTO PAL_MASSIVE_ARIMA_DATA_TBL VALUES ('GROUP_B', 5, 2.088819813  );
INSERT INTO PAL_MASSIVE_ARIMA_DATA_TBL VALUES ('GROUP_B', 6, 3.179302734  );
INSERT INTO PAL_MASSIVE_ARIMA_DATA_TBL VALUES ('GROUP_B', 7, -0.871376102 );
INSERT INTO PAL_MASSIVE_ARIMA_DATA_TBL VALUES ('GROUP_B', 8, -3.475633275 );
INSERT INTO PAL_MASSIVE_ARIMA_DATA_TBL VALUES ('GROUP_B', 9, 1.779244219  );
INSERT INTO PAL_MASSIVE_ARIMA_DATA_TBL VALUES ('GROUP_B', 10, 3.609159416 );
INSERT INTO PAL_MASSIVE_ARIMA_DATA_TBL VALUES ('GROUP_B', 11, 0.082170143 );
INSERT INTO PAL_MASSIVE_ARIMA_DATA_TBL VALUES ('GROUP_B', 12, -4.42439631 );
INSERT INTO PAL_MASSIVE_ARIMA_DATA_TBL VALUES ('GROUP_B', 13, 0.499210261 );
INSERT INTO PAL_MASSIVE_ARIMA_DATA_TBL VALUES ('GROUP_B', 14, 4.514017351 );
INSERT INTO PAL_MASSIVE_ARIMA_DATA_TBL VALUES ('GROUP_B', 15, -0.320607187);
INSERT INTO PAL_MASSIVE_ARIMA_DATA_TBL VALUES ('GROUP_B', 16, -3.70219307 );
INSERT INTO PAL_MASSIVE_ARIMA_PARAM_TBL VALUES ('GROUP_A', 'D', 1, NULL, NULL);
INSERT INTO PAL_MASSIVE_ARIMA_PARAM_TBL VALUES ('GROUP_A', 'Q', 0, NULL, NULL);
INSERT INTO PAL_MASSIVE_ARIMA_PARAM_TBL VALUES ('GROUP_A', 'SEASONAL_PERIOD', 4,
NULL, NULL);
INSERT INTO PAL_MASSIVE_ARIMA_PARAM_TBL VALUES ('GROUP_A', 'SEASONAL_P', 1,
NULL, NULL);
INSERT INTO PAL_MASSIVE_ARIMA_PARAM_TBL VALUES ('GROUP_A', 'METHOD', 1, NULL,
NULL);
INSERT INTO PAL_MASSIVE_ARIMA_PARAM_TBL VALUES ('GROUP_A', 'THREAD_RATIO', NULL,
1.0, NULL);
INSERT INTO PAL_MASSIVE_ARIMA_PARAM_TBL VALUES ('GROUP_B', 'D', 1, NULL, NULL);
INSERT INTO PAL_MASSIVE_ARIMA_PARAM_TBL VALUES ('GROUP_B', 'P', 1, NULL, NULL);
INSERT INTO PAL_MASSIVE_ARIMA_PARAM_TBL VALUES ('GROUP_B', 'SEASONAL_PERIOD', 4,
NULL, NULL);
```

```
INSERT INTO PAL_MASSIVE_ARIMA_PARAM_TBL VALUES ('GROUP_B', 'SEASONAL_P', 0,
NULL, NULL);
INSERT INTO PAL_MASSIVE_ARIMA_PARAM_TBL VALUES ('GROUP_B', 'METHOD', 1, NULL,
NULL);
INSERT INTO PAL_MASSIVE_ARIMA_PARAM_TBL VALUES ('GROUP_B', 'THREAD_RATIO', NULL,
1.0, NULL);
do begin
    lt_data = select * from PAL_MASSIVE_ARIMA_DATA_TBL;
    lt_control = select * from PAL_MASSIVE_ARIMA_PARAM_TBL;
    CALL _SYS_AFL.PAL_MASSIVE_ARIMA(:lt_data, :lt_control, lt_model, lt_fit,
lt_error);
    INSERT INTO PAL_MASSIVE_ARIMA_MODEL_TBL SELECT * FROM :lt_model;
    INSERT INTO PAL_MASSIVE_ARIMA_FIT_TBL SELECT * FROM :lt_fit;
    INSERT INTO PAL_MASSIVE_ARIMA_ERROR_TBL SELECT * FROM :lt_error;
end;
select * from PAL_MASSIVE_ARIMA_MODEL_TBL;
select * from PAL_MASSIVE_ARIMA_FIT_TBL;
select * from PAL_MASSIVE_ARIMA_ERROR_TBL;
```

### Expected Result

MODEL table:

| | GROUP_ID | KEY | VALUE |
|---|---|---|---|
| 1 | GROUP_B | p | 1 |
| 2 | GROUP_B | AR | -0.0276034 |
| 3 | GROUP_B | d | 1 |
| 4 | GROUP_B | q | 0 |
| 5 | GROUP_B | MA | |
| 6 | GROUP_B | s | 4 |
| 7 | GROUP_B | P | 0 |
| 8 | GROUP_B | SAR | |
| 9 | GROUP_B | D | 0 |
| 10 | GROUP_B | Q | 0 |
| 11 | GROUP_B | SMA | |
| 12 | GROUP_B | sigma^2 | 14.9867 |
| 13 | GROUP_B | log-likelihood | -41.5882 |
| 14 | GROUP_B | AIC | 87.1764 |
| 15 | GROUP_B | AICc | 88.1764 |
| 16 | GROUP_B | BIC | 88.5925 |
| 17 | GROUP_B | dy(n-p:n-1)_aux | |
| 18 | GROUP_B | dy_aux | 0 |
| 19 | GROUP_B | dy_0 | 3.72863;-3.82984;-2.40485;5.23101;1.09048;-4.05067;-2.60425;5.25487;1.82991;-3.52698;-4.506 |
| 20 | GROUP_B | x(n-d:n-1)_aux | |
| 21 | GROUP_B | y(n-d:n-1)_aux | 0 |
| 22 | GROUP_B | y(n-d:n-1)_0 | -3.70219 |
| 23 | GROUP_B | epsilon(n-q:n-1)_aux | |
| 24 | GROUP_A | p | 0 |
| 25 | GROUP_A | AR | |

FIT table:

| | GROUP_ID | TIMESTAMP | FITTED | RESIDUAL |
|---|---|---|---|---|
| 1 | GROUP_B | 1 | *NULL* | *NULL* |
| 2 | GROUP_B | 2 | -0.6361264310000001 | 3.7286350820000003 |
| 3 | GROUP_B | 3 | 2.9895856410481385 | -3.7269212010481385 |
| 4 | GROUP_B | 4 | -0.6316188338591844 | -2.5105721491408155 |
| 5 | GROUP_B | 5 | -3.0758087940244416 | 5.164628607024442 |
| 6 | GROUP_B | 6 | 1.9444261235198768 | 1.2348766104801234 |
| 7 | GROUP_B | 7 | 3.14920169647229 | -4.02057779847229 |
| 8 | GROUP_B | 8 | -0.7595635891150447 | -2.716069685884955 |
| 9 | GROUP_B | 9 | -3.403746919526964 | 5.182991138526964 |
| 10 | GROUP_B | 10 | 1.6341917274805913 | 1.9749676885194085 |
| 11 | GROUP_B | 11 | 3.558647532726384 | -3.476477389726384 |
| 12 | GROUP_B | 12 | 0.179527042793576 | -4.603923352793576 |
| 13 | GROUP_B | 13 | -4.299999748338608 | 4.799210009338608 |
| 14 | GROUP_B | 14 | 0.3633019736611738 | 4.150715377338826 |
| 15 | GROUP_B | 15 | 4.403195020310232 | -4.723802207310232 |
| 16 | GROUP_B | 16 | -0.18715510641420474 | -3.515037963585795 |
| 17 | GROUP_A | 1 | *NULL* | *NULL* |
| 18 | GROUP_A | 2 | -0.6361264310000001 | 3.7286350820000003 |
| 19 | GROUP_A | 3 | 3.092508651 | -3.829844211 |
| 20 | GROUP_A | 4 | -0.73733556 | -2.404855423 |
| 21 | GROUP_A | 5 | -3.1421909829999994 | 5.231010796 |
| 22 | GROUP_A | 6 | 5.620720661693623 | -2.4414179276936228 |
| 23 | GROUP_A | 7 | -0.4484671393499332 | -0.42290896265006683 |
| 24 | GROUP_A | 8 | -3.1493439703179757 | -0.3262893046820241 |
| 25 | GROUP_A | 9 | 1.4793733329429497 | 0.29987088605705026 |

ERROR_MSG table:

Empty

### Example 2: MASSIVE ARIMAX

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_MASSIVE_ARIMAX_DATA_TBL;
CREATE COLUMN TABLE PAL_MASSIVE_ARIMAX_DATA_TBL (
    GROUP_ID  NVARCHAR(40),
    TIMESTAMP INTEGER,
    Y         DOUBLE,
    X1        DOUBLE,
    X2        DOUBLE
);
DROP TABLE PAL_MASSIVE_ARIMAX_PARAM_TBL;
CREATE COLUMN TABLE PAL_MASSIVE_ARIMAX_PARAM_TBL (
    GROUP_ID   NVARCHAR(40),
    NAME       NVARCHAR(100),
    INTARGS    INTEGER,
    DOUBLEARGS DOUBLE,
    STRINGARGS NVARCHAR(100)
);
DROP TABLE PAL_MASSIVE_ARIMAX_MODEL_TBL;
CREATE COLUMN TABLE PAL_MASSIVE_ARIMAX_MODEL_TBL (
    GROUP_ID NVARCHAR(40),
    KEY      NVARCHAR(100),
    VALUE    NVARCHAR(5000)
);
DROP TABLE PAL_MASSIVE_ARIMAX_FIT_TBL;
CREATE COLUMN TABLE PAL_MASSIVE_ARIMAX_FIT_TBL (
```

```
    GROUP_ID   NVARCHAR(40),
    TIMESTAMP INTEGER,
    FITTED    DOUBLE,
    RESIDUAL  DOUBLE
);
DROP TABLE PAL_MASSIVE_ARIMAX_ERROR_TBL;
CREATE COLUMN TABLE PAL_MASSIVE_ARIMAX_ERROR_TBL (
    GROUP_ID        NVARCHAR(40),
    ERROR_TIMESTAMP NVARCHAR(100),
    ERRORCODE       INTEGER,
    ERROR_MESSAGE   NVARCHAR(200)
);
INSERT INTO PAL_MASSIVE_ARIMAX_DATA_TBL VALUES ('GROUP_A', 1, 0.882, 1.1,
-0.992 );
INSERT INTO PAL_MASSIVE_ARIMAX_DATA_TBL VALUES ('GROUP_A', 2, -0.077, 1.155,
-1.212 );
INSERT INTO PAL_MASSIVE_ARIMAX_DATA_TBL VALUES ('GROUP_A', 3, 0.135, 0.53,
-1.034 );
INSERT INTO PAL_MASSIVE_ARIMAX_DATA_TBL VALUES ('GROUP_A', 4, 0.373, 0.698,
-1.195 );
INSERT INTO PAL_MASSIVE_ARIMAX_DATA_TBL VALUES ('GROUP_A', 5, -0.437, 0.441,
-1.386 );
INSERT INTO PAL_MASSIVE_ARIMAX_DATA_TBL VALUES ('GROUP_A', 6, -0.556, 0.405,
-0.844 );
INSERT INTO PAL_MASSIVE_ARIMAX_DATA_TBL VALUES ('GROUP_A', 7, -0.432, -0.016,
-1.001 );
INSERT INTO PAL_MASSIVE_ARIMAX_DATA_TBL VALUES ('GROUP_A', 8, -0.46, 0.271,
-1.234 );
INSERT INTO PAL_MASSIVE_ARIMAX_DATA_TBL VALUES ('GROUP_A', 9, -0.698, -0.357,
-1.269 );
INSERT INTO PAL_MASSIVE_ARIMAX_DATA_TBL VALUES ('GROUP_A', 10, -0.765, -0.006,
-1.326 );
INSERT INTO PAL_MASSIVE_ARIMAX_DATA_TBL VALUES ('GROUP_A', 11, -0.833, -0.647,
-2.124 );
INSERT INTO PAL_MASSIVE_ARIMAX_DATA_TBL VALUES ('GROUP_A', 12, -0.767, -0.282,
-1.615 );
INSERT INTO PAL_MASSIVE_ARIMAX_DATA_TBL VALUES ('GROUP_A', 13, -0.356, -0.503,
-1.035 );
INSERT INTO PAL_MASSIVE_ARIMAX_DATA_TBL VALUES ('GROUP_A', 14, -0.496, -0.998,
-1.356 );
INSERT INTO PAL_MASSIVE_ARIMAX_DATA_TBL VALUES ('GROUP_A', 15, -0.684, -0.618,
-1.06 );
INSERT INTO PAL_MASSIVE_ARIMAX_DATA_TBL VALUES ('GROUP_A', 16, -0.953, -0.547,
-1.437 );
INSERT INTO PAL_MASSIVE_ARIMAX_DATA_TBL VALUES ('GROUP_A', 17, -0.869, -0.403,
-1.36 );
INSERT INTO PAL_MASSIVE_ARIMAX_DATA_TBL VALUES ('GROUP_A', 18, -0.831, -0.691,
-1.553 );
INSERT INTO PAL_MASSIVE_ARIMAX_DATA_TBL VALUES ('GROUP_A', 19, -0.858, -0.793,
-1.302 );
INSERT INTO PAL_MASSIVE_ARIMAX_DATA_TBL VALUES ('GROUP_A', 20, -0.769, -0.654,
-1.332 );
INSERT INTO PAL_MASSIVE_ARIMAX_DATA_TBL VALUES ('GROUP_B', 1, -0.732, 0.309,
-0.366 );
INSERT INTO PAL_MASSIVE_ARIMAX_DATA_TBL VALUES ('GROUP_B', 2, -0.682, -0.416,
0.034 );
INSERT INTO PAL_MASSIVE_ARIMAX_DATA_TBL VALUES ('GROUP_B', 3, -0.606, -0.206,
-0.129 );
INSERT INTO PAL_MASSIVE_ARIMAX_DATA_TBL VALUES ('GROUP_B', 4, -0.639, 1.17,
-0.284 );
INSERT INTO PAL_MASSIVE_ARIMAX_DATA_TBL VALUES ('GROUP_B', 5, -0.363, 0.858,
-0.292 );
INSERT INTO PAL_MASSIVE_ARIMAX_DATA_TBL VALUES ('GROUP_B', 6, 0.473, 0.281,
-0.508 );
INSERT INTO PAL_MASSIVE_ARIMAX_DATA_TBL VALUES ('GROUP_B', 7, 0.03, 0.599,
0.283 );
INSERT INTO PAL_MASSIVE_ARIMAX_DATA_TBL VALUES ('GROUP_B', 8, -0.688, -0.102,
-0.729 );
```

```
INSERT INTO PAL_MASSIVE_ARIMAX_DATA_TBL VALUES ('GROUP_B', 9, -0.963, -0.288,
-0.388 );
INSERT INTO PAL_MASSIVE_ARIMAX_DATA_TBL VALUES ('GROUP_B', 10, -0.879, -0.503,
-0.464 );
INSERT INTO PAL_MASSIVE_ARIMAX_DATA_TBL VALUES ('GROUP_B', 11, -1.132, -0.05,
-0.213 );
INSERT INTO PAL_MASSIVE_ARIMAX_DATA_TBL VALUES ('GROUP_B', 12, -1.031, 0.264,
0.132 );
INSERT INTO PAL_MASSIVE_ARIMAX_DATA_TBL VALUES ('GROUP_B', 13, -1.014, -0.059,
-0.043 );
INSERT INTO PAL_MASSIVE_ARIMAX_DATA_TBL VALUES ('GROUP_B', 14, -1.026, 0.521,
-0.828 );
INSERT INTO PAL_MASSIVE_ARIMAX_DATA_TBL VALUES ('GROUP_B', 15, -1.242, -0.437,
-0.803 );
INSERT INTO PAL_MASSIVE_ARIMAX_DATA_TBL VALUES ('GROUP_B', 16, -1.018, -0.068,
-0.975 );
INSERT INTO PAL_MASSIVE_ARIMAX_DATA_TBL VALUES ('GROUP_B', 17, -0.957, -0.691,
-1.082 );
INSERT INTO PAL_MASSIVE_ARIMAX_DATA_TBL VALUES ('GROUP_B', 18, -0.985, -0.249,
-0.777 );
INSERT INTO PAL_MASSIVE_ARIMAX_DATA_TBL VALUES ('GROUP_B', 19, -0.463, -0.413,
-0.583 );
INSERT INTO PAL_MASSIVE_ARIMAX_DATA_TBL VALUES ('GROUP_B', 20, -0.983, -0.534,
-1.448 );
INSERT INTO PAL_MASSIVE_ARIMAX_PARAM_TBL VALUES ('GROUP_A', 'D', 1, NULL, NULL);
INSERT INTO PAL_MASSIVE_ARIMAX_PARAM_TBL VALUES ('GROUP_A', 'Q', 0, NULL, NULL);
INSERT INTO PAL_MASSIVE_ARIMAX_PARAM_TBL VALUES ('GROUP_A', 'SEASONAL_PERIOD',
4, NULL, NULL);
INSERT INTO PAL_MASSIVE_ARIMAX_PARAM_TBL VALUES ('GROUP_A', 'SEASONAL_P', 1,
NULL, NULL);
INSERT INTO PAL_MASSIVE_ARIMAX_PARAM_TBL VALUES ('GROUP_A', 'METHOD', 1, NULL,
NULL);
INSERT INTO PAL_MASSIVE_ARIMAX_PARAM_TBL VALUES ('GROUP_A', 'THREAD_RATIO',
NULL, 1.0, NULL);
INSERT INTO PAL_MASSIVE_ARIMAX_PARAM_TBL VALUES ('GROUP_A', 'BACKGROUND_SIZE',
-1, NULL, NULL);
INSERT INTO PAL_MASSIVE_ARIMAX_PARAM_TBL VALUES ('GROUP_B', 'D', 1, NULL, NULL);
INSERT INTO PAL_MASSIVE_ARIMAX_PARAM_TBL VALUES ('GROUP_B', 'P', 1, NULL, NULL);
INSERT INTO PAL_MASSIVE_ARIMAX_PARAM_TBL VALUES ('GROUP_B', 'SEASONAL_PERIOD',
4, NULL, NULL);
INSERT INTO PAL_MASSIVE_ARIMAX_PARAM_TBL VALUES ('GROUP_B', 'SEASONAL_P', 0,
NULL, NULL);
INSERT INTO PAL_MASSIVE_ARIMAX_PARAM_TBL VALUES ('GROUP_B', 'METHOD', 1, NULL,
NULL);
INSERT INTO PAL_MASSIVE_ARIMAX_PARAM_TBL VALUES ('GROUP_B', 'THREAD_RATIO',
NULL, 1.0, NULL);
INSERT INTO PAL_MASSIVE_ARIMAX_PARAM_TBL VALUES ('GROUP_B', 'BACKGROUND_SIZE',
-1, NULL, NULL);
do begin
    lt_data = select * from PAL_MASSIVE_ARIMAX_DATA_TBL;
    lt_control = select * from PAL_MASSIVE_ARIMAX_PARAM_TBL;
    CALL _SYS_AFL.PAL_MASSIVE_ARIMA(:lt_data, :lt_control, lt_model, lt_fit,
lt_error);
    INSERT INTO PAL_MASSIVE_ARIMAX_MODEL_TBL SELECT * FROM :lt_model;
    INSERT INTO PAL_MASSIVE_ARIMAX_FIT_TBL SELECT * FROM :lt_fit;
    INSERT INTO PAL_MASSIVE_ARIMAX_ERROR_TBL SELECT * FROM :lt_error;
end;
select * from PAL_MASSIVE_ARIMAX_MODEL_TBL;
select * from PAL_MASSIVE_ARIMAX_FIT_TBL;
select * from PAL_MASSIVE_ARIMAX_ERROR_TBL;
```

### Expected Result

MODEL table:

| | GROUP_ID | KEY | VALUE |
|---|---|---|---|
| 1 | GROUP_A | p | 0 |
| 2 | GROUP_A | AR | |
| 3 | GROUP_A | d | 1 |
| 4 | GROUP_A | q | 0 |
| 5 | GROUP_A | MA | |
| 6 | GROUP_A | s | 4 |
| 7 | GROUP_A | P | 1 |
| 8 | GROUP_A | SAR | 0.662262 |
| 9 | GROUP_A | D | 0 |
| 10 | GROUP_A | Q | 0 |
| 11 | GROUP_A | SMA | |
| 12 | GROUP_A | regressor | X1;X2 |
| 13 | GROUP_A | beta | -0.229894;0.457867 |
| 14 | GROUP_A | sigma^2 | 0.0689206 |
| 15 | GROUP_A | log-likelihood | -2.70385 |
| 16 | GROUP_A | AIC | 13.4077 |
| 17 | GROUP_A | AICc | 16.2648 |
| 18 | GROUP_A | BIC | 17.1854 |
| 19 | GROUP_A | dy(n-p:n-1)_aux | |
| 20 | GROUP_A | dy_aux | 0 |
| 21 | GROUP_A | dy_0 | -0.845624;-0.0131845;0.350338;-0.78163;-0.37544;0.0990996;0.144662;-0.366348;0.0397914;0.150016; |
| 22 | GROUP_A | x(n-d:n-1)_aux | 0 |
| 23 | GROUP_A | x(n-d:n-1)_0 | -0.654;-1.332 |
| 24 | GROUP_A | y(n-d:n-1)_aux | 0 |
| 25 | GROUP_A | y(n-d:n-1)_0 | -0.769 |

FIT table:

| | GROUP_ID | TIMESTAMP | FITTED | RESIDUAL |
|---|---|---|---|---|
| 1 | GROUP_A | 1 | *NULL* | *NULL* |
| 2 | GROUP_A | 2 | 0.7686248750332658 | -0.8456248750332658 |
| 3 | GROUP_A | 3 | 0.14818450197873864 | -0.013184501978738621 |
| 4 | GROUP_A | 4 | 0.022661008214823697 | 0.3503389917851763 |
| 5 | GROUP_A | 5 | 0.34463010669119426 | -0.7816301066911943 |
| 6 | GROUP_A | 6 | -0.7405853468925061 | 0.18458534689250605 |
| 7 | GROUP_A | 7 | -0.5398313042931528 | 0.1078313042931529 |
| 8 | GROUP_A | 8 | -0.37264642033948586 | -0.08735357966051416 |
| 9 | GROUP_A | 9 | -0.8492962449888803 | 0.15129624498888045 |
| 10 | GROUP_A | 10 | -1.0534317625750225 | 0.28843176257502257 |
| 11 | GROUP_A | 11 | -0.9173861789860904 | 0.08438617898609047 |
| 12 | GROUP_A | 12 | -0.5880518497951787 | -0.17894815020482135 |
| 13 | GROUP_A | 13 | -0.6932488512352135 | 0.33724885123521353 |
| 14 | GROUP_A | 14 | -0.3628254654385894 | -0.1331745345614106 |
| 15 | GROUP_A | 15 | -0.34848082349444187 | -0.3355191765055582 |
| 16 | GROUP_A | 16 | -0.9280013937796684 | -0.02499860622033155 |
| 17 | GROUP_A | 17 | -0.8881790624355012 | 0.0191790624355012 |
| 18 | GROUP_A | 18 | -0.9619032480535873 | 0.13090324805358736 |
| 19 | GROUP_A | 19 | -0.8490318880705227 | -0.008968111929477357 |
| 20 | GROUP_A | 20 | -0.9567129920953821 | 0.187712992095382 |
| 21 | GROUP_B | 1 | *NULL* | *NULL* |
| 22 | GROUP_B | 2 | -0.6452618938645001 | -0.03673810613549995 |
| 23 | GROUP_B | 3 | -0.7132817247774011 | 0.10728172477740112 |
| 24 | GROUP_B | 4 | -0.6752769626753176 | 0.03627696267531758 |
| 25 | GROUP_B | 5 | -0.6329966019621556 | 0.2699966019621556 |

ERROR_MSG table:

Empty

## _SYS_AFL.PAL_MASSIVE_ARIMA_FORECAST

This function is the MASSIVE version of _SYS_AFL.PAL_ARIMA_FORECAST, by which multiple time series can be forecasted in a single call.

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<MODEL table>` | IN |
| 3 | `<PARAMETER table>` | IN |
| 4 | `<RESULT table>` | OUT |
| 5 | `<ERROR_MSG table>` | OUT |

### Input Table

| Table | Column | Data Type | Description |
|---|---|---|---|
| DATA | 1st column | INTEGER, VRACHAR or NVARCHAR | Group ID. |
| | 2nd column | INTEGER | Time stamp. Reference for Output Table is "ID" |
| | Other columns | INTEGER, DOUBLE, DECIMAL (p,s), VARCHAR, or NVARCHAR | External (Intervention) data. |
| MODEL | 1st column | INTEGER, VRACHAR, or NVARCHAR | Group ID. |
| | 2nd column | VARCHAR or NVARCHAR | Model key. |
| | 3rd column | VARCHAR or NVARCHAR | Model value. |

### Parameter Table

Same as _SYS_AFL.PAL_ARIMA_FORECAST.

### Output Table

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| RESULT | 1st column | VARCHAR | GROUP_ID | Group ID. |
| | 2nd column | INTEGER | ID (in input table) column name | Time stamp. |
| | 3rd column | DOUBLE | FORECAST | Forecast value. |
| | 4th column | DOUBLE | SE | Standard error. |
| | 5th column | DOUBLE | LOWER_80 | Lower 80% value. |
| | 6th column | DOUBLE | UPPER_80 | Upper 80% value. |
| | 7th column | DOUBLE | LOWER_95 | Lower 95% value. |
| | 8th column | DOUBLE | UPPER_95 | Upper 95% value. |
| ERROR_MSG | 1st column | VARCHAR | GROUP_ID | Group ID. |
| | 2nd column | VARCHAR | ERROR_TIMESTAMP | Time stamp when the error occured. |
| | 3rd column | INTEGER | ERROR_CODE | Error code |
| | 4th column | VARCHAR | ERROR_MESSAGE | Error message. |

### Examples

Assume that:

- DM_PAL is a schema belonging to USER1;
- USER1 has been assigned the AFL__SYS_AFL_AFLPAL_EXECUTE or AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION role.

### Example: MASSIVE ARIMA_FORECAST

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_MASSIVE_PRED_ARIMA_DATA_TBL;
```

```
CREATE COLUMN TABLE PAL_MASSIVE_PRED_ARIMA_DATA_TBL ("GROUP_ID"  NVARCHAR(40),
"TIMESTAMP" INTEGER, "Y" DOUBLE);
DROP TABLE PAL_MASSIVE_PRED_ARIMA_PARAM_TBL;
CREATE COLUMN TABLE PAL_MASSIVE_PRED_ARIMA_PARAM_TBL (
    GROUP_ID   NVARCHAR(40),
    NAME       NVARCHAR(100),
    INTARGS    INTEGER,
    DOUBLEARGS DOUBLE,
    STRINGARGS NVARCHAR(100)
);
INSERT INTO PAL_MASSIVE_ARIMA_PARAM_TBL VALUES ('GROUP_A', 'FORECAST_METHOD', 1,
NULL, NULL);
INSERT INTO PAL_MASSIVE_ARIMA_PARAM_TBL VALUES ('GROUP_A', 'FORECAST_LENGTH',
10, NULL, NULL);
INSERT INTO PAL_MASSIVE_ARIMA_PARAM_TBL VALUES ('GROUP_B', 'FORECAST_METHOD', 2,
NULL, NULL);
INSERT INTO PAL_MASSIVE_ARIMA_PARAM_TBL VALUES ('GROUP_B', 'FORECAST_LENGTH',
10, NULL, NULL);
CALL _SYS_AFL.PAL_MASSIVE_ARIMA_FORECAST(PAL_MASSIVE_PRED_ARIMA_DATA_TBL,
PAL_MASSIVE_ARIMA_MODEL_TBL, PAL_MASSIVE_ARIMA_PARAM_TBL, ?, ?);
```

### Expected Result

RESULT table:

| | GROUP_ID | TIMESTAMP | FORECAST | SE | LO80 | HI80 | LO95 | HI95 |
|---|---|---|---|---|---|---|---|---|
| 1 | GROUP_A | 0 | 0.9616211696000008 | 1.3113504489647305 | -0.7189426209716476 | 2.642184960171649 | -1.6085785017547085 | 3.53182084095471 |
| 2 | GROUP_A | 1 | 4.764584262400001 | 1.854529589949969 | 2.387908157340439 | 7.141260367459564 | 1.1297730291633004 | 8.399395495636703 |
| 3 | GROUP_A | 2 | 0.18505815208000165 | 2.2713256041351713 | -2.7257637185506356 | 3.095880022710639 | -4.266658264303185 | 4.6367745684631885 |
| 4 | GROUP_A | 3 | -3.018096160799998 | 2.622700897929461 | -6.379223741943295 | 0.3430314203432987 | -8.158495503509418 | 2.1223031819094205 |
| 5 | GROUP_A | 4 | 1.3996336762472277 | 3.6604599719410715 | -3.2914361217137964 | 6.090703474208252 | -5.774736092198283 | 8.574003444692739 |
| 6 | GROUP_A | 5 | 5.001937224418729 | 4.463112637203509 | -0.7177737020147026 | 10.721648150852161 | -3.745602872445211 | 13.749477321282669 |
| 7 | GROUP_A | 6 | 0.6640452297836532 | 5.141962817694084 | -5.925647503157836 | 7.253737962725142 | -9.414016782235315 | 10.742107241802621 |
| 8 | GROUP_A | 7 | -2.3700978489315463 | 5.741096482792405 | -9.727611528647119 | 4.987415830784027 | -13.62244027573128 | 8.882244577868187 |
| 9 | GROUP_A | 8 | 1.8145348909937193 | 6.846460994852877 | -6.959560889881974 | 10.588630671869414 | -11.604282186322104 | 15.233351968309544 |
| 10 | GROUP_A | 9 | 5.226766494749499 | 7.796657455817356 | -4.765055461068147 | 15.218588450567143 | -10.054401438984112 | 20.507934428483107 |
| 11 | GROUP_B | 0 | -3.608846894628 | 3.8712659428150893 | -8.570075506174565 | 1.352381716918564 | -11.196388776971634 | 3.9786949877156337 |
| 12 | GROUP_B | 1 | -3.6114234817028255 | 5.399764007057077 | -10.531501844474835 | 3.3086548810691854 | -14.194766544030442 | 6.971919580624791 |
| 13 | GROUP_B | 2 | -3.6113523591391643 | 6.58410351654288 | -12.049223388971726 | 4.826518670693396 | -16.515958223836613 | 9.293253505558285 |
| 14 | GROUP_B | 3 | -3.611354322363738 | 7.585695315396181 | -13.332817325200343 | 6.110108680472868 | -18.479044055508897 | 11.256335410781421 |
| 15 | GROUP_B | 4 | -3.6113542681720645 | 8.469660244280066 | -14.465664293549672 | 7.242955757205543 | -20.2115834391922 | 12.988874902848071 |
| 16 | GROUP_B | 5 | -3.611354269667939 | 9.269709565116678 | -15.490969102262348 | 8.268260562926471 | -21.77965130775228 | 14.556942768416405 |
| 17 | GROUP_B | 6 | -3.6113542696266476 | 10.005992514618221 | -16.43455398872066 | 9.211845449467361 | -23.222739382547836 | 16.00003084329454 |
| 18 | GROUP_B | 7 | -3.6113542696277876 | 10.691691025428106 | -17.31331228664723 | 10.090603747391654 | -24.56668377858996 | 17.343975239334384 |
| 19 | GROUP_B | 8 | -3.611354269627756 | 11.335988168830003 | -18.139012579368185 | 10.91630404011267 | -25.829482984960485 | 18.60677444570497 |
| 20 | GROUP_B | 9 | -3.611354269627757 | 11.94558489754635 | -18.92024248624808 | 11.697533946992564 | -27.02427062776229 | 19.801562088506778 |

ERROR_MSG table:

Empty

## _SYS_AFL.PAL_MASSIVE_ARIMA_EXPLAIN

This function is the MASSIVE version of _SYS_AFL.PAL_ARIMA_EXPLAIN, by which multiple time series can be forecasted and explained in a single call.

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |

| Position | Table Name | Parameter Type |
|---|---|---|
| 2 | `<MODEL table>` | IN |
| 3 | `<PARAMETER table>` | IN |
| 4 | `<RESULT table>` | OUT |
| 5 | `<DECOMPOSITION table>` | OUT |
| 6 | `<ERROR_MSG table>` | OUT |

### Input Table

| Table | Column | Data Type | Description |
|---|---|---|---|
| DATA | 1st column | INTEGER, VRACHAR or NVARCHAR | Group ID. |
| | 2nd column | INTEGER | Time stamp. Reference for Output Table is "ID" |
| | Other columns | INTEGER, DOUBLE, DECIMAL (p,s), VARCHAR, or NVARCHAR | External (Intervention) data. |
| MODEL | 1st column | INTEGER, VRACHAR or NVARCHAR | Group ID. |
| | 2nd column | VARCHAR or NVARCHAR | Model key. |
| | 3rd column | VARCHAR or NVARCHAR | Model value. |

### Parameter Table

Same as _SYS_AFL.PAL_ARIMA_EXPLAIN.

### Output Table

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| RESULT | 1st column | VARCHAR | GROUP_ID | Group ID. |
| | 2nd column | INTEGER | ID (in input table) column name | Time stamp. |
| | 3rd column | DOUBLE | FORECAST | Forecast value. |
| | 4th column | DOUBLE | SE | Standard error. |
| | 5th column | DOUBLE | LOWER_80 | Lower 80% value. |
| | 6th column | DOUBLE | UPPER_80 | Upper 80% value. |
| | 7th column | DOUBLE | LOWER_95 | Lower 95% value. |
| | 8th column | DOUBLE | UPPER_95 | Upper 95% value. |
| DECOMPOSITION | 1st column | VARCHAR | GROUP_ID | Group ID. |
| | 2nd column | INTEGER | ID (in input table) column name | Time stamp. |
| | 3rd column | DOUBLE | TREND | The trend component. |

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| | 4th column | DOUBLE | SEASONAL | The seasonal component. |
| | 5th column | DOUBLE | TRANSITORY | The transitory component. |
| | 6th column | DOUBLE | IRREGULAR | The irregular component. |
| | 7th column | NCLOB | EXOGENOUS | The contributions of exogenous data to forecast values. |
| ERROR_MSG | 1st column | VARCHAR | GROUP_ID | Group ID. |
| | 2nd column | VARCHAR | ERROR_TIMESTAMP | Time stamp when the error occured. |
| | 3rd column | INTEGER | ERROR_CODE | Error code |
| | 4th column | VARCHAR | ERROR_MESSAGE | Error message. |

### Examples

Assume that:

- DM_PAL is a schema belonging to USER1;
- USER1 has been assigned the AFL__SYS_AFL_AFLPAL_EXECUTE or AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION role.

**Example: MASSIVE ARIMA_EXPLAIN**

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_MASSIVE_PRED_ARIMAX_DATA_TBL;
CREATE COLUMN TABLE PAL_MASSIVE_PRED_ARIMAX_DATA_TBL (
    GROUP_ID  NVARCHAR(40),
    TIMESTAMP INTEGER,
    X1        DOUBLE,
    X2        DOUBLE
);
DROP TABLE PAL_MASSIVE_PRED_ARIMA_PARAM_TBL;
CREATE COLUMN TABLE PAL_MASSIVE_PRED_ARIMA_PARAM_TBL (
    GROUP_ID   NVARCHAR(40),
    NAME       NVARCHAR(100),
    INTARGS    INTEGER,
    DOUBLEARGS DOUBLE,
    STRINGARGS NVARCHAR(100)
);
INSERT INTO PAL_MASSIVE_PRED_ARIMAX_DATA_TBL VALUES ('GROUP_A', 21, -0.78,
-1.362 );
INSERT INTO PAL_MASSIVE_PRED_ARIMAX_DATA_TBL VALUES ('GROUP_A', 22, -0.575,
-1.459 );
INSERT INTO PAL_MASSIVE_PRED_ARIMAX_DATA_TBL VALUES ('GROUP_A', 23, -0.575,
-1.522 );
INSERT INTO PAL_MASSIVE_PRED_ARIMAX_DATA_TBL VALUES ('GROUP_A', 24, -0.792,
-1.255 );
INSERT INTO PAL_MASSIVE_PRED_ARIMAX_DATA_TBL VALUES ('GROUP_A', 25, -0.189,
-1.096 );
INSERT INTO PAL_MASSIVE_PRED_ARIMAX_DATA_TBL VALUES ('GROUP_B', 21, -0.5,
-0.414 );
INSERT INTO PAL_MASSIVE_PRED_ARIMAX_DATA_TBL VALUES ('GROUP_B', 22, -0.543,
-0.748 );
```

```
INSERT INTO PAL_MASSIVE_PRED_ARIMAX_DATA_TBL VALUES ('GROUP_B', 23, -0.275,
-0.009 );
INSERT INTO PAL_MASSIVE_PRED_ARIMAX_DATA_TBL VALUES ('GROUP_B', 24, -0.604,
-1.213 );
INSERT INTO PAL_MASSIVE_PRED_ARIMAX_DATA_TBL VALUES ('GROUP_B', 25, -0.749,
-0.74 );
INSERT INTO PAL_MASSIVE_ARIMAX_PARAM_TBL VALUES ('GROUP_A', 'FORECAST_METHOD',
1, NULL, NULL);
INSERT INTO PAL_MASSIVE_ARIMAX_PARAM_TBL VALUES ('GROUP_B', 'FORECAST_METHOD',
1, NULL, NULL);
CALL _SYS_AFL.PAL_MASSIVE_ARIMA_EXPLAIN(PAL_MASSIVE_PRED_ARIMAX_DATA_TBL,
PAL_MASSIVE_ARIMAX_MODEL_TBL, PAL_MASSIVE_ARIMAX_PARAM_TBL, ?, ?, ?);
```

**Expected Result**

RESULT table:

| | GROUP_ID | TIMESTAMP | FORECAST | SE | LO80 | HI80 | LO95 | HI95 |
|---|---|---|---|---|---|---|---|---|
| 1 | GROUP_B | 0 | -0.7950801885376 | 0.3402249256006973 | -1.2310961223910248 | -0.3590642546841752 | -1.461908794617645 | -0.12825158245755497 |
| 2 | GROUP_B | 1 | -0.8500982331802904 | 0.4703606267678667 | -1.4528898351359034 | -0.24730663122467733 | -1.7719881286627457 | 0.07179166230216472 |
| 3 | GROUP_B | 2 | -0.7349111791266804 | 0.5719972043585068 | -1.4679553403667334 | -0.0018670178866272424 | -1.8560051077699966 | 0.386182749516636 |
| 4 | GROUP_B | 3 | -0.9255406745418573 | 0.6581046958123086 | -1.7689360636695128 | -0.08214528541420163 | -2.215402186564933 | 0.36432083748121835 |
| 5 | GROUP_B | 4 | -0.8429569914647594 | 0.7341823468991711 | -1.7838498464980859 | 0.0979358635685671 | -2.2819279608226464 | 0.5960139778931275 |
| 6 | GROUP_A | 0 | -0.6995639497882 | 0.2625273319104127 | -1.0360063770526533 | -0.3631215225237468 | -1.2141080693486601 | -0.18501983022773982 |
| 7 | GROUP_A | 1 | -0.7512643653564001 | 0.3712697132813287 | -1.2270658089515134 | -0.2754629217612867 | -1.4789396376781263 | -0.023589093034673936 |
| 8 | GROUP_A | 2 | -0.8896309023444 | 0.454710677244333 | -1.4723662801882296 | -0.30689552450057056 | -1.780847460158912 | 0.001585655470111913 |
| 9 | GROUP_A | 3 | -0.6282926843023999 | 0.5250546638208254 | -1.3011775388313063 | 0.04459217022650652 | -1.6573809234233203 | 0.40079555481852047 |
| 10 | GROUP_A | 4 | -0.6582197259511409 | 0.6827283066351346 | -1.533171552776011 | 0.21673210087372918 | -1.9963426287369659 | 0.6799031768346842 |

DECOMPOSITION table:

| | GROUP_ID | TIMESTAMP | TREND | SEASONAL | TRANSITORY | IRREGULAR | EXOGENOUS |
|---|---|---|---|---|---|---|---|
| 1 | GROUP_B | 0 | -0.85174273512762 | 0.00235184312519152 5 | NULL | 0.054310703464832605 | [{"attr":"X2","val":0.18762410188 |
| 2 | GROUP_B | 1 | -0.80970976831811 | -0.0015331192848127 78 | NULL | -0.0388553455773579 | [{"attr":"X2","val":-0.0394425635 |
| 3 | GROUP_B | 2 | -0.81311154534354 | 0.001819145303142753 | NULL | 0.0763812209137177 | [{"attr":"X2","val":0.13861152853 |
| 4 | GROUP_B | 3 | -0.84684622902707 | -0.0001584038006153016 | NULL | -0.07853604171416904 | [{"attr":"X2","val":-0.1838043767 |
| 5 | GROUP_B | 4 | -0.64299135635809 | -0.0036044745617069334 | NULL | -0.19636116054496044 | [{"attr":"X2","val":0.09453460198 |
| 6 | GROUP_A | 0 | -0.80084914624313 | 0.007362300412301173 | NULL | 0.12139039022195457 | [{"attr":"X2","val":0.01255965068 |
| 7 | GROUP_A | 1 | -0.81371538604068 | -0.002038682119226043 | NULL | 0.02182078451768535 | [{"attr":"X2","val":-0.0860546735 |
| 8 | GROUP_A | 2 | -0.77407322641459 | -0.0032187441933462786 | NULL | 0.005143995527502596 | [{"attr":"X2","val":-0.0284112643 |
| 9 | GROUP_A | 3 | -0.65684220183243 | 0.004984249270330117 | NULL | 0.09610020920150075 | [{"attr":"X2","val":0.16722375815 |
| 10 | GROUP_A | 4 | -0.47159111644473 | -0.015124627194926663 | NULL | -0.05532643696054372 | [{"attr":"X2","val":-0.0505296025 |

ERROR_MSG table:

Empty

## Related Information

Auto ARIMA [page 627]

# 5.5.2.1 Explaining the Forecasts of ARIMA

Simply getting the forecasting value is not enough for diving deeply into the model and data. Understanding the reason behind why such predictions are made is also a crucial demand for user to trust the predictions and make decisions.

Financial analyst may be curious about which latent stock affects the stock of interest the most, relevant departments will be benefited a lot if they have a knowledge of what are the major factors of varying weather,

unemployment rates, electricity consumption and so on. In all those scenarios, the importance of exogenous variables (referring to variables that are not affected by others) can be quantified as the contributions they make.

Let's recap the formula of an ARIMAX model with p exogenous variables $\mathbf{x}_t = [x_t^{(1)}, x_t^{(2)}, \ldots, x_t^{(p)}]$:

$$\Phi(B)\phi(B^s)\tilde{y}_t = \theta(B)\Theta(B^s)\epsilon_t + \sum_{i=1}^{p}\beta_i\tilde{x}_t^{(i)},$$

where $\tilde{y}_t = (1-B)^d(1-B^s)^D y_t$, $\tilde{x}_t^{(i)} = (1-B)^d(1-B^s)^D x_t^{(i)}$ and $\beta_i$ is the corresponding regression coefficients. As can be seen from the formula, there are two major parts that can be considered individually in explaining the forecasted values, which are, ARIMA part and regressor part.

To interpret the ARIMA part, we have implemented a decomposition method derived from the Digital Signal Processing realm. From the frequency domain, any time series has some possible components in low-frequency, high-frequency, and band-frequency areas as well as some irregular ones. Particularly, trend part can be assumed to have low frequencies and seasonal part to have high frequencies. Instead of using Fast Fourier Transform (FFT), the auto-regressive and moving-average parts of an ARIMA model also reveal those components inside a time series. Therefore, our algorithm decomposes the forecasted values into trend, seasonal, transitory, and irregular parts directly from the trained model. If a specific component is marked "?" in the output table, then it means the corresponding component cannot be decomposed by our setting.

To interpret the regressor part, we adopted the famous LinearSHAP algorithm, which is able to generate the contribution of each exogenous feature to the forecasted values given the regression coefficients and background data.

Combining the two parts above, we took a step further by adding a new function named PAL_ARIMA_EXPLAIN. It inherits all the functionalities and parameters of PAL_ARIMA_FORECAST as well as being capable of decomposing the forecasts into different interpretable parts.

## _SYS_AFL.PAL_ARIMA_EXPLAIN

To explain the exogenous features, we must set a non-zero value of BACKGROUND_SIZE during training in _SYS_AFL.PAL_ARIMA.

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<MODEL table>` | IN |
| 3 | `<PARAMETER table>` | IN |
| 4 | `<FORECAST table>` | OUT |
| 5 | `<DECOMPOSITION table>` | OUT |

### Note

The parameter `BACKGROUND_SIZE` must be set to a non-zero positive value during training, then one can obtain a non-empty decomposition table.

### Input Table

| Table | Column | Reference for Output Table | Data Type | Description |
|---|---|---|---|---|
| FORECAST_DATA | 1st column | ID | INTEGER | Time stamp. |
| | Other columns | | INTEGER, DOUBLE, or DECIMAL (p,s) | External (Intervention) data. |

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| MODEL | 1st column | VARCHAR or NVARCHAR | ROW_INDEX | Model key. |
| | 2nd column | VARCHAR or NVARCHAR | MODEL_CONTENT | Model value. |

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| THREAD_RATIO | DOUBLE | -1 | The ratio of available threads.<br>• 0: single thread<br>• 0~1: percentage<br>• Others: heuristically determined | |
| TOP_K_ATTRIBUTIONS | INTEGER | 10 | Specifies the number of attributes with the largest contribution that will be output. | Zero-contributed attributes will not be output |
| TREND_MOD | DOUBLE | 0.4 | The real AR roots with inverse modulus larger than TREND_MOD will be integrated into trend component. | Cannot be smaller than 0 |
| TREND_WIDTH | DOUBLE | 0.035 | Specifies the bandwidth of spectrum of trend component in unit of rad. | Cannot be smaller than 0 |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| SEASONAL_WIDTH | DOUBLE | 0.035 | Specifies the bandwidth of spectrum of seasonal component in unit of rad. | Cannot be smaller than 0 |

### Output Table

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| RESULT | 1st column | INTEGER | ID (in input table) column name | Time stamp. |
| | 2nd column | DOUBLE | FORECAST | Forecast value. |
| | 3rd column | DOUBLE | SE | Standard error. |
| | 4th column | DOUBLE | LOWER_80 | Lower 80% value. |
| | 5th column | DOUBLE | UPPER_80 | Upper 80% value. |
| | 6th column | DOUBLE | LOWER_95 | Lower 95% value. |
| | 7th column | DOUBLE | UPPER_95 | Upper 95% value. |

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| DECOMPOSITION | 1st column | INTEGER | ID (in input table) column name | Time stamp. |
| | 2nd column | DOUBLE | TREND | The trend component. |
| | 3rd column | DOUBLE | SEASONAL | The seasonal component. |
| | 4th column | DOUBLE | TRANSITORY | The transitory component. |
| | 5th column | DOUBLE | IRREGULAR | The irregular component. |
| | 6th column | NCLOB | EXOGENOUS | The contributions of exogenous data to forecast values. |

### Examples

Assume that:

- DM_PAL is a schema belonging to USER1;
- USER1 has been assigned the AFL__SYS_AFL_AFLPAL_EXECUTE or AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION role.

#### Example 1: ARIMA forecast and explain

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_ARIMA_DATA_TBL;
CREATE COLUMN TABLE PAL_ARIMA_DATA_TBL ("TIMESTAMP" INTEGER, "Y" DOUBLE);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL ( "NAME" VARCHAR
(50),"INT_VALUE" INTEGER,"DOUBLE_VALUE" DOUBLE,"STRING_VALUE" VARCHAR (100));
```

```
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FORECAST_METHOD', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FORECAST_LENGTH', 10, NULL, NULL);
CALL _SYS_AFL.PAL_ARIMA_EXPLAIN (PAL_ARIMA_DATA_TBL, PAL_ARIMA_MODEL_TBL,
"#PAL_PARAMETER_TBL", ?, ?);
```

### Expected Result

DECOMPOSITION table:

| | TIMESTAMP | TREND | SEASONAL | TRANSITORY | IRREGULAR | EXOGENOUS |
|---|---|---|---|---|---|---|
| 1 | 0 | 0.20662908320915735 | 1.3511066197462227 | ? | 0.00004705982629708464 | |
| 2 | 1 | 0.20034565007586802 | 3.565598294388556 | ? | 0.00004313319406587672 6 | |
| 3 | 2 | 0.19428195689659145 | -0.75990371172938... | ? | 0.00002280055630138018 7 | |
| 4 | 3 | 0.1884596070460145 | -3.808150393976914 | ? | -0.00012503951059269342 | |
| 5 | 4 | 0.18307560355316777 | 1.1760407129847208 | ? | -0.001182714236211337 | |
| 6 | 5 | 0.1795647787565716 | 3.1076410009048843 | ? | -0.00867543347409693 | |
| 7 | 6 | 0.18814196247626686 | -0.61525452458472... | ? | -0.06177570090980079 | |
| 8 | 7 | 0.24689168451774118 | -3.18655039124075... | ? | -0.20564995194510918 | |
| 9 | 8 | 0.3640554690400997 | 0.7401196154537515 | ? | 0.07993871123454413 | |
| 10 | 9 | 0.45674364449559873 | 2.183465477995427 | ? | 0.21435329421566388 | |

### Example 2: ARIMAX forecast and explain

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_ARIMA_DATA_TBL;
CREATE COLUMN TABLE PAL_ARIMA_DATA_TBL ("TIMESTAMP" INTEGER, "X" DOUBLE);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(1, 0.8);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(2, 1.2);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(3, 1.34845613096197);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(4, 1.32261090809898);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(5, 1.38095306748554);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(6, 1.54066648969168);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(7, 1.50920806756785);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(8, 1.48461408893443);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(9, 1.43784887380224);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(10, 1.64251548718992);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL ( "NAME" VARCHAR
(50),"INT_VALUE" INTEGER,"DOUBLE_VALUE" DOUBLE,"STRING_VALUE" VARCHAR (100));
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FORECAST_METHOD', 1, NULL, NULL);
CALL _SYS_AFL.PAL_ARIMA_EXPLAIN (PAL_ARIMA_DATA_TBL, PAL_ARIMA_MODEL_TBL,
"#PAL_PARAMETER_TBL", ?, ?);
```

### Expected Result

DECOMPOSITION table:

| | TIMESTAMP | TREND | SEASONAL | TRANSITORY | IRREGULAR | EXOGENOUS |
|---|---|---|---|---|---|---|
| 1 | 0 | 1.178439636262948 | ? | ? | ? | [{"attr":"X","val":4.0406787154930996e77,"pct":100.0}] |
| 2 | 1 | 1.25231892094667 | ? | ? | ? | [{"attr":"X","val":2.188247511170205e77,"pct":100.0}] |
| 3 | 2 | 1.362116277805039 | ? | ? | ? | [{"attr":"X","val":1.5007355875027054e77,"pct":100.0}] |
| 4 | 3 | 1.351502571277516 | ? | ? | ? | [{"attr":"X","val":1.6204268307879109e77,"pct":100.0}] |
| 5 | 4 | 1.3887693735616... | ? | ? | ? | [{"attr":"X","val":1.3502397393498019e77,"pct":100.0}] |
| 6 | 5 | 1.4428012436813... | ? | ? | ? | [{"attr":"X","val":6.105944217401744e76,"pct":100.0}] |
| 7 | 6 | 1.4582339095619... | ? | ? | ? | [{"attr":"X","val":7.562808286925349e76,"pct":100.0}] |
| 8 | 7 | 1.3996259694843... | ? | ? | ? | [{"attr":"X","val":8.701774623400305e76,"pct":100.0}] |
| 9 | 8 | 1.50865385847656 | ? | ? | ? | [{"attr":"X","val":1.0867508218093852e77,"pct":100.0}] |
| 10 | 9 | 1.2250897667774... | ? | ? | ? | [{"attr":"X","val":1.3892376900331427e76,"pct":100.0}] |

## 5.5.3 Auto ARIMA

Although the ARIMA model is useful and powerful in time series analysis, it is somehow difficult to choose appropriate orders. It is necessary, therefore, to determine the orders automatically. In PAL, the auto-ARIMA function identifies the orders of an ARIMA model.

The orders of an ARIMA model is `(p, d, q) (P, D, Q)`$_m$, where `m` is the seasonal period according to some information criterion such as AICc, AIC, and BIC. If order selection succeeds, the function gives the optimal model as in the ARIMATRAIN function.

At the first sight, there are 7 parameters, i.e. p, d, q, P, D, Q, and m, to be determined. As a first stage, however, the seasonality m can be estimated using other time series technique, such as SEASONALITYTEST in PAL. Due to the fact that over-differencing causes that a series has lower criterion information, d and D should be firstly determined. The commonly applied scheme is unit root test, Kwiatkowski-Phillips-Schmidt-Shin (KPSS) Test, Canova-Hansen (CH) Test, etc.

To this point, the remaining task is to obtain the optimal p, q, P, Q. In general, two different approaches can be adopted.

Intuitively, we can try all possible (p, q, P, Q) quadruplets thoroughly from a subset, and that enjoys least criterion information. Consider that, for example, orders are constraint to be no larger than mp, mq, mP, and mQ, therefore in total mp * mq * mP * mQ + 1 models should be checked, and we call such method **exhaustive search**. Obviously, it is a hard burden when there is a long series.

Alternatively, by evaluating the series properties firstly, e.g. ACF, PACF, one can have an approximate knowledge and guess a current optimal model. From this model, change p, q, P, Q by 1 once, and choose that with least criterion as the new optimal model. Repeat such operation until no optimal model can be found. Such a search based on the assumption that the criterion information function of orders (p, q, P, Q) is convex, which is guaranteed usually. This strategy is called **stepwise search**.

Comparably, exhaustive search costs much more time, but yields more accurate result, and is more easily to parallelize. On the other side, stepwise search is more efficient, but its result may not be the optimal.

The constant part must be tested via the criterion information to decide whether to include it or not, when d + D is not larger than 1. In other situations, the constant part is excluded.

Subsequently, one can use functions such as PAL_ARIMA_FORECAST, which is described in the ARIMA topic, to make forecast.

### Prerequisite

- The input data should not have missing value.
- The ID column must not have any identical value.
- For ARIMAX, the valid covariates' names in training should also be present in the forecast input data.

### _SYS_AFL.PAL_AUTOARIMA

This function automatically determines the ARIMA orders, and gives the optimal model.

### Overview of all Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<MODEL table>` | OUT |
| 4 | `<FIT table>` | OUT |

### Input Table

| Table | Column | Reference for Output Table | Data Type | Description | Constraint |
|---|---|---|---|---|---|
| DATA | 1st column | ID | INTEGER | Time stamp | No identical value is allowed. |
| | 2nd column | | INTEGER, DOUBLE, or DECIMAL(p, s) | Raw data | |
| | (For auto ARIMAX only) Other columns | | INTEGER, DOUBLE, DECIMAL(p, s), VARCHAR, or NVARCHAR | External (Intervention) data | |

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| SEASONAL_PERIOD | INTEGER | -1 | Value of the seasonal period.<br>• Negative: Automatically identify seasonality by means of auto-correlation scheme.<br>• 0 or 1: Non-seasonal.<br>• Others: Seasonal period. | |
| SEASONALITY_CRITERION | DOUBLE | 0.2 | The criterion of the auto-correlation coefficient for accepting seasonality, in the range of (0, 1). The larger it is, the less probable a time series is regarded to be seasonal. | Valid only when SEASONAL_PERIOD is negative. Refer to *Seasonality Test* for more information. |
| D | INTEGER | -1 | Order of first-differencing.<br>•<br>• Others: Uses the specified value as the first-differencing order.Negative: Automatically identifies first-differencing order with KPSS test. | |
| KPSS_SIGNIFICANCE_LEVEL | DOUBLE | 0.05 | The significance level for KPSS test. Supported values are 0.01, 0.025, 0.05, and 0.1. The smaller it is, the larger probable a time series is considered as first-stationary, that is, the less probable it needs first-differencing. | Valid only when D is negative. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| MAX_D | INTEGER | 2 | The maximum value of D when KPSS test is applied. | |
| SEASONAL_D | INTEGER | -1 | Order of seasonal-differencing.<br>• Negative: Automatically identifies seasonal-differencing order Canova-Hansen test<br>• Others: Uses the specified value as the seasonal-differencing order. | |
| CH_SIGNIFICANCE_LEVEL | DOUBLE | 0.05 | The significance level for Canova-Hansen test. Supported values are 0.01, 0.025, 0.05, 0.1, and 0.2. The smaller it is, the larger probable a time series is considered seasonal-stationary, that is, the less probable it needs seasonal-differencing. | Valid only when SEASONAL_D is negative. |
| MAX_SEASONAL_D | INTEGER | 1 | The maximum value of SEASONAL_D when Canova-Hansen test is applied. | |
| MAX_P | INTEGER | 5 | The maximum value of AR order p. | |
| MAX_Q | INTEGER | 5 | The maximum value of MA order q. | |
| MAX_SEASONAL_P | INTEGER | 2 | Negative: Automatically identifies first-differencing orderThe maximum value of SAR order P. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| MAX_SEASONAL_Q | INTEGER | 2 | The maximum value of SMA order Q. | |
| INFORMATION_CRITERION | INTEGER | 0 | The information criterion for order selection.<br>• 0: AICC<br>• 1: AIC<br>• 2: BIC | |
| SEARCH_STRATEGY | INTEGER | 1 | The search strategy for optimal ARMA model.<br>• 0: Exhaustive traverse.<br>• 1: Stepwise traverse. | |
| MAX_ORDER | INTEGER | 15 | The maximum value of (p + q + P + Q). | Valid only when SEARCH_STRATEGY is 0. |
| INITIAL_P | INTEGER | 0 | Order p of user-defined initial model. | Valid only when SEARCH_STRATEGY is 1. |
| INITIAL_Q | INTEGER | 0 | Order q of user-defined initial model. | Valid only when SEARCH_STRATEGY is 1. |
| INITIAL_SEASONAL_P | INTEGER | 0 | Order P of user-defined initial model. | Valid only when SEARCH_STRATEGY is 1. |
| INITIAL_SEASONAL_Q | INTEGER | 0 | Order Q of user-defined initial model. | Valid only when SEARCH_STRATEGY is 1. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| GUESS_STATES | INTEGER | 1 | If employing ACF/PACF to guess initial ARMA models, besides user-defined model:<br>• 0: No guess. Besides user-defined model, uses states (2, 2) $(1, 1)_m$, (1, 0) $(1, 0)_m$, and (0, 1) $(0, 1)_m$ meanwhile as starting states.<br>• 1: Guesses starting states taking advantage of ACF/PACF. | Valid only when SEARCH_STRATEGY is 1. |
| MAX_SEARCH_ITERATIONS | INTEGER | (MAX_P+1)*<br>(MAX_Q+1)*<br>(MAX_SEASONAL_P+1)*<br>(MAX_SEASONAL_Q+1) | The maximum iterations for searching optimal ARMA states. | Valid only when SEARCH_STRATEGY is 1. |
| METHOD | INTEGER | 2 | The object function for numeric optimization:<br>• 0: Uses the conditional sum of squares (CSS).<br>• 1: Uses the maximized likelihood estimation (MLE).<br>• 2: Firstly uses CSS to approximate starting values, and then uses MLE to fit (CSS-MLE). | |
| ALLOW_LINEAR | INTEGER | 1 | Controls whether to check linear model ARMA(0,0)$(0,0)_m$.<br>• 0: No<br>• 1: Yes | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| FORECAST_METHOD | INTEGER | 1 | Store information for the subsequent forecast method.<br>• 0: Formula forecast<br>• 1: Innovations algorithm | |
| OUTPUT_FITTED | INTEGER | 1 | If output fitted and residuals<br>• 0: not output<br>• 1: output | |
| THREAD_RATIO | DOUBLE | -1 | The ratio of available threads.<br>• 0: single thread<br>• 0~1: percentage<br>• Others: heuristically determined | |
| DEPENDENT_VARIABLE | VARCHAR or NVARCHAR | The second column's name | The dependent variable, that is. time series. | Valid only for ARIMAX; cannot be the first column name (time stamp column). |
| EXCLUDED_FEATURE | VARCHAR or NVARCHAR | No default value | Excludes the indicated feature column. | Valid only for ARIMAX; cannot be the first column name (time stamp column) and the column name of dependent variable. |
| CATEGORICAL_VARIABLE | VARCHAR or NVARCHAR | Detected from input data | Indicates which variables are treated as categorical. The default behaviour is:<br>• string: categorical<br>• integer and double: continuous | Valid only for integer or string column. |

## Output Tables

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| MODEL | 1st column | NVARCHAR(100) | KEY | Name |
| | 2nd column | NVARCHAR(5000) | VALUE | Value |
| FIT | 1st column | ID (in input table) column name | ID (in input table) column name | Time stamp |
| | 2nd column | DOUBLE | FITTED | Fitted values |
| | 3rd column | DOUBLE | RESIDUALS | Residuals values |

> **Note**
>
> The first d + s * D values of FITTED and RESIDUALS are absent.

### Examples

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

### Example 1: Auto ARIMA

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_ARIMA_DATA_TBL;
CREATE COLUMN TABLE PAL_ARIMA_DATA_TBL ("TIMESTAMP" INTEGER, "Y" DOUBLE);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(1   , -24.525 );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(2   , 34.72   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(3   , 57.325  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(4   , 10.34   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(5   , -12.89  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(6   , 39.045  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(7   , 57.3    );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(8   , 6.735   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(9   , -19.365 );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(10  , 34.085  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(11  , 52.455  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(12  , 8.445   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(13  , -13.595 );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(14  , 36.73   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(15  , 54.81   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(16  , 4.625   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(17  , -15.595 );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(18  , 36.61   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(19  , 58.51   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(20  , 6.725   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(21  , -9.815  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(22  , 38.65   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(23  , 55.5    );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(24  , 12.415  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(25  , -17.28  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(26  , 36.105  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(27  , 50.43   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(28  , 7.17    );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(29  , -18.97  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(30  , 39.775  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(31  , 57.825  );
```

```
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(32  , 0.49    );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(33  , -19.475 );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(34  , 31.53   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(35  , 50.025  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(36  , 6.47    );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(37  , -20.585 );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(38  , 36.94   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(39  , 54.37   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(40  , 10.705  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(41  , -15.965 );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(42  , 33.415  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(43  , 55.41   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(44  , -0.62   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(45  , -24.52  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(46  , 37.345  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(47  , 59.44   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(48  , 3.91    );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(49  , -19.305 );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(50  , 39.525  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(51  , 55.545  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(52  , 3.96    );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(53  , -21.69  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(54  , 33.255  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(55  , 57.795  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(56  , 7.535   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(57  , -21.865 );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(58  , 36.89   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(59  , 52.17   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(60  , 0.94    );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(61  , -23.82  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(62  , 35.025  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(63  , 50.5    );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(64  , 4.29    );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(65  , -15.27  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(66  , 36.335  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(67  , 51.435  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(68  , 3.365   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(69  , -25.535 );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(70  , 33.425  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(71  , 52.785  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(72  , 3.95    );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(73  , -17.735 );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(74  , 31.95   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(75  , 53.555  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(76  , 6.745   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(77  , -20     );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(78  , 31.355  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(79  , 54.71   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(80  , 3.835   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(81  , -23.145 );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(82  , 35.23   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(83  , 55.1    );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(84  , 3.73    );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(85  , -17.605 );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(86  , 33.9    );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(87  , 55.42   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(88  , 3.505   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(89  , -23.895 );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(90  , 34.445  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(91  , 55.635  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(92  , 3.595   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(93  , -21.8   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(94  , 33.45   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(95  , 56.54   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(96  , 5.775   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(97  , -21.27  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(98  , 32.14   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(99  , 53.46   );
```

```
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(100 , -1.935  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(101 , -14.275 );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(102 , 35.485  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(103 , 57.305  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(104 , 7.325   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(105 , -21.545 );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(106 , 34.11   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(107 , 53.885  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(108 , 1.625   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(109 , -15.75  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(110 , 36.58   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(111 , 54.89   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(112 , 3.8     );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(113 , -18.035 );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(114 , 37.495  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(115 , 51.875  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(116 , -6.58   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(117 , -18.17  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(118 , 33.565  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(119 , 52.605  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(120 , 3.665   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(121 , -26.47  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(122 , 31.495  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(123 , 52.375  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(124 , 2.135   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(125 , -19.87  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(126 , 36.46   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(127 , 53.11   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(128 , 6.64    );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(129 , -21.835 );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(130 , 34.625  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(131 , 54.61   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(132 , -5.11   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(133 , -17.395 );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(134 , 36.685  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(135 , 52.68   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(136 , 1.105   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(137 , -25.42  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(138 , 32.595  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(139 , 51.615  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(140 , 8.525   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(141 , -15.215 );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(142 , 34.455  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(143 , 56.165  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(144 , 2.335   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(145 , -19.575 );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(146 , 34.445  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(147 , 53.48   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(148 , 7.025   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(149 , -19.655 );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(150 , 32.355  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(151 , 53.41   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(152 , 1.64    );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(153 , -17.325 );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(154 , 37.505  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(155 , 50.875  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(156 , 1.555   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(157 , -16.195 );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(158 , 38.485  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(159 , 53.795  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(160 , 2.23    );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(161 , -17.235 );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(162 , 36.255  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(163 , 55.895  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(164 , 2.025   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(165 , -19.19  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(166 , 35.215  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(167 , 54.215  );
```

```
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(168 , -5.73   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(169 , -9.995  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(170 , 35.27   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(171 , 54.665  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(172 , 7.615   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(173 , -14.625 );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(174 , 40.195  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(175 , 54.13   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(176 , 3.225   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(177 , -19.245 );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(178 , 32.695  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(179 , 51.16   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(180 , 1.265   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(181 , -22.03  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(182 , 35.295  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(183 , 55.21   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(184 , 1.97    );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(185 , -27.47  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(186 , 35.83   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(187 , 53.405  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(188 , 6.855   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(189 , -15.515 );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(190 , 31.675  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(191 , 54.205  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(192 , 1.505   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(193 , -23.06  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(194 , 30.915  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(195 , 54.07   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(196 , 4.785   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(197 , -18.9   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(198 , 31.5    );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(199 , 52.565  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(200 , 2.895   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(201 , -10.22  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(202 , 39.195  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(203 , 56.27   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(204 , 9.155   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(205 , -17.245 );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(206 , 37.905  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(207 , 59.035  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(208 , 7.17    );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(209 , -15.815 );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(210 , 31.77   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(211 , 51.58   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(212 , 1.74    );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(213 , -18.805 );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(214 , 35.875  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(215 , 56.17   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(216 , 14.525  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(217 , -10.39  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(218 , 30.98   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(219 , 58.91   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(220 , 1.465   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(221 , -25.78  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(222 , 29.75   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(223 , 56.385  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(224 , 7.5     );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(225 , -22.755 );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(226 , 31.735  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(227 , 53.655  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(228 , 4.825   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(229 , -20.685 );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(230 , 35.48   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(231 , 58.655  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(232 , 7.605   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(233 , -11.89  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(234 , 36.8    );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(235 , 55.04   );
```

```
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(236 , 12.16   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(237 , -19.905 );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(238 , 34.95   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(239 , 55.69   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(240 , 7.225   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(241 , -15.38  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(242 , 35.365  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(243 , 54.855  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(244 , 5.235   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(245 , -19.81  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(246 , 33.12   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(247 , 53.27   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(248 , 6.525   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(249 , -8.875  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(250 , 39.43   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(251 , 58.28   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(252 , 5.665   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(253 , -19.31  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(254 , 34.25   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(255 , 58.675  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(256 , 12.91   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(257 , -7      );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(258 , 38.5    );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(259 , 58.895  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(260 , 8.65    );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(261 , -13.97  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(262 , 35.015  );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(263 , 56.47   );
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(264 , 3.535   );
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL ( "NAME" VARCHAR (50),"INT_VALUE"
INTEGER,"DOUBLE_VALUE" DOUBLE,"STRING_VALUE" VARCHAR (100));
INSERT INTO PAL_PARAMETER_TBL VALUES ('SEARCH_STRATEGY', 1,NULL,NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('ALLOW_LINEAR', 1, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 1.0, NULL);
DROP TABLE PAL_ARIMA_MODEL_TBL;  -- for the forecast followed
CREATE COLUMN TABLE PAL_ARIMA_MODEL_TBL ("KEY" NVARCHAR(100), "VALUE"
NVARCHAR(5000));
do begin
    lt_data = select * from PAL_ARIMA_DATA_TBL;
    lt_control = select * from PAL_PARAMETER_TBL;
    CALL _SYS_AFL.PAL_AUTOARIMA(:lt_data, :lt_control, lt_model, lt_fit);
    INSERT INTO PAL_ARIMA_MODEL_TBL SELECT * FROM :lt_model;
    SELECT * FROM PAL_ARIMA_MODEL_TBL;
    SELECT* FROM :lt_fit;
end;
```

### Expected Result

MODEL:

| | KEY | VALUE |
|---|---|---|
| 1 | p | 1 |
| 2 | AR | 0.255777 |
| 3 | d | 0 |
| 4 | q | 0 |
| 5 | MA | |
| 6 | s | 4 |
| 7 | P | 1 |
| 8 | SAR | -0.746053 |
| 9 | D | 1 |
| 10 | Q | 2 |
| 11 | SMA | -0.251103;-0.53074 |
| 12 | sigma^2 | 10.8814 |
| 13 | log-likelihood | -682.439 |
| 14 | AIC | 1374.87 |
| 15 | AICc | 1375.11 |
| 16 | BIC | 1392.68 |
| 17 | dy(n-p:n-1)_aux | |
| 18 | dy_aux | 0 |
| 19 | dy_0 | 11.6349;4.325;-0.025;-3.60... |
| 20 | x(n-d:n-1)_aux | |
| 21 | y(n-d:n-1)_aux | 0 |
| 22 | y(n-d:n-1)_0 | -13.97;35.015;56.47;3.535 |
| 23 | epsilon(n-q:n-1)_aux | 0 |
| 24 | epsilon(n-q:n-1)_0 | 7.13752;0.794608;2.24808;... |

FIT:

| | TIMESTAMP | FITTED | RESIDUALS |
|---|---|---|---|
| 1 | 1 | ? | ? |
| 2 | 2 | ? | ? |
| 3 | 3 | ? | ? |
| 4 | 4 | ? | ? |
| 5 | 5 | -24.525 | 11.634999999999998 |
| 6 | 6 | 37.58393114616126 | 1.4610688538387424 |
| 7 | 7 | 57.992624310436426 | -0.6926243104364291 |
| 8 | 8 | 8.622870602048886 | -1.8878706020488851 |
| 9 | 9 | -20.325920769416687 | 0.9609207694166892 |
| 10 | 10 | 36.52819408038735 | -2.44319408038735 |
| 11 | 11 | 56.59589508541181 | -4.14089508541181 |
| 12 | 12 | 7.202374765196849 | 1.2426252348031515 |
| 13 | 13 | -17.502194728078386 | 3.907194728078387 |
| 14 | 14 | 37.738185108701686 | -1.008185108701685 |
| 15 | 15 | 56.10081298901723 | -1.2908129890172244 |
| 16 | 16 | 7.184968179320091 | -2.5599681793200912 |
| 17 | 17 | -19.301194350653738 | 3.7061943506537363 |
| 18 | 18 | 36.447754739052826 | 0.16224526094717362 |
| 19 | 19 | 55.33718295029 | 3.1728170497099946 |
| 20 | 20 | 8.587827360724397 | -1.8628273607243981 |
| 21 | 21 | -16.554026902154163 | 6.739026902154162 |
| 22 | 22 | 38.20849403807501 | 0.4415059619249919 |
| 23 | 23 | 56.29033122933757 | -0.7903312293375646 |
| 24 | 24 | 6.446427215835893 | 5.968572784164106 |
| 25 | 25 | -15.310967329108024 | -1.969032670891977 |
| 26 | 26 | 36.16049963597254 | -0.05549963597254237 |
| 27 | 27 | 56.115180308600756 | -5.68518030860076 |
| 28 | 28 | 6.0529352961770275 | 1.1170647038229724 |

### Example 2: Auto ARIMAX

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_ARIMA_DATA_TBL;
CREATE COLUMN TABLE PAL_ARIMA_DATA_TBL ("TIMESTAMP" INTEGER, "Y" DOUBLE, "X"
DOUBLE);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(1, 1.2, 0.8);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(2, 1.34845613096197, 1.2);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(3, 1.32261090809898, 1.34845613096197);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(4, 1.38095306748554, 1.32261090809898);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(5, 1.54066648969168, 1.38095306748554);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(6, 1.50920806756785, 1.54066648969168);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(7, 1.48461408893443, 1.50920806756785);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(8, 1.43784887380224, 1.48461408893443);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(9, 1.64251548718992, 1.43784887380224);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(10, 1.74292337447476, 1.64251548718992);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(11, 1.91137546943257, 1.74292337447476);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(12, 2.07735796176367, 1.91137546943257);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(13, 2.01741246166924, 2.07735796176367);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(14, 1.87176938196573, 2.01741246166924);
```

```
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(15, 1.83354723357744, 1.87176938196573);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(16, 1.66104978144571, 1.83354723357744);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(17, 1.65115984070812, 1.66104978144571);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(18, 1.69470966154593, 1.65115984070812);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(19, 1.70459802935728, 1.69470966154593);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(20, 1.61246059980916, 1.70459802935728);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(21, 1.53949706614636, 1.61246059980916);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(22, 1.59231354902055, 1.53949706614636);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(23, 1.81741927705578, 1.59231354902055);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(24, 1.80224252773564, 1.81741927705578);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(25, 1.81881576781466, 1.80224252773564);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(26, 1.78089755157948, 1.81881576781466);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(27, 1.61473635574416, 1.78089755157948);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(28, 1.42002147867225, 1.61473635574416);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES(29, 1.49971641345022, 1.42002147867225);
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL ( "NAME" VARCHAR (50),"INT_VALUE"
INTEGER,"DOUBLE_VALUE" DOUBLE,"STRING_VALUE" VARCHAR (100));
INSERT INTO PAL_PARAMETER_TBL VALUES ('SEARCH_STRATEGY', 1,NULL,NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('ALLOW_LINEAR', 1, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 1.0, NULL);
DROP TABLE PAL_ARIMA_MODEL_TBL;  -- for the forecast followed
CREATE COLUMN TABLE PAL_ARIMA_MODEL_TBL ("KEY" NVARCHAR(100), "VALUE"
NVARCHAR(5000));
do begin
    lt_data = select * from PAL_ARIMA_DATA_TBL;
    lt_control = select * from PAL_PARAMETER_TBL;
    CALL _SYS_AFL.PAL_AUTOARIMA(:lt_data, :lt_control, lt_model, lt_fit);
    INSERT INTO PAL_ARIMA_MODEL_TBL SELECT * FROM :lt_model;
    SELECT * FROM PAL_ARIMA_MODEL_TBL;
    SELECT* FROM :lt_fit;
end;
```

### Expected Result

MODEL:

| | KEY | VALUE |
|---|---|---|
| 1 | p | 0 |
| 2 | AR | |
| 3 | d | 0 |
| 4 | q | 1 |
| 5 | MA | 0.463624 |
| 6 | s | 13 |
| 7 | P | 0 |
| 8 | SAR | |
| 9 | D | 0 |
| 10 | Q | 0 |
| 11 | SMA | |
| 12 | mu | 0.642409 |
| 13 | regressor | X |
| 14 | beta | 0.618268 |
| 15 | sigma^2 | 0.00898333 |
| 16 | log-likelihood | 27.0593 |
| 17 | AIC | -46.1187 |
| 18 | AICc | -44.452 |
| 19 | BIC | -40.6495 |
| 20 | dy(n-p:n-1)_aux | |
| 21 | dy_aux | 0 |
| 22 | dy_0 | 0.705385;0.606534;0... |

FIT:

| | TIMESTAMP | FITTED | RESIDUALS |
|---|---|---|---|
| 1 | 1 | 1.137023662035878 | 0.06297633796412189 |
| 2 | 2 | 1.408362779568391 | -0.05990664860642113 |
| 3 | 3 | 1.4493600203397872 | -0.12674911224080707 |
| 4 | 4 | 1.4018325455689025 | -0.020879478083362615 |
| 5 | 5 | 1.4865445176037024 | 0.05412197208797759 |
| 6 | 6 | 1.6200375106539509 | -0.11082944308610082 |
| 7 | 7 | 1.5241252515110801 | -0.039511162576650126 |
| 8 | 8 | 1.5419808089686218 | -0.10413193516638183 |
| 9 | 9 | 1.4831074476247528 | 0.15940803956516703 |
| 10 | 10 | 1.7318297096294515 | 0.011093664845308646 |
| 11 | 11 | 1.7251465882868535 | 0.18622888114571642 |
| 12 | 12 | 1.9104921690751635 | 0.16686579268850674 |
| 13 | 13 | 2.0041366809006935 | 0.013275780768546241 |
| 14 | 14 | 1.8958661728961497 | -0.02409679093041965 |
| 15 | 15 | 1.7884928304873744 | 0.04505440309006566 |
| 16 | 16 | 1.796921481713965 | -0.13587170026825507 |
| 17 | 17 | 1.6063899849769134 | 0.04476985573120666 |
| 18 | 18 | 1.6840252153237028 | 0.010684446222227341 |
| 19 | 19 | 1.6951478550158823 | 0.009450174341397652 |
| 20 | 20 | 1.7006892805737406 | -0.08822868076458074 |
| 21 | 21 | 1.598437309616289 | -0.058940243469928884 |
| 22 | 22 | 1.5669051094417148 | 0.025408439578835214 |

## _SYS_AFL.PAL_MASSIVE_AUTOARIMA

This function is the MASSIVE version of _SYS_AFL.PAL_AUTOARIMA, by which multiple time series can be analyzed in a single call.

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<MODEL table>` | OUT |
| 4 | `<FIT table>` | OUT |
| 5 | `<ERROR_MSG table>` | OUT |

### Input Table

| Table | Column | Data Type | Description |
|---|---|---|---|
| DATA | 1st column | INTEGER, VRACHAR or NVARCHAR | Group ID. |
| | 2nd column | INTEGER | Time stamp. Reference for Output Table is "ID" |
| | 3rd column | INTEGER, DOUBLE, or DECIMAL(p,s) | Raw data. |
| | (For ARIMAX only) Other columns | INTEGER, DOUBLE, DECIMAL(p,s), VARCHAR, or NVARCHAR | External (Intervention) data. |

### Parameter Table

Same as _SYS_AFL.PAL_AUTOARIMA.

### Output Table

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| MODEL | 1st column | VARCHAR | GROUP_ID | Group ID. |
| | 2nd column | VARCHAR or NVARCHAR | ROW_INDEX | Model key. |
| | 3rd column | VARCHAR or NVARCHAR | MODEL_CONTENT | Model value. |
| FIT | 1st column | VARCHAR | GROUP_ID | Group ID. |
| | 2nd column | INTEGER | ID (in input table) column name | Time stamp. |
| | 3rd column | DOUBLE | FITTED | Fitted values. |
| | 4th column | DOUBLE | RESIDUALS | Residuals values. |
| ERROR_MSG | 1st column | VARCHAR | GROUP_ID | Group ID. |
| | 2nd column | VARCHAR | ERROR_TIMESTAMP | Time stamp when the error occured. |
| | 3rd column | INTEGER | ERROR_CODE | Error code |
| | 4th column | VARCHAR | ERROR_MESSAGE | Error message. |

### Examples

Assume that:

- DM_PAL is a schema belonging to USER1;
- USER1 has been assigned the AFL__SYS_AFL_AFLPAL_EXECUTE or AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION role.

### Example 1: MASSIVE Auto ARIMA

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_MASSIVE_AUTOARIMA_DATA_TBL;
CREATE COLUMN TABLE PAL_MASSIVE_AUTOARIMA_DATA_TBL (
    GROUP_ID  NVARCHAR(40),
    TIMESTAMP INTEGER,
```

```
    Y         DOUBLE
);
DROP TABLE PAL_MASSIVE_AUTOARIMA_PARAM_TBL;
CREATE COLUMN TABLE PAL_MASSIVE_AUTOARIMA_PARAM_TBL (
    GROUP_ID   NVARCHAR(40),
    NAME       NVARCHAR(100),
    INTARGS    INTEGER,
    DOUBLEARGS DOUBLE,
    STRINGARGS NVARCHAR(100)
);
DROP TABLE PAL_MASSIVE_AUTOARIMA_MODEL_TBL;
CREATE COLUMN TABLE PAL_MASSIVE_AUTOARIMA_MODEL_TBL (
    GROUP_ID NVARCHAR(40),
    KEY      NVARCHAR(100),
    VALUE    NVARCHAR(5000)
);
DROP TABLE PAL_MASSIVE_AUTOARIMA_FIT_TBL;
CREATE COLUMN TABLE PAL_MASSIVE_AUTOARIMA_FIT_TBL (
    GROUP_ID   NVARCHAR(40),
    TIMESTAMP INTEGER,
    FITTED    DOUBLE,
    RESIDUAL  DOUBLE
);
DROP TABLE PAL_MASSIVE_AUTOARIMA_ERROR_TBL;
CREATE COLUMN TABLE PAL_MASSIVE_AUTOARIMA_ERROR_TBL (
    GROUP_ID        NVARCHAR(40),
    ERROR_TIMESTAMP NVARCHAR(100),
    ERRORCODE       INTEGER,
    ERROR_MESSAGE   NVARCHAR(200)
);
INSERT INTO PAL_MASSIVE_AUTOARIMA_DATA_TBL VALUES ('GROUP_A', 1, -0.636126431 );
INSERT INTO PAL_MASSIVE_AUTOARIMA_DATA_TBL VALUES ('GROUP_A', 2, 3.092508651  );
INSERT INTO PAL_MASSIVE_AUTOARIMA_DATA_TBL VALUES ('GROUP_A', 3, -0.73733556  );
INSERT INTO PAL_MASSIVE_AUTOARIMA_DATA_TBL VALUES ('GROUP_A', 4, -3.142190983 );
INSERT INTO PAL_MASSIVE_AUTOARIMA_DATA_TBL VALUES ('GROUP_A', 5, 2.088819813  );
INSERT INTO PAL_MASSIVE_AUTOARIMA_DATA_TBL VALUES ('GROUP_A', 6, 3.179302734  );
INSERT INTO PAL_MASSIVE_AUTOARIMA_DATA_TBL VALUES ('GROUP_A', 7, -0.871376102 );
INSERT INTO PAL_MASSIVE_AUTOARIMA_DATA_TBL VALUES ('GROUP_A', 8, -3.475633275 );
INSERT INTO PAL_MASSIVE_AUTOARIMA_DATA_TBL VALUES ('GROUP_A', 9, 1.779244219  );
INSERT INTO PAL_MASSIVE_AUTOARIMA_DATA_TBL VALUES ('GROUP_A', 10, 3.609159416 );
INSERT INTO PAL_MASSIVE_AUTOARIMA_DATA_TBL VALUES ('GROUP_A', 11, 0.082170143 );
INSERT INTO PAL_MASSIVE_AUTOARIMA_DATA_TBL VALUES ('GROUP_A', 12, -4.42439631 );
INSERT INTO PAL_MASSIVE_AUTOARIMA_DATA_TBL VALUES ('GROUP_A', 13, 0.499210261 );
INSERT INTO PAL_MASSIVE_AUTOARIMA_DATA_TBL VALUES ('GROUP_A', 14, 4.514017351 );
INSERT INTO PAL_MASSIVE_AUTOARIMA_DATA_TBL VALUES ('GROUP_A', 15, -0.320607187);
INSERT INTO PAL_MASSIVE_AUTOARIMA_DATA_TBL VALUES ('GROUP_A', 16, -3.70219307 );
INSERT INTO PAL_MASSIVE_AUTOARIMA_DATA_TBL VALUES ('GROUP_B', 1, -0.636126431 );
INSERT INTO PAL_MASSIVE_AUTOARIMA_DATA_TBL VALUES ('GROUP_B', 2, 3.092508651  );
INSERT INTO PAL_MASSIVE_AUTOARIMA_DATA_TBL VALUES ('GROUP_B', 3, -0.73733556  );
INSERT INTO PAL_MASSIVE_AUTOARIMA_DATA_TBL VALUES ('GROUP_B', 4, -3.142190983 );
INSERT INTO PAL_MASSIVE_AUTOARIMA_DATA_TBL VALUES ('GROUP_B', 5, 2.088819813  );
INSERT INTO PAL_MASSIVE_AUTOARIMA_DATA_TBL VALUES ('GROUP_B', 6, 3.179302734  );
INSERT INTO PAL_MASSIVE_AUTOARIMA_DATA_TBL VALUES ('GROUP_B', 7, -0.871376102 );
INSERT INTO PAL_MASSIVE_AUTOARIMA_DATA_TBL VALUES ('GROUP_B', 8, -3.475633275 );
INSERT INTO PAL_MASSIVE_AUTOARIMA_DATA_TBL VALUES ('GROUP_B', 9, 1.779244219  );
INSERT INTO PAL_MASSIVE_AUTOARIMA_DATA_TBL VALUES ('GROUP_B', 10, 3.609159416 );
INSERT INTO PAL_MASSIVE_AUTOARIMA_DATA_TBL VALUES ('GROUP_B', 11, 0.082170143 );
INSERT INTO PAL_MASSIVE_AUTOARIMA_DATA_TBL VALUES ('GROUP_B', 12, -4.42439631 );
INSERT INTO PAL_MASSIVE_AUTOARIMA_DATA_TBL VALUES ('GROUP_B', 13, 0.499210261 );
INSERT INTO PAL_MASSIVE_AUTOARIMA_DATA_TBL VALUES ('GROUP_B', 14, 4.514017351 );
INSERT INTO PAL_MASSIVE_AUTOARIMA_DATA_TBL VALUES ('GROUP_B', 15, -0.320607187);
INSERT INTO PAL_MASSIVE_AUTOARIMA_DATA_TBL VALUES ('GROUP_B', 16, -3.70219307 );
INSERT INTO PAL_MASSIVE_AUTOARIMA_PARAM_TBL VALUES ('GROUP_A',
'SEARCH_STRATEGY', 1,NULL,NULL);
INSERT INTO PAL_MASSIVE_AUTOARIMA_PARAM_TBL VALUES ('GROUP_A', 'ALLOW_LINEAR',
1, NULL, NULL);
INSERT INTO PAL_MASSIVE_AUTOARIMA_PARAM_TBL VALUES ('GROUP_A', 'THREAD_RATIO',
NULL, 1.0, NULL);
```

```
INSERT INTO PAL_MASSIVE_AUTOARIMA_PARAM_TBL VALUES ('GROUP_B',
'SEARCH_STRATEGY', 1,NULL,NULL);
INSERT INTO PAL_MASSIVE_AUTOARIMA_PARAM_TBL VALUES ('GROUP_B', 'ALLOW_LINEAR',
1, NULL, NULL);
INSERT INTO PAL_MASSIVE_AUTOARIMA_PARAM_TBL VALUES ('GROUP_B', 'THREAD_RATIO',
NULL, 1.0, NULL);
do begin
    lt_data = select * from PAL_MASSIVE_AUTOARIMA_DATA_TBL;
    lt_control = select * from PAL_MASSIVE_AUTOARIMA_PARAM_TBL;
    CALL _SYS_AFL.PAL_MASSIVE_AUTOARIMA(:lt_data, :lt_control, lt_model, lt_fit,
lt_error);
    INSERT INTO PAL_MASSIVE_AUTOARIMA_MODEL_TBL SELECT * FROM :lt_model;
    INSERT INTO PAL_MASSIVE_AUTOARIMA_FIT_TBL SELECT * FROM :lt_fit;
    INSERT INTO PAL_MASSIVE_AUTOARIMA_ERROR_TBL SELECT * FROM :lt_error;
end;
select * from PAL_MASSIVE_AUTOARIMA_MODEL_TBL;
select * from PAL_MASSIVE_AUTOARIMA_FIT_TBL;
select * from PAL_MASSIVE_AUTOARIMA_ERROR_TBL;
```

### Expected Result

MODEL table:

| | GROUP_ID | KEY | VALUE |
|---|---|---|---|
| 1 | GROUP_A | p | 0 |
| 2 | GROUP_A | AR | |
| 3 | GROUP_A | d | 0 |
| 4 | GROUP_A | q | 2 |
| 5 | GROUP_A | MA | -0.0133823;-0.848042 |
| 6 | GROUP_A | s | 4 |
| 7 | GROUP_A | P | 1 |
| 8 | GROUP_A | SAR | 0.89674 |
| 9 | GROUP_A | D | 0 |
| 10 | GROUP_A | Q | 0 |
| 11 | GROUP_A | SMA | |
| 12 | GROUP_A | sigma^2 | 0.710826 |
| 13 | GROUP_A | log-likelihood | -22.407 |
| 14 | GROUP_A | AIC | 52.814 |
| 15 | GROUP_A | AICc | 56.4503 |
| 16 | GROUP_A | BIC | 55.9043 |
| 17 | GROUP_A | dy(n-p:n-1)_aux | |
| 18 | GROUP_A | dy_aux | 0 |
| 19 | GROUP_A | dy_0 | -0.636126;3.0925;-0.737335;-3.14219;2.08881;3.1793;-0.871376;-3.47563;1.77924;3.60915;0.0821701;-4.42439;0.49 |
| 20 | GROUP_A | x(n-d:n-1)_aux | |
| 21 | GROUP_A | y(n-d:n-1)_aux | |
| 22 | GROUP_A | epsilon(n-q:n-1)_aux | 0 |
| 23 | GROUP_A | epsilon(n-q:n-1)_0 | -0.260125;0.66466 |
| 24 | GROUP_B | p | 0 |
| 25 | GROUP_B | AR | |

FIT table:

| | GROUP_ID | TIMESTAMP | FITTED | RESIDUAL |
|---|---|---|---|---|
| 1 | GROUP_A | 1 | 0 | -0.636126431 |
| 2 | GROUP_A | 2 | 0.0007523737885910542 | 3.091756277211409 |
| 3 | GROUP_A | 3 | 0.5880405534790061 | -1.325376113479006 |
| 4 | GROUP_A | 4 | -2.8459244427064063 | -0.29626654029359356 |
| 5 | GROUP_A | 5 | 0.46422899524395866 | 1.6245908177560415 |
| 6 | GROUP_A | 6 | 2.962516529296795 | 0.21678620470320498 |
| 7 | GROUP_A | 7 | -1.9633762402008905 | 1.0920001382008904 |
| 8 | GROUP_A | 8 | -3.0124186600322043 | -0.4632146149677956 |
| 9 | GROUP_A | 9 | 0.9923493042492556 | 0.7868949147507444 |
| 10 | GROUP_A | 10 | 3.2150344707686402 | 0.39412494523135955 |
| 11 | GROUP_A | 11 | -1.436431711184517 | 1.518601854184517 |
| 12 | GROUP_A | 12 | -3.4647395997598345 | -0.9596567102401652 |
| 13 | GROUP_A | 13 | 0.3467901806911708 | 0.1524200803088292 |
| 14 | GROUP_A | 14 | 4.032320487170395 | 0.48169686382960464 |
| 15 | GROUP_A | 15 | -0.06048130779241728 | -0.26012587920758273 |
| 16 | GROUP_A | 16 | -4.366853767028242 | 0.6646606970282418 |
| 17 | GROUP_B | 1 | 0 | -0.636126431 |
| 18 | GROUP_B | 2 | 0.0007523737885910542 | 3.091756277211409 |
| 19 | GROUP_B | 3 | 0.5880405534790061 | -1.325376113479006 |
| 20 | GROUP_B | 4 | -2.8459244427064063 | -0.29626654029359356 |
| 21 | GROUP_B | 5 | 0.46422899524395866 | 1.6245908177560415 |
| 22 | GROUP_B | 6 | 2.962516529296795 | 0.21678620470320498 |
| 23 | GROUP_B | 7 | -1.9633762402008905 | 1.0920001382008904 |
| 24 | GROUP_B | 8 | -3.0124186600322043 | -0.4632146149677956 |
| 25 | GROUP_B | 9 | 0.9923493042492556 | 0.7868949147507444 |

ERROR_MSG table:

Empty

### Example 2: MASSIVE Auto ARIMAX

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_MASSIVE_AUTOARIMAX_DATA_TBL;
CREATE COLUMN TABLE PAL_MASSIVE_AUTOARIMAX_DATA_TBL (
    GROUP_ID  NVARCHAR(40),
    TIMESTAMP INTEGER,
    Y         DOUBLE,
    X1        DOUBLE,
    X2        DOUBLE
);
DROP TABLE PAL_MASSIVE_AUTOARIMAX_PARAM_TBL;
CREATE COLUMN TABLE PAL_MASSIVE_AUTOARIMAX_PARAM_TBL (
    GROUP_ID   NVARCHAR(40),
    NAME       NVARCHAR(100),
    INTARGS    INTEGER,
    DOUBLEARGS DOUBLE,
    STRINGARGS NVARCHAR(100)
);
DROP TABLE PAL_MASSIVE_AUTOARIMAX_MODEL_TBL;
CREATE COLUMN TABLE PAL_MASSIVE_AUTOARIMAX_MODEL_TBL (
    GROUP_ID NVARCHAR(40),
    KEY      NVARCHAR(100),
    VALUE    NVARCHAR(5000)
);
DROP TABLE PAL_MASSIVE_AUTOARIMAX_FIT_TBL;
CREATE COLUMN TABLE PAL_MASSIVE_AUTOARIMAX_FIT_TBL (
    GROUP_ID   NVARCHAR(40),
    TIMESTAMP INTEGER,
    FITTED    DOUBLE,
    RESIDUAL  DOUBLE
);
DROP TABLE PAL_MASSIVE_AUTOARIMAX_ERROR_TBL;
```

```
CREATE COLUMN TABLE PAL_MASSIVE_AUTOARIMAX_ERROR_TBL (
    GROUP_ID        NVARCHAR(40),
    ERROR_TIMESTAMP NVARCHAR(100),
    ERRORCODE       INTEGER,
    ERROR_MESSAGE   NVARCHAR(200)
);
INSERT INTO PAL_MASSIVE_AUTOARIMAX_DATA_TBL VALUES ('GROUP_A', 1, 0.882, 1.1,
-0.992 );
INSERT INTO PAL_MASSIVE_AUTOARIMAX_DATA_TBL VALUES ('GROUP_A', 2, -0.077, 1.155,
-1.212 );
INSERT INTO PAL_MASSIVE_AUTOARIMAX_DATA_TBL VALUES ('GROUP_A', 3, 0.135, 0.53,
-1.034 );
INSERT INTO PAL_MASSIVE_AUTOARIMAX_DATA_TBL VALUES ('GROUP_A', 4, 0.373, 0.698,
-1.195 );
INSERT INTO PAL_MASSIVE_AUTOARIMAX_DATA_TBL VALUES ('GROUP_A', 5, -0.437, 0.441,
-1.386 );
INSERT INTO PAL_MASSIVE_AUTOARIMAX_DATA_TBL VALUES ('GROUP_A', 6, -0.556, 0.405,
-0.844 );
INSERT INTO PAL_MASSIVE_AUTOARIMAX_DATA_TBL VALUES ('GROUP_A', 7, -0.432,
-0.016, -1.001 );
INSERT INTO PAL_MASSIVE_AUTOARIMAX_DATA_TBL VALUES ('GROUP_A', 8, -0.46, 0.271,
-1.234 );
INSERT INTO PAL_MASSIVE_AUTOARIMAX_DATA_TBL VALUES ('GROUP_A', 9, -0.698,
-0.357, -1.269 );
INSERT INTO PAL_MASSIVE_AUTOARIMAX_DATA_TBL VALUES ('GROUP_A', 10, -0.765,
-0.006, -1.326 );
INSERT INTO PAL_MASSIVE_AUTOARIMAX_DATA_TBL VALUES ('GROUP_A', 11, -0.833,
-0.647, -2.124 );
INSERT INTO PAL_MASSIVE_AUTOARIMAX_DATA_TBL VALUES ('GROUP_A', 12, -0.767,
-0.282, -1.615 );
INSERT INTO PAL_MASSIVE_AUTOARIMAX_DATA_TBL VALUES ('GROUP_A', 13, -0.356,
-0.503, -1.035 );
INSERT INTO PAL_MASSIVE_AUTOARIMAX_DATA_TBL VALUES ('GROUP_A', 14, -0.496,
-0.998, -1.356 );
INSERT INTO PAL_MASSIVE_AUTOARIMAX_DATA_TBL VALUES ('GROUP_A', 15, -0.684,
-0.618, -1.06 );
INSERT INTO PAL_MASSIVE_AUTOARIMAX_DATA_TBL VALUES ('GROUP_A', 16, -0.953,
-0.547, -1.437 );
INSERT INTO PAL_MASSIVE_AUTOARIMAX_DATA_TBL VALUES ('GROUP_A', 17, -0.869,
-0.403, -1.36 );
INSERT INTO PAL_MASSIVE_AUTOARIMAX_DATA_TBL VALUES ('GROUP_A', 18, -0.831,
-0.691, -1.553 );
INSERT INTO PAL_MASSIVE_AUTOARIMAX_DATA_TBL VALUES ('GROUP_A', 19, -0.858,
-0.793, -1.302 );
INSERT INTO PAL_MASSIVE_AUTOARIMAX_DATA_TBL VALUES ('GROUP_A', 20, -0.769,
-0.654, -1.332 );
INSERT INTO PAL_MASSIVE_AUTOARIMAX_DATA_TBL VALUES ('GROUP_B', 1, -0.732, 0.309,
-0.366 );
INSERT INTO PAL_MASSIVE_AUTOARIMAX_DATA_TBL VALUES ('GROUP_B', 2, -0.682,
-0.416, 0.034 );
INSERT INTO PAL_MASSIVE_AUTOARIMAX_DATA_TBL VALUES ('GROUP_B', 3, -0.606,
-0.206, -0.129 );
INSERT INTO PAL_MASSIVE_AUTOARIMAX_DATA_TBL VALUES ('GROUP_B', 4, -0.639, 1.17,
-0.284 );
INSERT INTO PAL_MASSIVE_AUTOARIMAX_DATA_TBL VALUES ('GROUP_B', 5, -0.363, 0.858,
-0.292 );
INSERT INTO PAL_MASSIVE_AUTOARIMAX_DATA_TBL VALUES ('GROUP_B', 6, 0.473, 0.281,
-0.508 );
INSERT INTO PAL_MASSIVE_AUTOARIMAX_DATA_TBL VALUES ('GROUP_B', 7, 0.03, 0.599,
0.283 );
INSERT INTO PAL_MASSIVE_AUTOARIMAX_DATA_TBL VALUES ('GROUP_B', 8, -0.688,
-0.102, -0.729 );
INSERT INTO PAL_MASSIVE_AUTOARIMAX_DATA_TBL VALUES ('GROUP_B', 9, -0.963,
-0.288, -0.388 );
INSERT INTO PAL_MASSIVE_AUTOARIMAX_DATA_TBL VALUES ('GROUP_B', 10, -0.879,
-0.503, -0.464 );
INSERT INTO PAL_MASSIVE_AUTOARIMAX_DATA_TBL VALUES ('GROUP_B', 11, -1.132,
-0.05, -0.213 );
```

```
INSERT INTO PAL_MASSIVE_AUTOARIMAX_DATA_TBL VALUES ('GROUP_B', 12, -1.031,
0.264, 0.132 );
INSERT INTO PAL_MASSIVE_AUTOARIMAX_DATA_TBL VALUES ('GROUP_B', 13, -1.014,
-0.059, -0.043 );
INSERT INTO PAL_MASSIVE_AUTOARIMAX_DATA_TBL VALUES ('GROUP_B', 14, -1.026,
0.521, -0.828 );
INSERT INTO PAL_MASSIVE_AUTOARIMAX_DATA_TBL VALUES ('GROUP_B', 15, -1.242,
-0.437, -0.803 );
INSERT INTO PAL_MASSIVE_AUTOARIMAX_DATA_TBL VALUES ('GROUP_B', 16, -1.018,
-0.068, -0.975 );
INSERT INTO PAL_MASSIVE_AUTOARIMAX_DATA_TBL VALUES ('GROUP_B', 17, -0.957,
-0.691, -1.082 );
INSERT INTO PAL_MASSIVE_AUTOARIMAX_DATA_TBL VALUES ('GROUP_B', 18, -0.985,
-0.249, -0.777 );
INSERT INTO PAL_MASSIVE_AUTOARIMAX_DATA_TBL VALUES ('GROUP_B', 19, -0.463,
-0.413, -0.583 );
INSERT INTO PAL_MASSIVE_AUTOARIMAX_DATA_TBL VALUES ('GROUP_B', 20, -0.983,
-0.534, -1.448 );
INSERT INTO PAL_MASSIVE_AUTOARIMAX_PARAM_TBL VALUES ('GROUP_A', 'D', 0, NULL,
NULL);
INSERT INTO PAL_MASSIVE_AUTOARIMAX_PARAM_TBL VALUES ('GROUP_A', 'Q', 1, NULL,
NULL);
INSERT INTO PAL_MASSIVE_AUTOARIMAX_PARAM_TBL VALUES ('GROUP_A',
'SEASONAL_PERIOD', 4, NULL, NULL);
INSERT INTO PAL_MASSIVE_AUTOARIMAX_PARAM_TBL VALUES ('GROUP_A', 'SEASONAL_P', 1,
NULL, NULL);
INSERT INTO PAL_MASSIVE_AUTOARIMAX_PARAM_TBL VALUES ('GROUP_A', 'METHOD', 1,
NULL, NULL);
INSERT INTO PAL_MASSIVE_AUTOARIMAX_PARAM_TBL VALUES ('GROUP_A', 'THREAD_RATIO',
NULL, 1.0, NULL);
INSERT INTO PAL_MASSIVE_AUTOARIMAX_PARAM_TBL VALUES ('GROUP_A',
'BACKGROUND_SIZE', -1, NULL, NULL);
INSERT INTO PAL_MASSIVE_AUTOARIMAX_PARAM_TBL VALUES ('GROUP_B', 'D', 1, NULL,
NULL);
INSERT INTO PAL_MASSIVE_AUTOARIMAX_PARAM_TBL VALUES ('GROUP_B', 'P', 1, NULL,
NULL);
INSERT INTO PAL_MASSIVE_AUTOARIMAX_PARAM_TBL VALUES ('GROUP_B',
'SEASONAL_PERIOD', 4, NULL, NULL);
INSERT INTO PAL_MASSIVE_AUTOARIMAX_PARAM_TBL VALUES ('GROUP_B', 'SEASONAL_P', 0,
NULL, NULL);
INSERT INTO PAL_MASSIVE_AUTOARIMAX_PARAM_TBL VALUES ('GROUP_B', 'METHOD', 1,
NULL, NULL);
INSERT INTO PAL_MASSIVE_AUTOARIMAX_PARAM_TBL VALUES ('GROUP_B', 'THREAD_RATIO',
NULL, 1.0, NULL);
INSERT INTO PAL_MASSIVE_AUTOARIMAX_PARAM_TBL VALUES ('GROUP_B',
'BACKGROUND_SIZE', -1, NULL, NULL);
do begin
    lt_data = select * from PAL_MASSIVE_AUTOARIMAX_DATA_TBL;
    lt_control = select * from PAL_MASSIVE_AUTOARIMAX_PARAM_TBL;
    CALL _SYS_AFL.PAL_MASSIVE_AUTOARIMA(:lt_data, :lt_control, lt_model, lt_fit,
lt_error);
    INSERT INTO PAL_MASSIVE_AUTOARIMAX_MODEL_TBL SELECT * FROM :lt_model;
    INSERT INTO PAL_MASSIVE_AUTOARIMAX_FIT_TBL SELECT * FROM :lt_fit;
    INSERT INTO PAL_MASSIVE_AUTOARIMAX_ERROR_TBL SELECT * FROM :lt_error;
end;
select * from PAL_MASSIVE_AUTOARIMAX_MODEL_TBL;
select * from PAL_MASSIVE_AUTOARIMAX_FIT_TBL;
select * from PAL_MASSIVE_AUTOARIMAX_ERROR_TBL;
```

### Expected Result

MODEL table:

| | GROUP_ID | KEY | VALUE |
|---|---|---|---|
| 1 | GROUP_B | p | 1 |
| 2 | GROUP_B | AR | -0.0453778 |
| 3 | GROUP_B | d | 1 |
| 4 | GROUP_B | q | 0 |
| 5 | GROUP_B | MA | |
| 6 | GROUP_B | s | 4 |
| 7 | GROUP_B | P | 0 |
| 8 | GROUP_B | SAR | |
| 9 | GROUP_B | D | 0 |
| 10 | GROUP_B | Q | 0 |
| 11 | GROUP_B | SMA | |
| 12 | GROUP_B | regressor | X1;X2 |
| 13 | GROUP_B | beta | -0.028055;0.165995 |
| 14 | GROUP_B | sigma^2 | 0.115753 |
| 15 | GROUP_B | log-likelihood | -6.47609 |
| 16 | GROUP_B | AIC | 20.9521 |
| 17 | GROUP_B | AICc | 23.8093 |
| 18 | GROUP_B | BIC | 24.7299 |
| 19 | GROUP_B | dy(n-p:n-1)_aux | |
| 20 | GROUP_B | dy_aux | 0 |
| 21 | GROUP_B | dy_0 | -0.0367381;0.108948;0.031333;0.268574;0.855667;-0.56538;-0.569679;-0.336822;0.0905838;-0.281955;0.0525408; |
| 22 | GROUP_B | x(n-d:n-1)_aux | 0 |
| 23 | GROUP_B | x(n-d:n-1)_0 | -0.534;-1.448 |
| 24 | GROUP_B | y(n-d:n-1)_aux | 0 |
| 25 | GROUP_B | y(n-d:n-1)_0 | -0.983 |

FIT table:

| | GROUP_ID | TIMESTAMP | FITTED | RESIDUAL |
|---|---|---|---|---|
| 1 | GROUP_B | 1 | *NULL* | *NULL* |
| 2 | GROUP_B | 2 | -0.6452618938645001 | -0.03673810613549995 |
| 3 | GROUP_B | 3 | -0.7132817247774011 | 0.10728172477740112 |
| 4 | GROUP_B | 4 | -0.6752769626753176 | 0.03627696267531758 |
| 5 | GROUP_B | 5 | -0.6329966019621556 | 0.2699966019621556 |
| 6 | GROUP_B | 6 | -0.3948545623393839 | 0.8678545623393839 |
| 7 | GROUP_B | 7 | 0.5565525066874544 | -0.5265525066874543 |
| 8 | GROUP_B | 8 | -0.0926649603260653 | -0.5953350396739346 |
| 9 | GROUP_B | 9 | -0.6003264995801652 | -0.36267350041983487 |
| 10 | GROUP_B | 10 | -0.9542995164887476 | 0.07529951648874754 |
| 11 | GROUP_B | 11 | -0.8541546102180018 | -0.2778453897819982 |
| 12 | GROUP_B | 12 | -1.0707463341256336 | 0.03974633412563361 |
| 13 | GROUP_B | 13 | -1.0533715920897049 | 0.039371592089704796 |
| 14 | GROUP_B | 14 | -1.1622567564092412 | 0.13625675640924106 |
| 15 | GROUP_B | 15 | -1.0010802089234625 | -0.24091979107653752 |
| 16 | GROUP_B | 16 | -1.2696940011878082 | 0.2516940011878081 |
| 17 | GROUP_B | 17 | -1.030213180516417 | 0.07321318051641705 |
| 18 | GROUP_B | 18 | -0.9215526531749267 | -0.06344734682507329 |
| 19 | GROUP_B | 19 | -0.9451905605788544 | 0.48219056057885434 |
| 20 | GROUP_B | 20 | -0.6252085019703542 | -0.3577914980296458 |
| 21 | GROUP_A | 1 | 0.20916839811175647 | 0.6728316018882435 |
| 22 | GROUP_A | 2 | 0.11546320543366188 | -0.19246320543366188 |
| 23 | GROUP_A | 3 | -0.07551898263882856 | 0.21051898263882857 |
| 24 | GROUP_A | 4 | -0.0610758796002811 | 0.4340758796002811 |
| 25 | GROUP_A | 5 | -0.3959234469186099 | -0.04107655308139005 |

ERROR_MSG table:

Empty

## Related Information

ARIMA [page 593]
Seasonality Test [page 934]

# 5.5.4 Auto Exponential Smoothing

Auto exponential smoothing (previously named forecast smoothing) is used to calculate optimal parameters of a set of smoothing functions in PAL, including Single Exponential Smoothing, Double Exponential Smoothing, and Triple Exponential Smoothing.

This function also outputs the forecasting results based on these optimal parameters. This optimization is computed by exploring of the parameter space which includes all possible parameter combinations. The quality assessment is done by comparing historic and forecast values. In PAL, MSE (mean squared error) or MAPE (mean absolute percentage error) is used to evaluate the quality of the parameters.

The parameter optimization is based on global and local search algorithms. The global search algorithm used in this function is simulated annealing, whereas the local search algorithm is Nelder Mead. Those algorithms allow for efficient search processes.

To evaluate the flexibility of the function, a train-and-test scheme is carried out. In other words, a partition of the time series is allowed, of which the former one is used to train the parameters, whereas the latter one is applied to test.

The logic of Auto Exponential Smoothing is implemented as follows:

If MODELSELECTION is set to 1:

1. Check if FORECAST_MODEL_NAME is set:
   - If FORECAST_MODEL_NAME is set by user, the model (TESM/DESM/SESM) defined will be used. (For details, please see section 4).
   - Otherwise, test whether there is seasonality in the time series (use SEASONALITY_CRITERION to set the threshold).
2. If there is seasonality, test whether there is a trend in the series:
   - If both trend and seasonality tests are positive, TES will be chosen. and optimize alpha, beta, gamma.
   - If there is seasonality but no trend, TES will be chosen, but BETA and TREND_START will be set to zero. Only alpha and gamma will be optimized.
3. If there is no seasonality, test whether there is a trend in the time series:
   - If there is a trend, DES will be chosen and optimize alpha, beta, and phi.
   - If there is no trend, SES will be chosen and optimize alpha.
4. For all cases above:

- If the model is set or detected as a DESM and DAMPED is not set, the algorithm will try both DAMPED=1 and DAMPED=0, and return the result with smaller accuracy measure. If DAMPED is set to 1, phi will be optimized.
- If the model is set or detected as a TESM:
  - If CYCLE is provided by user, use the user-defined CYCLE value. Otherwise, CYCLE will be calculated automatically.
  - If the SEASONAL parameter is defined by user, use the multiplicative/additive seasonality as user defined. Otherwise, use the seasonality detected by the algorithm.
  - If DAMPED is not set, the algorithm will try both DAMPED=1 and DAMPED=0, and return the result with smaller accuracy measure. If DAMPED is set to 1, phi will be optimized.

If MODELSELECTION is set to 0, you need to specify the model (SESM, DESM, TESM).

1. If TESM is selected, and CYCLE is not provided, CYCLE will be calculated automatically.
2. If any of alpha, beta, gamma and phi is not provided, it will be calculated automatically per the logic when MODELSELECTION is set to 1.

The algorithms we used for optimization are Simulated Annealing and Nelder-Mead. Simulated Annealing is a metaheuristic algorithm, it can find global minimum for most of times. For each iteration, the algorithm randomly selects a solution close the current one, measure its quality, then decides to accept it or reject it. During the search, the temperature is progressively decreased from an initial positive value to zero and the possibility of rejecting a new solution increased. Once Simulated Annealing is stopped due to its stop condition, Nelder-Mead is called to refine the result because Nelder-Mead is a heuristic search method that can converge to non-stationary points. Different to Simulated Annealing, there is no random component in Nelder-Mead.

Because of the randomness in Simulated Annealing, it is possible the results are different between two runs. But we can use OPTIMIZER_RANDOM_SEED to set the random seed.

Stop conditions:

Simulated Annealing:

- The maximal iteration is reached.
- The temperature is lower than 0.01(It is hard coded in current version).
- The new solution is rejected for certain (it is a parameter exposed to user) times consecutively.

Nelder-Mead:

- Execution time is greater than time budget.
- Difference between the worst and the best error is less than 0.0001 (hard coded).

## Prerequisites

- No missing or null data in the inputs.
- The data is numeric, not categorical.

## _SYS_AFL.PAL_AUTO_EXPSMOOTH

This function is used to calculate optimal parameters and output forecast results.

### Overview of all Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<FORECAST table>` | OUT |
| 4 | `<STATISTICS table>` | OUT |

### Input Table

| Table | Column | Data Type | Description |
|---|---|---|---|
| DATA | 1st column | INTEGER | Index of timestamp |
| | 2nd column | INTEGER, DOUBLE, or DECIMAL(p, s) | Raw data |

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| MODELSELECTION | INTEGER | 0 | When this is set to 1, the algorithm will select the best model among Single/Double/Triple/Damped Double/Damped Triple Exponential Smoothing models.<br><br>If FORECAST_MODEL_NAME is set, the model defined by FORECAST_MODEL_NAME will be used.<br><br>When this is set to 0, the algorithm will not perform the model selection. | |
| FORECAST_MODEL_NAME | VARCHAR | No default value | Name of the statistical model used for calculating the forecast.<br>• SESM: Single Exponential Smoothing<br>• DESM: Double Exponential Smoothing<br>• TESM: Triple Exponential Smoothing | This parameter must be set unless MODELSELECTION is set to 1. |
| OPTIMIZER_TIME_BUDGET | INTEGER | 1 | Time budget for Nelder-Mead optimization process. The time unit is second and the value should be larger than zero. | |
| MAX_ITERATION | INTEGER | 100 | Maximum number of iterations for simulated annealing. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| OPTIMIZER_RANDOM_SEED | INTEGER | System time | Random seed for simulated annealing. The value should be larger than zero. | |
| THREAD_RATIO | DOUBLE | 1.0 | Specifies the ratio of total number of threads that can be used by this function. The value range is from 0 to 1, where 0 means only using one thread, and 1 means using at most all the currently available threads. Values outside the range are ignored and this function heuristically determines the number of threads to use. | |
| EXPOST_FLAG | INTEGER | 1 | • 0: Does not output the expost forecast, and just outputs the forecast values.<br>• 1: Outputs the expost forecast and the forecast values. | |
| ALPHA | DOUBLE | Computed automatically | Weight for smoothing.<br>Value range: $0 < \alpha < 1$ | |
| BETA | DOUBLE | Computed automatically | Weight for the trend component.<br>Value range: 0 Value range: $0 \leq \beta < 1$<br>**Note**: If it is not set, the optimized value will be computed automatically. | Only valid when the model is set by user or identified by the algorithm as a DESM/TESM.<br>Value 0 is allowed under TESM model only. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| GAMMA | DOUBLE | Computed automatically | Weight for the seasonal component.<br>Value range: $0 < \gamma < 1$ | Only valid when the model is set by user or identified by the algorithm as TESM. |
| PHI | DOUBLE | Computed automatically | Value of the damped smoothing constant $\Phi$ ($0 < \Phi < 1$). | Only valid when the model is set by user or identified by the algorithm as a DAMPED model. |
| FORECAST_NUM | INTEGER | 0 | Number of values to be forecast. | |
| CYCLE | INTEGER | Computed automatically | Length of a cycle ($L > 1$). For example, the cycle of quarterly data is 4, and the cycle of monthly data is 12. | Only valid when the model is set by user or identified by the algorithm as a TESM. |
| SEASONAL | INTEGER | If MODELSELECTION is set to 1, the default value will be computed automatically. Otherwise, the default value is 0. | • 0: Multiplicative triple exponential smoothing<br>• 1: Additive triple exponential smoothing | Only valid when the model is set by user or identified by the algorithm as a TESM. |
| INITIAL_METHOD | INTEGER | 0 or 1 | Initialization method for the trend and seasonal components. Refer to *Triple Exponential Smoothing* for detailed information on initialization method. | Only valid when the model is set by user or identified by the algorithm as a TESM. |
| TRAINING_RATIO | DOUBLE | 1.0 | The ratio of training data to the whole time series.<br>Assuming the size of time series is N, and the training ratio is r, the first N*r time series is used to train, whereas only the latter N*(1-r) one is used to test. | If this parameter is set to 0.0 or 1.0, or the resulting training data (N*r) is less than 1 or equal to the size of time series, no train-and-test procedure is carried out. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| DAMPED | INTEGER | If MODELSELECTION is set to 1, the default value will be computed automatically. Otherwise, the default value is 0. | For DESM:<br>• 0: Uses the Holt's linear method.<br>• 1: Uses the additive damped trend Holt's linear method.<br>For TESM:<br>• 0: Uses the Holt Winter method.<br>• 1: Uses the additive damped seasonal Holt Winter method. | |
| ACCURACY_MEASURE | VARCHAR | MSE | The criterion used for the optimization. Available values are MSE and MAPE. | |
| SEASONALITY_CRITERION | DOUBLE | 0.5 | The criterion of the auto-correlation coefficient for accepting seasonality, in the range of (0, 1). The larger it is, the less probable a time series is regarded to be seasonal. | Only valid when FORECAST_MODEL_NAME is TESM or MODELSELECTION is set to 1, and CYCLE is not defined. |
| TREND_TEST_METHOD | INTEGER | 1 | The method to test trend:<br>• 1: MK test<br>• 2: Difference-sign test | Only valid when MODELSELECTION is set to 1. |
| TREND_TEST_ALPHA | DOUBLE | 0.05 | Tolerance probability for trend test. The value range is (0, 0.5). | Only valid when MODELSELECTION is set to 1. |
| ALPHA_MIN | DOUBLE | 0.0000000001 | Sets the minimum value of ALPHA. | Only valid when ALPHA is not defined. |
| BETA_MIN | DOUBLE | 0.0000000001 | Sets the minimum value of BETA. | Only valid when BETA is not defined. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| GAMMA_MIN | DOUBLE | 0.0000000001 | Sets the minimum value of GAMMA. | Only valid when GAMMA is not defined. |
| PHI_MIN | DOUBLE | 0.0000000001 | Sets the minimum value of PHI. | Only valid when PHI is not defined. |
| ALPHA_MAX | DOUBLE | 1.0 | Sets the maximum value of ALPHA. | Only valid when ALPHA is not defined. |
| BETA_MAX | DOUBLE | 1.0 | Sets the maximum value of BETA. | Only valid when BETA is not defined. |
| GAMMA_MAX | DOUBLE | 1.0 | Sets the maximum value of GAMMA. | Only valid when GAMMA is not defined. |
| PHI_MAX | DOUBLE | 1.0 | Sets the maximum value of PHI. | Only valid when PHI is not defined. |
| PREDICTION_CONFIDENCE_1 | DOUBLE | 0.8 | Prediction confidence for interval 1 | Only valid when the upper and lower columns are provided in the result table. |
| PREDICTION_CONFIDENCE_2 | DOUBLE | 0.95 | Prediction confidence for interval 2 | Only valid when the upper and lower columns are provided in the result table. |
| LEVEL_START | DOUBLE | No default value | The initial value for level component S. If this value is not provided, it will be calculated in the way as described in Triple Exponential Smoothing.<br>**Note**: LEVEL_START cannot be zero. If it is set to zero, 0.0000000001 will be used instead.<br>Refer to *Triple Exponential Smoothing* for detailed information | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| TREND_START | DOUBLE | No default value | The initial value for trend component B. If this value is not provided, it will be calculated in the way as described in Triple Exponential Smoothing.<br><br>Refer to *Triple Exponential Smoothing* for detailed information | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| SEASON_START | INTEGER or DOUBLE | No default value | A list of initial values for seasonal component **C**. If this parameter is not used, the initial values of **C** will be calculated in the way as described in Triple Exponential Smoothing.<br><br>Two values must be provided for each cycle:<br>• Cycle ID: An INTEGER which represents which cycle the initial value is used for.<br>• Initial value: A DOUBLE precision number which represents the initial value for the corresponding cycle.<br>For example: To give the initial value 0.5 to the 3rd cycle, insert tuple (`'SEASON_START', 3, 0.5, NULL`) into the parameter table.<br><br>**Note**: The initial values of all cycles must be provided if this parameter is used.<br><br>Refer to *Triple Exponential Smoothing* for detailed information | |

> **Note**
>
> Cycle determines the seasonality within the time series data by considering the seasonal factor of a data point$_{t\text{-}CYCLE+1}$ in the forecast calculation of data point$_{t+1}$. Additionally, the algorithm of TESM takes an entire

CYCLE as the base to calculate the first forecast value for data point$_{CYCLE+1}$. The value range for CYCLE should be 2 ≤ CYCLE ≤ total number of data points/2.

For example, there is one year of weekly data (52 data points) as input time series. The value for CYCLE should be within the range of 2 <= CYCLE <= 26. If CYCLE is 4 then we get the first forecast value for data point 5 (for example, week 201205) by considering the seasonal factor of data point 1 (for example, week 201201). The second forecast value for data point 6 (for example, week 201206) takes into account of the seasonal factor of data point 2 (for example, week 201202). If CYCLE is 2 then we get the first forecast value for data point 3 (for example, week 201203) by considering the seasonal factor of data point 1 (for example, week 201201). The second forecast value for data point 4 (for example, week 201204) takes into account of the seasonal factor of data point 2 (for example, week 201202).

**Output Tables**

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| STATISTICS | 1st column | NVARCHAR(100) | STAT_NAME | Name of related statistics.<br>• MSE: the statistics of MSE (Mean Squared Error)<br>• NUMBER_OF_ITERATIONS: the sum of numbers of iterations of both optimizers (simulated annealing and Nelder Mead)<br>• ALPHA/BETA/GAMMA: the optimal parameters or values set by user<br>• (For train-and-test scheme only) NUMBER_OF_TRAINING: the size of time series used to train<br>• (For train-and-test scheme only) NUMBER_OF_TESTING: the size of time series used to test<br>• (For train-and-test scheme only) TEST_MSE: the statistics of MSE calculated from the test data |
| | 2nd column | NVARCHAR(100) | STAT_VALUE | Parameter values |
| FORECAST | 1st column | INTEGER | TIMESTAMP | Index of time stamp |
| | 2nd column | DOUBLE | VALUE | Result of smoothing forecast |
| | 3rd column | DOUBLE | PI1_LOWER | Lower bound of prediction interval 1 |
| | 4th column | DOUBLE | PI1_UPPER | Upper bound of prediction interval 1 |

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| | 5th column | DOUBLE | PI2_LOWER | Lower bound of prediction interval 2 |
| | 6th column | DOUBLE | PI2_UPPER | Upper bound of prediction interval 2 |

### Examples

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

For Triple Exponential Smoothing

```
SET SCHEMA DM_PAL;
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME " VARCHAR(100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (100)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FORECAST_MODEL_NAME', NULL, NULL,'TESM');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('ALPHA', NULL,0.4, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('BETA', NULL,0.4, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('GAMMA', NULL,0.4, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('CYCLE',4, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FORECAST_NUM',3, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SEASONAL',0, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('INITIAL_METHOD',1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('TRAINING_RATIO',NULL, 0.75, NULL);
DROP TABLE PAL_FORECASTTRIPLESMOOTHING_DATA_TBL;
CREATE COLUMN TABLE PAL_FORECASTTRIPLESMOOTHING_DATA_TBL ("TIMESTAMP" INT,
"VALUE" DOUBLE);
INSERT INTO PAL_FORECASTTRIPLESMOOTHING_DATA_TBL VALUES (1,362.0);
INSERT INTO PAL_FORECASTTRIPLESMOOTHING_DATA_TBL VALUES (2,385.0);
INSERT INTO PAL_FORECASTTRIPLESMOOTHING_DATA_TBL VALUES (3,432.0);
INSERT INTO PAL_FORECASTTRIPLESMOOTHING_DATA_TBL VALUES (4,341.0);
INSERT INTO PAL_FORECASTTRIPLESMOOTHING_DATA_TBL VALUES (5,382.0);
INSERT INTO PAL_FORECASTTRIPLESMOOTHING_DATA_TBL VALUES (6,409.0);
INSERT INTO PAL_FORECASTTRIPLESMOOTHING_DATA_TBL VALUES (7,498.0);
INSERT INTO PAL_FORECASTTRIPLESMOOTHING_DATA_TBL VALUES (8,387.0);
INSERT INTO PAL_FORECASTTRIPLESMOOTHING_DATA_TBL VALUES (9,473.0);
INSERT INTO PAL_FORECASTTRIPLESMOOTHING_DATA_TBL VALUES (10,513.0);
INSERT INTO PAL_FORECASTTRIPLESMOOTHING_DATA_TBL VALUES (11,582.0);
INSERT INTO PAL_FORECASTTRIPLESMOOTHING_DATA_TBL VALUES (12,474.0);
INSERT INTO PAL_FORECASTTRIPLESMOOTHING_DATA_TBL VALUES (13,544.0);
INSERT INTO PAL_FORECASTTRIPLESMOOTHING_DATA_TBL VALUES (14,582.0);
INSERT INTO PAL_FORECASTTRIPLESMOOTHING_DATA_TBL VALUES (15,681.0);
INSERT INTO PAL_FORECASTTRIPLESMOOTHING_DATA_TBL VALUES (16,557.0);
INSERT INTO PAL_FORECASTTRIPLESMOOTHING_DATA_TBL VALUES (17,628.0);
INSERT INTO PAL_FORECASTTRIPLESMOOTHING_DATA_TBL VALUES (18,707.0);
INSERT INTO PAL_FORECASTTRIPLESMOOTHING_DATA_TBL VALUES (19,773.0);
INSERT INTO PAL_FORECASTTRIPLESMOOTHING_DATA_TBL VALUES (20,592.0);
INSERT INTO PAL_FORECASTTRIPLESMOOTHING_DATA_TBL VALUES (21,627.0);
INSERT INTO PAL_FORECASTTRIPLESMOOTHING_DATA_TBL VALUES (22,725.0);
INSERT INTO PAL_FORECASTTRIPLESMOOTHING_DATA_TBL VALUES (23,854.0);
INSERT INTO PAL_FORECASTTRIPLESMOOTHING_DATA_TBL VALUES (24,661.0);
```

```
CALL _SYS_AFL.PAL_AUTO_EXPSMOOTH(PAL_FORECASTTRIPLESMOOTHING_DATA_TBL,
#PAL_PARAMETER_TBL, ?,?);
```

### Expected Results

FORECAST:

| | TIMESTAMP | VALUE | PI1_LOWER | PI1_UPPER | PI2_LOWER | PI2_UPPER |
|---|---|---|---|---|---|---|
| 1 | 1 | 320.0185017610371 | ? | ? | ? | ? |
| 2 | 2 | 374.22511287989374 | ? | ? | ? | ? |
| 3 | 3 | 458.64978231597735 | ? | ? | ? | ? |
| 4 | 4 | 364.3760780620642 | ? | ? | ? | ? |
| 5 | 5 | 416.00900775181265 | ? | ? | ? | ? |
| 6 | 6 | 430.0187143564633 | ? | ? | ? | ? |
| 7 | 7 | 487.9822208386645 | ? | ? | ? | ? |
| 8 | 8 | 387.8283483049524 | ? | ? | ? | ? |
| 9 | 9 | 435.8180489270685 | ? | ? | ? | ? |
| 10 | 10 | 487.8385306263397 | ? | ? | ? | ? |
| 11 | 11 | 596.9056231535724 | ? | ? | ? | ? |
| 12 | 12 | 468.2732838231433 | ? | ? | ? | ? |
| 13 | 13 | 546.4964927740829 | ? | ? | ? | ? |
| 14 | 14 | 585.2154739008896 | ? | ? | ? | ? |
| 15 | 15 | 681.2352509973642 | ? | ? | ? | ? |
| 16 | 16 | 541.8975758566653 | ? | ? | ? | ? |
| 17 | 17 | 632.2501376517993 | ? | ? | ? | ? |
| 18 | 18 | 674.55582842255 | ? | ? | ? | ? |
| 19 | 19 | 807.148603754242 | ? | ? | ? | ? |
| 20 | 20 | 649.5103230097602 | ? | ? | ? | ? |
| 21 | 21 | 743.6885972977457 | ? | ? | ? | ? |
| 22 | 22 | 807.0413100633278 | ? | ? | ? | ? |
| 23 | 23 | 931.8296547304078 | ? | ? | ? | ? |
| 24 | 24 | 746.1103554425168 | ? | ? | ? | ? |
| 25 | 25 | 850.3303858970734 | 784.73163... | 915.9291... | 750.00574... | 950.6550... |
| 26 | 26 | 918.7625218945378 | 839.71610... | 997.8089... | 797.87146... | 1,039.65... |
| 27 | 27 | 1,056.51070570657... | 953.61825... | 1,159.40... | 899.15028... | 1,213.87... |

STATISTICS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | MSE | 467.811415778471 |
| 2 | NUMBER_OF_ITERATIONS | 110 |
| 3 | SA_NUMBER_OF_ITERAT... | 100 |
| 4 | NM_NUMBER_OF_ITERA... | 10 |
| 5 | NM_EXECUTION_TIME | 0.000154 |
| 6 | SA_STOP_COND | MAX_ITERATION |
| 7 | NM_STOP_COND | ERROR_DIFFERENCE |
| 8 | ALPHA | 0.4 |
| 9 | BETA | 0.4 |
| 10 | GAMMA | 0.4 |
| 11 | CYCLE | 4 |
| 12 | NUMBER_OF_TRAINING | 18 |
| 13 | NUMBER_OF_TESTING | 6 |
| 14 | TEST_MSE | 6353.632907713298 |

For Model Selection:

```
SET SCHEMA DM_PAL;
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME " VARCHAR(100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (100)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FORECAST_NUM',3, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MODELSELECTION',1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('CYCLE',12, NULL, NULL);
DROP TABLE PAL_FORECASTSMOOTHING_DATA_TBL;
CREATE COLUMN TABLE PAL_FORECASTSMOOTHING_DATA_TBL ("TIMESTAMP" INT, "VALUE"
DOUBLE);
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    1    ,    0    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    2    ,    23    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    3    ,    21    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    4    ,    2    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    5    ,    20    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    6    ,    0    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    7    ,    12    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    8    ,    2    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    9    ,    25    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    10    ,    4    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    11    ,    30    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    12    ,    2    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    13    ,    375    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    14    ,    200    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    15    ,    200    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    16    ,    375    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    17    ,    350    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    18    ,    250    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    19    ,    275    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    20    ,    250    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    21    ,    125    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    22    ,    25    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    23    ,    100    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    24    ,    325    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    25    ,    175    );
```

```
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    26    ,    100    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    27    ,    150    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    28    ,    300    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    29    ,    200    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    30    ,    250    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    31    ,    225    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    32    ,    250    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    33    ,    300    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    34    ,    150    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    35    ,    300    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    36    ,    200    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    37    ,    375    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    38    ,    200    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    39    ,    200    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    40    ,    375    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    41    ,    350    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    42    ,    250    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    43    ,    275    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    44    ,    250    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    45    ,    125    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    46    ,    25    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    47    ,    100    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    48    ,    325    );
CALL _SYS_AFL.PAL_AUTO_EXPSMOOTH(PAL_FORECASTSMOOTHING_DATA_TBL,
#PAL_PARAMETER_TBL, ?,?);
```

### Expected Results

FORECAST:

| | TIMESTAMP | VALUE | PI1_LOWER | PI1_UPPER | PI2_LOWER | PI2_UPPER |
|---|---|---|---|---|---|---|
| 14 | 15 | 189.25804182288994 | ? | ? | ? | ? |
| 15 | 16 | 200.5177149881617 | ? | ? | ? | ? |
| 16 | 17 | 270.60884059106786 | ? | ? | ? | ? |
| 17 | 18 | 305.7222288694876 | ? | ? | ? | ? |
| 18 | 19 | 291.38287026577905 | ? | ? | ? | ? |
| 19 | 20 | 290.87863463613115 | ? | ? | ? | ? |
| 20 | 21 | 281.095172716014 | ? | ? | ? | ? |
| 21 | 22 | 229.167494137989 | ? | ? | ? | ? |
| 22 | 23 | 159.4680542986415 | ? | ? | ? | ? |
| 23 | 24 | 141.91807656664008 | ? | ? | ? | ? |
| 24 | 25 | 212.01989214673995 | ? | ? | ? | ? |
| 25 | 26 | 202.0440749831026 | ? | ? | ? | ? |
| 26 | 27 | 168.23941651974854 | ? | ? | ? | ? |
| 27 | 28 | 164.5875333549908 | ? | ? | ? | ? |
| 28 | 29 | 216.43109949961521 | ? | ? | ? | ? |
| 29 | 30 | 213.0176973673904 | ? | ? | ? | ? |
| 30 | 31 | 228.78521631322502 | ? | ? | ? | ? |
| 31 | 32 | 229.5986107935081 | ? | ? | ? | ? |
| 32 | 33 | 239.02191466546748 | ? | ? | ? | ? |
| 33 | 34 | 263.0096216628069 | ? | ? | ? | ? |
| 34 | 35 | 223.77460110562765 | ? | ? | ? | ? |
| 35 | 36 | 253.0262154871063 | ? | ? | ? | ? |
| 36 | 37 | 235.2951054310857 | ? | ? | ? | ? |
| 37 | 38 | 287.3361888007843 | ? | ? | ? | ? |
| 38 | 39 | 256.9541121122527 | ? | ? | ? | ? |
| 39 | 40 | 237.49421097124912 | ? | ? | ? | ? |
| 40 | 41 | 288.45560630619076 | ? | ? | ? | ? |
| 41 | 42 | 311.79379900891547 | ? | ? | ? | ? |
| 42 | 43 | 290.3365447978979 | ? | ? | ? | ? |
| 43 | 44 | 285.6546933671645 | ? | ? | ? | ? |
| 44 | 45 | 273.5415147017091 | ? | ? | ? | ? |
| 45 | 46 | 220.43764032057587 | ? | ? | ? | ? |
| 46 | 47 | 150.27469967446598 | ? | ? | ? | ? |
| 47 | 48 | 132.6945778104217 | ? | ? | ? | ? |
| 48 | 49 | 203.0230814861557 | 69.604533... | 336.4416... | -1.022979... | 407.0691... |
| 49 | 50 | 203.58044880288907 | 61.662473... | 345.4984... | -13.46436... | 420.6252... |
| 50 | 51 | 204.09706777335848 | 54.160699... | 354.0334... | -25.21081... | 433.4049... |

STATISTICS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | FORECAST_MODEL_NAME | DESM |
| 2 | MSE | 10400.539085812015 |
| 3 | NUMBER_OF_ITERATIONS | 311 |
| 4 | SA_NUMBER_OF_ITERATI... | 100 |
| 5 | NM_NUMBER_OF_ITERA... | 211 |
| 6 | NM_EXECUTION_TIME | 0.002152 |
| 7 | SA_STOP_COND | MAX_ITERATION |
| 8 | NM_STOP_COND | ERROR_DIFFEREN... |
| 9 | ALPHA | 0.3625855849047114 |
| 10 | BETA | 0.00000000108809... |
| 11 | PHI | 0.9268913963187103 |
| 12 | DAMPED | true |

For Model Selection with defined FORECAST_MODEL_NAME:

```
SET SCHEMA DM_PAL;
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME " VARCHAR(100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (100)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FORECAST_MODEL_NAME',NULL, NULL, 'TESM');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FORECAST_NUM',3, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MODELSELECTION',1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('CYCLE',12, NULL, NULL);
DROP TABLE PAL_FORECASTSMOOTHING_DATA_TBL;
CREATE COLUMN TABLE PAL_FORECASTSMOOTHING_DATA_TBL ("TIMESTAMP" INT, "VALUE"
DOUBLE);
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    1    ,    0    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    2    ,    23    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    3    ,    21    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    4    ,    2    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    5    ,    20    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    6    ,    0    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    7    ,    12    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    8    ,    2    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    9    ,    25    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    10    ,    4    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    11    ,    30    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    12    ,    2    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    13    ,    375    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    14    ,    200    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    15    ,    200    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    16    ,    375    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    17    ,    350    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    18    ,    250    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    19    ,    275    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    20    ,    250    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    21    ,    125    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    22    ,    25    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    23    ,    100    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    24    ,    325    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    25    ,    175    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    26    ,    100    );
```

```
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    27    ,    150    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    28    ,    300    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    29    ,    200    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    30    ,    250    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    31    ,    225    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    32    ,    250    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    33    ,    300    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    34    ,    150    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    35    ,    300    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    36    ,    200    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    37    ,    375    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    38    ,    200    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    39    ,    200    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    40    ,    375    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    41    ,    350    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    42    ,    250    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    43    ,    275    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    44    ,    250    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    45    ,    125    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    46    ,    25    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    47    ,    100    );
INSERT INTO PAL_FORECASTSMOOTHING_DATA_TBL VALUES (    48    ,    325    );
CALL _SYS_AFL.PAL_AUTO_EXPSMOOTH(PAL_FORECASTSMOOTHING_DATA_TBL,
#PAL_PARAMETER_TBL, ?,?);
```

Expected Results

FORECAST:

| | TIMESTAMP | VALUE | PI1_LOWER | PI1_UPPER | PI2_LOWER | PI2_UPPER |
|---|---|---|---|---|---|---|
| 15 | 15 | 174.373445344966... | ? | ? | ? | ? |
| 16 | 16 | 369.306074073794... | ? | ? | ? | ? |
| 17 | 17 | 334.5899124754728 | ? | ? | ? | ? |
| 18 | 18 | 294.433382182060... | ? | ? | ? | ? |
| 19 | 19 | 274.9780597926273 | ? | ? | ? | ? |
| 20 | 20 | 259.0756972013552 | ? | ? | ? | ? |
| 21 | 21 | 233.284207238669... | ? | ? | ? | ? |
| 22 | 22 | 91.37998560012652 | ? | ? | ? | ? |
| 23 | 23 | 128.5261528401553 | ? | ? | ? | ? |
| 24 | 24 | 129.159794188415... | ? | ? | ? | ? |
| 25 | 25 | 322.6962426046035 | ? | ? | ? | ? |
| 26 | 26 | 126.830605448393... | ? | ? | ? | ? |
| 27 | 27 | 121.6493220460936 | ? | ? | ? | ? |
| 28 | 28 | 291.6608795836966 | ? | ? | ? | ? |
| 29 | 29 | 242.5022936363649 | ? | ? | ? | ? |
| 30 | 30 | 170.132661169157... | ? | ? | ? | ? |
| 31 | 31 | 189.398071208799... | ? | ? | ? | ? |
| 32 | 32 | 192.437214546754... | ? | ? | ? | ? |
| 33 | 33 | 199.316111346482... | ? | ? | ? | ? |
| 34 | 34 | 150.4317371158296 | ? | ? | ? | ? |
| 35 | 35 | 238.770048956019... | ? | ? | ? | ? |
| 36 | 36 | 296.245934368595... | ? | ? | ? | ? |
| 37 | 37 | 394.341818920740... | ? | ? | ? | ? |
| 38 | 38 | 235.7165696235234 | ? | ? | ? | ? |
| 39 | 39 | 230.256284855173... | ? | ? | ? | ? |
| 40 | 40 | 378.084862662457... | ? | ? | ? | ? |
| 41 | 41 | 319.7506086426555 | ? | ? | ? | ? |
| 42 | 42 | 272.526801232336... | ? | ? | ? | ? |
| 43 | 43 | 255.291179303462... | ? | ? | ? | ? |
| 44 | 44 | 245.2846622211183 | ? | ? | ? | ? |
| 45 | 45 | 225.5990973332117 | ? | ? | ? | ? |
| 46 | 46 | 88.3811580726637 | ? | ? | ? | ? |
| 47 | 47 | 128.4044947355799 | ? | ? | ? | ? |
| 48 | 48 | 130.377877951570... | ? | ? | ? | ? |
| 49 | 49 | 324.2218306519815 | 222.34396... | 426.0996... | 168.41308... | 480.0305... |
| 50 | 50 | 187.567738925899... | 77.902306... | 297.2331... | 19.848932... | 355.2865... |
| 51 | 51 | 206.732240177103... | 86.266753... | 327.1977... | 22.496176... | 390.9683... |

STATISTICS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | FORECAST_MODEL_NAME | TESM |
| 2 | MSE | 6529.128113502934 |
| 3 | NUMBER_OF_ITERATIONS | 493 |
| 4 | SA_NUMBER_OF_ITERATI... | 100 |
| 5 | NM_NUMBER_OF_ITERA... | 393 |
| 6 | NM_EXECUTION_TIME | 0.00324 |
| 7 | SA_STOP_COND | MAX_ITERATION |
| 8 | NM_STOP_COND | ERROR_DIFFERE... |
| 9 | ALPHA | 0.248168271761591 |
| 10 | BETA | 0.9999608319952... |
| 11 | GAMMA | 0.0000000001 |
| 12 | PHI | 0.605391471032646 |
| 13 | CYCLE | 12 |
| 14 | SEASONAL | Additive |
| 15 | DAMPED | true |

> **Note**
>
> The wrapper procedure generation approach to call PAL functions is still supported. For more information, refer to the older versions (prior to SAP HANA Platform 2.0 SPS 02) of this document.

### Related Information

Triple Exponential Smoothing [page 924]

## 5.5.5 Bayesian Change Point Detection

Bayesian Change Point Detection(BCPD) detects abrupt changes in the time series. It, to some extent, can been assumed as an enhanced version of seasonality test in additive mode. Similarly, it decomposes a time series into three components: trend, season and random, but with a remarkable difference that it is capable of detecting change points within both trend and season parts, using a quasi RJ-MCMC method.

Mathematically, the time series Y (t) can be formulated as below

$Y(t) = T(t,\theta_T) + S(t,\theta_S) + \epsilon(t)$

where $\theta_T$ is the parameters in the trend part, composed of the location of trend change points and the coefficients $\gamma$ of each segments. Specifically, any segment of the trend part can been assumed as

$$\text{trend_segment}(t)= \sum_{i=0}^{\text{trend_order}} Y_i * t^i$$

Likewise, $\theta_S$ is the parameters in the season part, composed of the location of season change points and the coefficients of each segments, i.e. (α, β) and we may have

$$\text{season_segment}(t)=\sum_{l=1}^{\text{harmonic_order}}\left[\alpha_l*\sin\left(\frac{2\pi lt}{\text{period}}\right)+\beta_l*\cos\left(\frac{2\pi lt}{\text{period}}\right)\right]$$

Periods in different season segments can vary from one to another.

The decomposation graphs are shown below, red and yellow dashed lines denotes the position where seasonal and trend change points occur respectively.

## Prerequisites

- No null data in the inputs. The time periods should be unique (no duplication) and equal sampling.
- The length of time series must be at least 1.

## _SYS_AFL.PAL_BCPD

This function detects change points within trend and season parts after the decomposation of target time series.

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<TREND_CHANGE_POINT table>` | OUT |
| 4 | `<SEASON_CHANGE_POINT table>` | OUT |
| 5 | `<PERIOD_LIST table>` | OUT |
| 6 | `<DECOMPOSED table>` | OUT |

### Input Table

| Table | Column | Data Type | Description | Dependency |
|---|---|---|---|---|
| DATA | 1st Column | INTEGER | Time stamp. Time stamp does not need to be in order, but must be unique and equal sampling | |
| | 2nd Column | INTEGER, DOUBLE, or DECIMAL(p,s) | Time series value | |

### Parameter Table

Mandatory Parameters

The following parameters are mandatory and must be given a value.

| NAME | DATA TYPE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|
| MAX_TCP_NUM | INTEGER | Maximum number of trend change points while decomposition | Cannot be larger than the length of data |
| MAX_SCP_NUM | INTEGER | Maximum order of harmonic waves within seasonal segments while Maximum number of season change points while decomposition | Cannot be larger than the length of data |

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

Parameters for Trend Component:

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| TREND_ORDER | INTEGER | 1 | Order of trend segments that used for decomposition | In the range [0, 3], positive if trend component needed after decomposition |

Parameters for Seasonal Component:

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| MAX_HARMONIC_ORDER | INTEGER | 0 | Maximum order of harmonic waves within seasonal segments while decomposition | Be positive if seasonal component needed after decomposition |
| MIN_PERIOD | INTEGER | 1 | Minimum possible period within seasonal segments while decomposition | Cannot be larger than MAX_PERIOD |
| MAX_PERIOD | INTEGER | Half of the data length | Maximum possible period within seasonal segments while decomposition | Cannot be larger than the length of data |

Other Optional Parameters:

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| RANDOM_SEED | INTEGER | 0 | 0 means using machine time as seed | |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| MAX_ITER | INTEGER | 5000 | BCPD is iterative, the more iterations, the more precise will the result be rendered. | |
| INTERVAL_RATIO | DOUBLE | 0.1 | Regulates the interval between change points, which should be larger than the corresponding portion of total length (say INTERVAL_RATIO * TOTAL_LENGTH) | Ranging from 0 to 1 |

### Output Table

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| TREND_CHANGE_POINT | 1st Column | INTEGER | ROW_INDEX | Indicates the id of current row. |
| | 2nd Column | INTEGER | TREND_CP | Indicates the position of change points in the trend part. |

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| SEASON_CHANGE_POINT | 1st Column | INTEGER | ROW_INDEX | Indicates the id of current row. |
| | 2nd Column | INTEGER | SEASON_CP | Indicates the position of change points in the season part. |

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| PERIOD_LIST | 1st Column | INTEGER | ROW_INDEX | Indicates the id of current row. |
| | 2nd Column | INTEGER | PERIOD | Indicates the period within each season segment. |

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| DECOMPOSED | 1st Column | INTEGER | ID | Time stamp that is monotonically increasing sorted. |
| | 2nd Column | DOUBLE | SEASONAL | The seasonal component. |
| | 3rd Column | DOUBLE | TREND | The trend component.. |
| | 4th Column | DOUBLE | RANDOM | The random component.. |

### Examples

Assume that:

- DM_PAL is a schema belonging to USER1;
- USER1 has been assigned the AFL__SYS_AFL_AFLPAL_EXECUTE or AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION role.

### Example

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_BCPD_DATA_TBL;
CREATE COLUMN TABLE PAL_BCPD_DATA_TBL (
    "TIME_STAMP" INTEGER,
    "SERIES" DOUBLE
);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (0, 0.0);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (1, 3.83);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (2, -5.33);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (3, -1.5);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (4, 2.33);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (5, -6.83);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (6, -3.0);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (7, 0.83);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (8, -8.33);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (9, -4.5);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (10, -0.67);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (11, -9.83);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (12, -6.0);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (13, -2.17);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (14, -11.33);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (15, -7.5);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (16, -3.67);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (17, -12.83);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (18, -9.0);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (19, -5.17);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (20, -14.33);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (21, -10.5);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (22, -6.67);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (23, -15.83);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (24, -15.83);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (25, -11.075);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (26, -12.891);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (27, -18.769);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (28, -20.585);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (29, -15.83);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (30, -11.075);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (31, -12.891);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (32, -18.769);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (33, -20.585);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (34, -15.83);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (35, -11.075);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (36, -12.891);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (37, -18.769);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (38, -20.585);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (39, -15.83);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (40, -11.075);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (41, -12.891);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (42, -18.769);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (43, -20.585);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (44, -15.83);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (45, -11.075);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (46, -12.891);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (47, -18.769);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (48, -20.585);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (49, -15.83);
```

```
INSERT INTO PAL_BCPD_DATA_TBL VALUES (50, -11.075);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (51, -12.891);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (52, -18.769);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (53, -20.585);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (54, 20.585);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (55, 24.525);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (56, 25.52);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (57, 22.845);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (58, 18.536);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (59, 15.861);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (60, 16.856);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (61, 20.795);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (62, 24.735);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (63, 25.73);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (64, 23.055);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (65, 18.746);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (66, 16.071);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (67, 17.066);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (68, 21.005);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (69, 24.945);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (70, 25.94);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (71, 23.265);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (72, 18.956);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (73, 16.281);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (74, 17.276);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (75, 21.215);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (76, 25.155);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (77, 26.15);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (78, 23.475);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (79, 19.166);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (80, 16.491);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (81, 17.486);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (82, 21.425);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (83, 25.365);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (84, 26.36);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (85, 23.685);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (86, 19.376);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (87, 16.701);
INSERT INTO PAL_BCPD_DATA_TBL VALUES (88, 17.696);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR(100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR(100)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('TREND_ORDER', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MAX_TCP_NUM', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MAX_SCP_NUM', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MAX_HARMONIC_ORDER', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MIN_PERIOD', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MAX_PERIOD', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('RANDOM_SEED', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MAX_ITER', 10000, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('INTERVAL_RATIO', NULL, 0.2, NULL);
DROP TABLE PAL_TCP_TBL;
DROP TABLE PAL_SCP_TBL;
DROP TABLE PAL_PERIODLIST_TBL;
DROP TABLE PAL_DECOMP_TBL;
CREATE COLUMN TABLE PAL_TCP_TBL (ROW_INDEX INTEGER, TCP INTEGER);
CREATE COLUMN TABLE PAL_SCP_TBL (ROW_INDEX INTEGER, SCP INTEGER);
CREATE COLUMN TABLE PAL_PERIODLIST_TBL (ROW_INDEX INTEGER, SCP INTEGER);
CREATE COLUMN TABLE PAL_DECOMP_TBL (ROW_INDEX INTEGER, SEASON DOUBLE, TREND
DOUBLE, RANDOM DOUBLE);
DO BEGIN
    lt_data = SELECT * FROM PAL_BCPD_DATA_TBL;
    lt_param = SELECT * FROM PAL_PARAMETER_TBL;
```

```
    CALL _SYS_AFL.PAL_BCPD (:lt_data, :lt_param, lt_tcp, lt_scp, lt_period,
lt_decomp);
    INSERT INTO PAL_TCP_TBL SELECT * FROM :lt_tcp;
    INSERT INTO PAL_SCP_TBL SELECT * FROM :lt_scp;
    INSERT INTO PAL_PERIODLIST_TBL SELECT * FROM :lt_period;
    INSERT INTO PAL_DECOMP_TBL SELECT * FROM :lt_decomp;
END;
SELECT * FROM PAL_TCP_TBL;
SELECT * FROM PAL_SCP_TBL;
SELECT * FROM PAL_PERIODLIST_TBL;
SELECT * FROM PAL_DECOMP_TBL;
```

**Expected Result**

TREND CHANGE POINTS:

| | ROW_INDEX | TCP |
|---|---|---|
| 1 | 1 | 27 |
| 2 | 2 | 54 |

SEASON CHANGE POINTS:

| | ROW_INDEX | SCP |
|---|---|---|
| 1 | 1 | 26 |
| 2 | 2 | 53 |

PERIOD LIST:

| | ROW_INDEX | PERIOD |
|---|---|---|
| 1 | 1 | 3 |
| 2 | 2 | 5 |
| 3 | 3 | 7 |

DECOMPOSED:

| | ROW_INDEX | SEASON | TREND | RANDOM |
|---|---|---|---|---|
| 1 | 0 | -0.21576466579863457 | -0.05456160050763581 | 0.2703262663062704 |
| 2 | 1 | 4.262193472127376 | -0.578784000973146 | 0.14659052884576962 |
| 3 | 2 | -4.0464288063287395 | -1.1030064014386562 | -0.18056479223260... |
| 4 | 3 | -0.21576466579863574 | -1.6272288019041663 | 0.34299346770280204 |
| 5 | 4 | 4.262193472127378 | -2.1514512023696764 | 0.21925773024229844 |
| 6 | 5 | -4.046428806328737 | -2.675673602835187 | -0.10789759083607... |
| 7 | 6 | -0.21576466579863693 | -3.199896003300697 | 0.41566066909933386 |
| 8 | 7 | 4.262193472127378 | -3.724118403766207 | 0.29192493163882904 |
| 9 | 8 | -4.046428806328736 | -4.2483408042317174 | -0.03523038943954... |
| 10 | 9 | -0.2157646657986381 | -4.772563204697227 | 0.4883278704958651 |
| 11 | 10 | 4.262193472127382 | -5.2967856051627376 | 0.3645921330353552 |
| 12 | 11 | -4.046428806328736 | -5.821008005628247 | 0.03743681195698301 |
| 13 | 12 | -0.21576466579863926 | -6.345230406093758 | 0.560995071892397 |
| 14 | 13 | 4.262193472127383 | -6.869452806559268 | 0.4372593344318849 |
| 15 | 14 | -4.046428806328735 | -7.393675207024778 | 0.11010401335351272 |
| 16 | 15 | -0.2157646657986575 | -7.917897607490288 | 0.6336622732889458 |
| 17 | 16 | 4.262193472127383 | -8.442120007955799 | 0.5099265358284155 |
| 18 | 17 | -4.046428806328725 | -8.966342408421308 | 0.18277121475003355 |
| 19 | 18 | -0.21576466579864162 | -9.490564808886818 | 0.7063294746854596 |
| 20 | 19 | 4.262193472127384 | -10.01478720935233 | 0.5825937372249452 |
| 21 | 20 | -4.046428806328724 | -10.539009609817839 | 0.25543841614656326 |
| 22 | 21 | -0.21576466579864279 | -11.063232010283349 | 0.7789966760819913 |
| 23 | 22 | 4.262193472127384 | -11.58745441074886 | 0.6552609386214758 |
| 24 | 23 | -4.046428806328724 | -12.11167681121437 | 0.32810561754309386 |
| 25 | 24 | -0.21576466579864398 | -12.63589921167988 | -2.978336122521477 |
| 26 | 25 | 4.262193472127385 | -13.16012161214539 | -2.177071859981994 |
| 27 | 26 | 2.7796895790968987 | -13.6843440126109 | -1.9863455664859986 |
| 28 | 27 | -2.723986692822333 | -15.577825186803752 | -0.46718812037391... |
| 29 | 28 | -4.463205940163521 | -15.600799285575258 | -0.5209947742612222 |
| 30 | 29 | -0.03442627698915223 | -15.623773384346766 | -0.17180033866408... |
| 31 | 30 | 4.441929330878105 | -15.646747483118272 | 0.12981815224016735 |
| 32 | 31 | 2.7796895790968996 | -15.669721581889778 | -0.00096799720712... |
| 33 | 32 | -2.7239866928223324 | -15.692695680661286 | -0.3523176265163799 |
| 34 | 33 | -4.463205940163521 | -15.715669779432792 | -0.406124280403688 |
| 35 | 34 | -0.034426276989153... | -15.738643878204298 | -0.05692984480654... |
| 36 | 35 | 4.441929330878105 | -15.761617976975804 | 0.2446886460976998 |
| 37 | 36 | 2.7796895790969005 | -15.784592075747312 | 0.11390249665041186 |
| 38 | 37 | -2.723986692822331 | -15.807566174518819 | -0.23744713265884... |
| 39 | 38 | -4.463205940163521 | -15.830540273290325 | -0.2912537865461555 |

## 5.5.6 Bayesian Structural Time Series

Bayesian structural time series (BSTS) model is for time series analysis including forecasting, decomposition and feature selection.

Compared with ARIMA and Additive Model Time Series Analysis, BSTS is superior because it can deal with much larger scale exogenous data and interpret more precisely on forecasting. Besides, BSTS can sort the exogenous data by their importance to the target time series, which provides users with another perspective to dive into the data.

BSTS takes in a structural time series model for the target series with regression component capturing the contributions of several contemporaneous time series data. The system can be written in the equation form as follows:

$$y_t=\mu_t+\tau_t+\beta^T X_t+\varepsilon_t$$

$$\mu_t=\mu_{t-1}+\delta_{t-1}+u_t$$

$$\delta_t=\delta_{t-1}+v_t$$

$$\tau_t=-\sum_{s=1}^{S-1}\tau_{t-s}+w_t$$

where $y_t$ is the current value of target time series, of which the period is S. $\mu_t$, $\delta_t$, $\tau_t$ and $\beta^T X_t$ are the trend, slope of trend, seasonal and regression components respectively. $\varepsilon_t$, $u_t$, $v_t$ and $w_t$ then are independent gaussian random noise with variances $\sigma_\epsilon^2$, $\sigma_u^2$, $\sigma_v^2$ and $\sigma_w^2$.

As can be inferred from its name, BSTS uses Bayesian techniques inside. It, to be more specific, can be seen as a combination of three Bayesian methods altogether, i.e. kalman filter, spike-and-slab regression and Bayesian model averaging.

Kalman filter is employed to structure the target time series which extracts the inner trend and seasonal (if exists) term.

Spike-and-slab regression is a Bayesian variable selection technique that deals with exogenous contemporaneous time series data. It computes the regression coefficient (beta in the above equations) and simultaneously reveals the "importance" of each exogenous contemporaneous series to the target series. The "spike" is the probability of a particular regression coefficient to be zero and the "slab" is the prior distribution for the coefficient values. Different samples from the "spike" may include different exogenous series, hence we can safely assume that the more important an exogenous series is to the target series, the more likely it will be included.

Bayesian model average is used at length to convert samples from repeated Markov China Monte Carlo (MCMC) draws into the final model outputs.

In a single BSTS draw, the algorithm runs through the following three steps:

1. Sample $\mu_t$, $\delta_t$, $\tau_t$ from kalman filter.
2. Sample $\sigma_u^2$, $\sigma_v^2$ and $\sigma_w^2$ from inverse gamma distributions.
3. Sample $\beta$ and $\sigma_\epsilon^2$ by stochastic search variable selection (SSVS) algorithm which is applied to spike-and-slab regression.

## prerequisites

- No null data in the inputs. The time periods should be unique (no duplication) and equal sampling.
- The length of time series must be at least 1.

## _SYS_AFL.PAL_BSTS_TRAIN

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | <DATA table> | IN |
| 2 | <PARAMETER table> | IN |
| 3 | <STATS table> | OUT |
| 4 | <DECOMPOSE table> | OUT |
| 5 | <MODEL table> | OUT |

### Input Table

| Table | Column | Data Type | Description |
|---|---|---|---|
| DATA | 1st column | INTEGER | Time stamp. Time stamp does not need to be in order, but must be unique and equal sampling. |
| | 2nd column | INTEGER, DOUBLE, or DECIMAL(p,s) | Target time series value. |
| | Other columns | INTEGER, DOUBLE, or DECIMAL(p,s) | Exogenous contemporaneous time series value. |

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| BURN_IN | DOUBLE | 0.5 | Indicates the ratio of total MCMC draws that be ignored at the beginning. Ranging from 0 to 1. | |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| EX-PECTED_MODEL_SIZE | INTEGER | Half of the number of contemporaneous data columns. | Indicates the number of contemporaneous data that expected to be included in the model. | Valid only when exogenous data exist. |
| SEASONAL_PERIOD | INTEGER | -1 | Indicates the Value of the seasonal period.<br>• Negative value: Period determined automatically<br>• 0 and 1: Target time series is assumed non-seasonal<br>• 2 and larger: Period of target time series is set as the corresponding value | |
| MAX_ITER | INTEGER | 1000 | Indicates the total number of MCMC draws. | |
| RANDOM_SEED | INTEGER | 0 | 0 indicates using machine time as seed. | |

### Output Table

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| STATS | 1st Column | VARCHAR or NVARCHAR | DATA_NAME | Indicates the name of contemporaneous time series data |
| | 2nd Column | DOUBLE | INCLUSION_PROB | Indicates the inclusion probability of each contemporaneous time series data, the more important an exogenous series to the target series is, the more likely it will be included |
| | 3rd Column | DOUBLE | AVG_COEFF | Indicates the average value of coefficients when included for each contemporaneous time series data |

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| DECOMPOSE | 1st Column | INTEGER | TIME_STAMP | Time stamp that is monotonically increasing sorted. |
| | 2nd Column | DOUBLE | TREND | The trend component. |
| | 3rd Column | DOUBLE | SEASONAL | The seasonal component. |
| | 4th Column | DOUBLE | REGRESSION | The regression component. |
| | 5th Column | DOUBLE | RANDOM | The random component. |

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| MODEL | 1st Column | INTEGER | ROW_INDEX | Indicates the id of current row. |
| | 2nd Column | NVARCHAR | MODEL_CONTENT | JSON string. |

### Examples

Assume that:

- DM_PAL is a schema belonging to USER1;
- USER1 has been assigned the AFL__SYS_AFL_AFLPAL_EXECUTE or AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION role.

### Example

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_BSTS_DATA_TBL;
CREATE COLUMN TABLE PAL_BSTS_DATA_TBL (
    "TIME_STAMP" INTEGER,
        "TARGET_SERIES" DOUBLE,
        "FEATURE_01" DOUBLE,
    "FEATURE_02" DOUBLE,
    "FEATURE_03" DOUBLE,
    "FEATURE_04" DOUBLE,
    "FEATURE_05" DOUBLE,
    "FEATURE_06" DOUBLE,
    "FEATURE_07" DOUBLE,
    "FEATURE_08" DOUBLE,
    "FEATURE_09" DOUBLE,
    "FEATURE_10" DOUBLE
);
INSERT INTO PAL_BSTS_DATA_TBL VALUES
(0,2.536,1.488,-0.561,1.773,0.909,2.021,1.64,0.3,1.75,0.498,0.073);
INSERT INTO PAL_BSTS_DATA_TBL VALUES
(1,0.882,1.1,-0.992,0.9,0.148,1.28,1.014,0.18,-0.011,0.264,0.584);
INSERT INTO PAL_BSTS_DATA_TBL VALUES
(2,-0.077,1.155,-1.212,1.477,0.21,1.08,1.009,0.119,-0.028,0.031,0.448);
INSERT INTO PAL_BSTS_DATA_TBL VALUES
(3,0.135,0.53,-1.034,1.244,-0.308,1.067,0.734,0.727,-0.23,-0.143,-0.269);
INSERT INTO PAL_BSTS_DATA_TBL VALUES
(4,0.373,0.698,-1.195,0.643,0.57,1.125,0.502,0.598,0.625,-0.219,-1.006);
INSERT INTO PAL_BSTS_DATA_TBL VALUES
(5,-0.437,0.441,-1.386,0.049,-0.182,0.615,-0.226,-0.199,-0.401,-0.526,-1.124);
INSERT INTO PAL_BSTS_DATA_TBL VALUES
(6,-0.556,0.405,-0.844,0.198,-0.553,0.784,-0.518,-0.245,-0.976,-0.699,-0.504);
```

```
INSERT INTO PAL_BSTS_DATA_TBL VALUES
(7,-0.432,-0.016,-1.001,-0.401,-0.339,-0.163,-0.55,-0.871,-1.236,-0.884,-1.254);
INSERT INTO PAL_BSTS_DATA_TBL VALUES
(8,-0.46,0.271,-1.234,0.252,-0.666,0.321,-0.558,-0.359,-0.555,-0.778,-2.114);
INSERT INTO PAL_BSTS_DATA_TBL VALUES
(9,-0.698,-0.357,-1.269,-1.189,-1.911,-0.645,-1.468,-1.116,0.156,-1.182,-2.958);
INSERT INTO PAL_BSTS_DATA_TBL VALUES
(10,-0.765,-0.006,-1.326,0.242,0.175,-0.185,-0.912,-0.276,0.158,-0.917,-0.939);
INSERT INTO PAL_BSTS_DATA_TBL VALUES
(11,-0.833,-0.647,-2.124,-0.71,-0.14,-1.235,-1.924,-0.978,-0.572,-1.158,-1.758);
INSERT INTO PAL_BSTS_DATA_TBL VALUES
(12,-0.767,-0.282,-1.615,-0.489,-0.556,0.879,-0.593,-0.444,-1.992,-0.898,-0.831);
INSERT INTO PAL_BSTS_DATA_TBL VALUES
(13,-0.356,-0.503,-1.035,-0.201,-0.367,-0.34,-0.331,-0.397,-0.897,-0.844,-0.425);
INSERT INTO PAL_BSTS_DATA_TBL VALUES
(14,-0.496,-0.998,-1.356,-0.717,-0.514,-0.806,-1.27,-0.669,-0.338,-1.145,-1.21);
INSERT INTO PAL_BSTS_DATA_TBL VALUES
(15,-0.684,-0.618,-1.06,-0.012,-0.93,-0.093,-1.162,-0.805,-0.373,-1.04,-0.868);
INSERT INTO PAL_BSTS_DATA_TBL VALUES
(16,-0.953,-0.547,-1.437,0.007,-0.302,0.221,-1.277,-0.504,-0.512,-0.898,-1.441);
INSERT INTO PAL_BSTS_DATA_TBL VALUES
(17,-0.869,-0.403,-1.36,-0.386,-0.129,0.001,-1.214,-0.636,0.065,-1.069,-0.929);
INSERT INTO PAL_BSTS_DATA_TBL VALUES
(18,-0.831,-0.691,-1.553,-0.002,-0.348,-0.775,-0.428,-0.626,-0.489,-0.858,-1.033)
;
INSERT INTO PAL_BSTS_DATA_TBL VALUES
(19,-0.858,-0.793,-1.302,-0.401,-0.565,0.159,-0.657,-0.855,-0.629,-0.983,-0.942);
INSERT INTO PAL_BSTS_DATA_TBL VALUES
(20,-0.769,-0.654,-1.332,-0.052,-0.432,-0.134,-0.674,-0.562,0.569,-0.953,-1.054);
INSERT INTO PAL_BSTS_DATA_TBL VALUES
(21,-0.732,-0.78,-1.362,-0.518,-0.748,0.309,-0.366,-0.668,-0.649,-0.55,-1.276);
INSERT INTO PAL_BSTS_DATA_TBL VALUES
(22,-0.682,-0.575,-1.459,-0.39,0.024,-0.416,0.034,-0.693,-0.846,-0.701,-0.941);
INSERT INTO PAL_BSTS_DATA_TBL VALUES
(23,-0.606,-0.575,-1.522,0.589,-0.189,-0.206,-0.129,-0.681,0.457,-0.443,-0.218);
INSERT INTO PAL_BSTS_DATA_TBL VALUES
(24,-0.639,-0.792,-1.255,0.036,-0.19,1.17,-0.284,-0.483,0.024,-0.672,-1.335);
INSERT INTO PAL_BSTS_DATA_TBL VALUES
(25,-0.363,-0.189,-1.096,-0.005,-0.299,0.858,-0.292,0.131,-0.958,-0.453,0.058);
INSERT INTO PAL_BSTS_DATA_TBL VALUES
(26,0.473,0.56,-1.211,-0.024,-0.496,0.281,-0.508,-0.297,-0.025,-0.449,-0.401);
INSERT INTO PAL_BSTS_DATA_TBL VALUES
(27,0.03,0.045,-1.113,-0.113,0.441,0.599,0.283,-0.385,-0.316,-0.472,-0.597);
INSERT INTO PAL_BSTS_DATA_TBL VALUES
(28,-0.688,-0.233,-1.157,-0.408,-0.539,-0.102,-0.729,-0.575,-0.742,-0.585,-0.609)
;
INSERT INTO PAL_BSTS_DATA_TBL VALUES
(29,-0.963,-0.209,-1.211,-0.113,0.134,-0.288,-0.388,-0.535,-0.166,-0.524,-0.88);
INSERT INTO PAL_BSTS_DATA_TBL VALUES
(30,-0.879,-0.23,-1.22,-0.617,-0.03,-0.503,-0.464,-0.607,0.652,-0.326,-0.605);
INSERT INTO PAL_BSTS_DATA_TBL VALUES
(31,-1.132,-0.757,-1.427,-0.291,0.29,-0.05,-0.213,-0.851,-1.309,-0.798,-0.841);
INSERT INTO PAL_BSTS_DATA_TBL VALUES
(32,-1.031,-0.429,-1.555,-0.586,-0.283,0.264,0.132,-0.702,-0.767,-0.718,-0.369);
INSERT INTO PAL_BSTS_DATA_TBL VALUES
(33,-1.014,-0.342,-1.204,0.107,-0.195,-0.059,-0.043,-0.743,-0.671,-0.577,-0.632);
INSERT INTO PAL_BSTS_DATA_TBL VALUES
(34,-1.026,-0.819,-1.208,-0.056,-0.133,0.521,-0.828,-0.767,0.112,-0.828,0.007);
INSERT INTO PAL_BSTS_DATA_TBL VALUES
(35,-1.242,-0.724,-1.386,-0.443,-0.259,-0.437,-0.803,-0.989,-0.535,-0.737,-1.559)
;
INSERT INTO PAL_BSTS_DATA_TBL VALUES
(36,-1.018,-0.484,-1.163,-0.257,-0.645,-0.068,-0.975,-0.767,-0.896,-1.042,-1.585)
;
INSERT INTO PAL_BSTS_DATA_TBL VALUES
(37,-0.957,-0.74,-1.053,-0.52,-0.712,-0.691,-1.082,-0.695,-0.778,-0.71,-0.734);
INSERT INTO PAL_BSTS_DATA_TBL VALUES
(38,-0.985,-0.98,-0.788,-0.549,-0.602,-0.249,-0.777,-0.886,-1.033,-1.022,-1.522);
```

```
INSERT INTO PAL_BSTS_DATA_TBL VALUES
(39,-0.463,-0.43,-0.495,-0.205,-0.779,-0.413,-0.583,-0.718,-0.42,-0.628,-0.998);
INSERT INTO PAL_BSTS_DATA_TBL VALUES
(40,-0.983,-0.954,-0.712,-0.778,-1.309,-0.534,-1.448,-0.725,-0.978,-1.014,-0.917)
;
INSERT INTO PAL_BSTS_DATA_TBL VALUES
(41,-0.649,-0.58,-0.924,-0.412,-0.872,-0.5,-0.414,-0.609,-0.581,-0.946,-1.054);
INSERT INTO PAL_BSTS_DATA_TBL VALUES
(42,-0.756,-0.683,-1.117,-0.932,-0.854,-0.543,-0.748,-0.624,-0.5,-1.093,-0.283);
INSERT INTO PAL_BSTS_DATA_TBL VALUES
(43,-0.35,-0.556,-0.637,-0.584,-0.602,-0.275,-0.009,-0.576,-0.161,-0.848,-0.137);
INSERT INTO PAL_BSTS_DATA_TBL VALUES
(44,-0.7,-0.725,-1.322,-0.458,-0.563,-0.604,-1.213,-1.164,-0.955,-1.357,-0.888);
INSERT INTO PAL_BSTS_DATA_TBL VALUES
(45,-0.31,-0.12,-0.726,-0.343,0.199,-0.749,-0.74,-0.825,-0.944,-0.967,-0.006);
INSERT INTO PAL_BSTS_DATA_TBL VALUES
(46,-0.622,-0.666,-0.806,-0.756,-1.138,-0.947,-0.938,-1.363,-1.181,-1.506,-1.339)
;
INSERT INTO PAL_BSTS_DATA_TBL VALUES
(47,0.73,-0.282,-1.019,-0.001,0.057,-0.366,-0.774,-0.511,-1.127,-0.792,-0.368);
INSERT INTO PAL_BSTS_DATA_TBL VALUES
(48,-0.181,-0.145,-0.585,-0.052,-0.521,0.253,-1.157,-0.939,-0.388,-1.062,-0.547);
INSERT INTO PAL_BSTS_DATA_TBL VALUES
(49,-0.144,-0.12,-0.496,-0.414,-1.088,-0.251,-0.27,-0.856,-1.313,-1.161,0.15);
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL (
    PARAM_NAME VARCHAR(100),
    INT_VALUE INTEGER,
    DOUBLE_VALUE DOUBLE,
    STRING_VALUE VARCHAR(100)
);
INSERT INTO PAL_PARAMETER_TBL VALUES ('BURN_IN', NULL, 0.6, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('EXPECTED_MODEL_SIZE', 2, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('SEASONAL_PERIOD', 12, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('MAX_ITER', 2000, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('RANDOM_SEED', 1, NULL, NULL);
DROP TABLE PAL_BSTS_STATS_TBL;
CREATE COLUMN TABLE PAL_BSTS_STATS_TBL (
    DATA_NAME        NVARCHAR(5000),
    INCLUSION_PROB DOUBLE,
    AVG_COEFF      DOUBLE
);
DROP TABLE PAL_BSTS_DECOMP_TBL;
CREATE COLUMN TABLE PAL_BSTS_DECOMP_TBL (
    ROW_INDEX  INTEGER,
    TREND      DOUBLE,
    SEASONAL   DOUBLE,
    REGRESSION DOUBLE,
    RESIDUAL   DOUBLE
);
DROP TABLE PAL_BSTS_MODEL_TBL; -- for predict followed
CREATE COLUMN TABLE PAL_BSTS_MODEL_TBL (
    ROW_INDEX INTEGER,
    MODEL_CONTENT NVARCHAR(5000)
);
DO BEGIN
    lt_data = SELECT * FROM PAL_BSTS_DATA_TBL;
    lt_param = SELECT * FROM PAL_PARAMETER_TBL;
    CALL _SYS_AFL.PAL_BSTS_TRAIN (:lt_data, :lt_param, lt_stats, lt_decomp,
lt_model);
    INSERT INTO PAL_BSTS_STATS_TBL  SELECT * FROM :lt_stats;
    INSERT INTO PAL_BSTS_DECOMP_TBL SELECT * FROM :lt_decomp;
    INSERT INTO PAL_BSTS_MODEL_TBL  SELECT * FROM :lt_model;
END;
select * from PAL_BSTS_STATS_TBL;
select * from PAL_BSTS_DECOMP_TBL;
select * from PAL_BSTS_MODEL_TBL;
```

## Expected Result

STATS table:

| | DATA_NAME | INCLUSION_PROB | AVG_COEFF |
|---|---|---|---|
| 1 | FEATURE_08 | 0.485 | 0.17386063698656593 |
| 2 | FEATURE_01 | 0.4025 | 0.4378367662259623 |
| 3 | FEATURE_07 | 0.24625 | 0.18936175210938724 |
| 4 | FEATURE_09 | 0.23375 | 0.0813390675889522 |
| 5 | FEATURE_02 | 0.1975 | 0.09869311964906032 |
| 6 | FEATURE_04 | 0.14375 | 0.13013775765697738 |
| 7 | FEATURE_06 | 0.14125 | 0.06254361775385446 |
| 8 | FEATURE_10 | 0.10375 | 0.003326742672612815 |
| 9 | FEATURE_03 | 0.08875 | 0.009415066641457203 |
| 10 | FEATURE_05 | 0.0875 | 0.021849470925825138 |

DECOMPOSE table:

| | ID | TREND | SEASONAL | REGRESSION | RESIDUAL |
|---|---|---|---|---|---|
| 1 | 0 | 1.446124882129731 | 0.3032283157136452 | 0.45964480179789646 | 0.32700200035872706 |
| 2 | 1 | 0.7036434470887771 | 0.07395345054854312 | 0.20217105490533385 | -0.09776795254265402 |
| 3 | 2 | 0.10524913642512951 | -0.014033417751938605 | 0.20026865629050072 | -0.36848437496369163 |
| 4 | 3 | 0.0019214982321888077 | 0.1418575935126458 | 0.08905695972544389 | -0.09783605147027849 |
| 5 | 4 | 0.010727764123376655 | 0.013843195101120566 | 0.19412336248773093 | 0.15430567828777184 |
| 6 | 5 | -0.1676797573848211 | -0.2470190998403802 | -0.006602661495471006 | -0.015698481279327714 |
| 7 | 6 | -0.26798381008055583 | -0.1843901099465246 | -0.06522678354807418 | -0.03839929642484541 |
| 8 | 7 | -0.2943246691518285 | 0.014649391137762821 | -0.19585008809414353 | 0.043525366108209174 |
| 9 | 8 | -0.3703359040972245 | -0.06756487423973204 | -0.07144414649945681 | 0.04934492483641334 |
| 10 | 9 | -0.5054463327869585 | -0.008509677683663664 | -0.20017775392361142 | 0.01613376439423364 |
| 11 | 10 | -0.5913208246040814 | -0.07579523005987553 | -0.04844642621898045 | -0.04943751911706262 |
| 12 | 11 | -0.6025040766347264 | 0.10353335974629051 | -0.29401653784880294 | -0.04001274526276122 |
| 13 | 12 | -0.5326283278724755 | 0.18038744346723284 | -0.30137260748836764 | -0.11338650810638973 |
| 14 | 13 | -0.3575963332243108 | 0.0973438649730141 | -0.22924487881955344 | 0.13349734707085015 |
| 15 | 14 | -0.38388684911069626 | 0.07347363976724633 | -0.30647896148818393 | 0.12089217083163395 |
| 16 | 15 | -0.5196260936012225 | 0.14517901934390953 | -0.2458800283711157 | -0.06367289737157145 |
| 17 | 16 | -0.5665758404319092 | -0.06273267981138722 | -0.22476898239083898 | -0.09892249736586456 |
| 18 | 17 | -0.51724560128828 | -0.21896932899936328 | -0.15507466898040528 | 0.02228959926804852 |
| 19 | 18 | -0.4699169615073646 | -0.16589579447383732 | -0.25029786711987034 | 0.05511062310107229 |
| 20 | 19 | -0.5226864054874428 | -0.002084769525823725 | -0.2928201585381773 | -0.04040866644855623 |
| 21 | 20 | -0.5184998738156927 | -0.07893428754200045 | -0.15134465159310978 | -0.020221187049196954 |
| 22 | 21 | -0.47486735783695955 | -0.004995732552377895 | -0.2775699982410622 | 0.025433088630399725 |
| 23 | 22 | -0.43840641933206487 | -0.057549852430135884 | -0.24694288390746255 | 0.06089915566966331 |
| 24 | 23 | -0.5100701900402244 | 0.13715465411409447 | -0.1367581745351625 | -0.09632628953870759 |
| 25 | 24 | -0.4474916592860938 | 0.11384978931711823 | -0.2009620036519826 | -0.10439612637904183 |
| 26 | 25 | -0.24771686751604816 | 0.06438854548780207 | -0.14435386322960292 | -0.03531781474215101 |
| 27 | 26 | -0.0030382522772302322 | 0.15314408784444264 | 0.037497099244851124 | 0.28539706518793645 |
| 28 | 27 | -0.15399144710452614 | 0.15514397401871668 | -0.0552544503879026 | 0.08410192347371206 |
| 29 | 28 | -0.37211428107180233 | -0.10030887798056426 | -0.18120999639430665 | -0.034366844553326725 |
| 30 | 29 | -0.5619832449834735 | -0.2151545522285234 | -0.11084179111094547 | -0.07502041167705753 |
| 31 | 30 | -0.6496965839573852 | -0.1647155937565688 | -0.049820882703562136 | -0.014766939582483984 |
| 32 | 31 | -0.7519425855145766 | 0.00008351299305848665 | -0.3231354808642722 | -0.05700544661420959 |
| 33 | 32 | -0.7615374185459727 | -0.08321063298073397 | -0.22064083504909016 | 0.03438888657579686 |
| 34 | 33 | -0.8268545590484296 | -0.00798731673483306 | -0.1898056802771465 | 0.010647556060409125 |
| 35 | 34 | -0.8516158796051976 | -0.06770902610631184 | -0.21817276231403987 | 0.11149766802554928 |
| 36 | 35 | -0.9799650997890526 | 0.18951998299595818 | -0.27316797876697185 | -0.17838690443993377 |
| 37 | 36 | -0.8311410552311042 | 0.07311641228534155 | -0.26013924030150126 | 0.00016388324726390735 |
| 38 | 37 | -0.7556726592968595 | 0.07687318482031022 | -0.28714794586638365 | 0.008947420342932833 |
| 39 | 38 | -0.6978948923849896 | 0.1250636793769389 | -0.35497042142575563 | -0.05719836556619373 |
| 40 | 39 | -0.5346391203272357 | 0.18873722021005512 | -0.18694151854178123 | 0.06984341865896176 |

MODEL table:

| | ROW_INDEX | |
|---|---|---|
| 1 | 0 | {"seed":1,"period":12,"lenData":50,"stateSize":14,"numPredictors":10,"meanBeta":[0.1762292984059498,0.01949189113068 |
| 2 | 1 | 7772,0.07109158826024878,0.057785212624079689,0.08211448454062595,0.14374100635325477,0.117950240710447 |
| 3 | 2 | 29644384563,0.014195071993491395,0.010540037492401,0.013655713583914275,0.018196457156818996,0.02251539 |
| 4 | 3 | 1029087,0.020395789841956895,0.016910746369137365,0.013418092807880559,0.01064660235873009,0.0093920325 |
| 5 | 4 | Q_val":[0.015381495654235368,0.004239572658748407,0.005222247284788116]},{"Q_val":[0.0171613532301876,0.0035 |
| 6 | 5 | :[0.08680679416939098,0.003842946603812735,0.007569763560813947 9]},{"Q_val":[0.08951313900011386,0.00264966 |
| 7 | 6 | 004477460364384234,0.003431388328062538 7]},{"Q_val":[0.01909879436598598,0.0032877661093744577,0.00512894 |
| 8 | 7 | 1838383,0.011433296583963738,0.0027517783541040356]},{"Q_val":[0.032275373381164668,0.012686633279637424,0 |
| 9 | 8 | al":[0.01242915405963989,0.001958740408440088,0.009628493900706068]},{"Q_val":[0.009575013167537488,0.001804 |
| 10 | 9 | 49374675469398]},{"Q_val":[0.012373072156351372,0.006689516306395337 8,0.003082300441915435 5]},{"Q_val":[0.006 |
| 11 | 10 | 1878669045121137,0.00352519102784044,0.011486053885222969]},{"Q_val":[0.016598388500943974,0.005323988362 |
| 12 | 11 | ":[0.051144925227225728,0.05996121577367061,0.001893184076891696 9]},{"Q_val":[0.06374512194103445,0.0677703 |
| 13 | 12 | 4367381873]},{"Q_val":[0.03335052123245898,0.038188747105426159,0.009187328072815533]},{"Q_val":[0.0533030000 |
| 14 | 13 | 13502455297655428]},{"Q_val":[0.1414683843234032,0.005324737128131975,0.011288950368610622]},{"Q_val":[0.0833 |
| 15 | 14 | 4974512235852,0.009761467213629994,0.006854787284359715]},{"Q_val":[0.03022317549234309,0.006577556561895 |
| 16 | 15 | 379443456439926]},{"Q_val":[0.09984252118148016,0.0010061775759586157,0.022948195335146965]},{"Q_val":[0.1169 |
| 17 | 16 | 8093206341328847,-0.5071485309536645,1.0]},{"stateVal":[-0.07220334366826009,0.009988862945833344,-0.02229034 |
| 18 | 17 | 208178,-0.04822027324947057,-0.023075811131624740,-0.14112379465028428,-0.18211606589356667,-0.12528917430 |

# _SYS_AFL.PAL_BSTS_PREDICT

## Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | <FORECAST_DATA table> | IN |
| 2 | <MODEL table> | IN |
| 3 | <PARAMETER table> | IN |
| 4 | <FORECAST table> | OUT |
| 5 | <DECOMPOSE table> | OUT |

## Input Table

| Table | Column | Data Type | Description | Dependency |
|---|---|---|---|---|
| FORECAST_DATA | 1st Column | INTEGER | Time stamp. Time stamp does not need to be in order, but must be unique and equal sampling | |
| | Other columns | INTEGER, DOUBLE, or DECIMAL(p,s) | Exogenous contemporaneous time series value. | Columns must be in the same order as DATA table. The forecast length is the same as the length of data in this table. |

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| MODEL | 1st Column | INTEGER | ROW_INDEX | Indicates the id of current row. |
| | 2nd Column | NVARCHAR | MODEL_CONTENT | JSON string. |

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| FORECAST_LENGTH | INTEGER | 1 | Number of points to forecast. | Valid only when the FORECAST_DATA table is empty and DATA table in BSTS_TRAIN only contains target time series. Otherwise the forecast length is the same as the length of the data in FORECAST_DATA table. |

### Output Table

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| FORECAST | 1st Column | INTEGER | TIME_STAMP | Time stamp that is monotonically increasing sorted. |
| | 2nd Column | DOUBLE | FORECAST | Forecast value. |
| | 3rd Column | DOUBLE | SE | Standard error. |
| | 4th Column | DOUBLE | LOWER_80 | Lower 80% value. |
| | 5th Column | DOUBLE | UPPER_80 | Upper 80% value. |
| | 6th Column | DOUBLE | LOWER_95 | Lower 95% value. |
| | 7th Column | DOUBLE | UPPER_95 | Upper 95% value. |

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| DECOMPOSE | 1st Column | INTEGER | TIME_STAMP | Time stamp that is monotonically increasing sorted. |
| | 2nd Column | DOUBLE | TREND | The trend component. |
| | 3rd Column | DOUBLE | SEASONAL | The seasonal component. |

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| | 4th Column | DOUBLE | EXOGENOUS | The contributions of exogenous data to forecast values. |

### Examples

Assume that:

- DM_PAL is a schema belonging to USER1;
- USER1 has been assigned the AFL__SYS_AFL_AFLPAL_EXECUTE or AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION role.

### Example

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_BSTS_FORECAST_DATA_TBL;
CREATE COLUMN TABLE PAL_BSTS_FORECAST_DATA_TBL (
    "TIME_STAMP" INTEGER,
     "FEATURE_01" DOUBLE,
    "FEATURE_02" DOUBLE,
    "FEATURE_03" DOUBLE,
    "FEATURE_04" DOUBLE,
    "FEATURE_05" DOUBLE,
    "FEATURE_06" DOUBLE,
    "FEATURE_07" DOUBLE,
    "FEATURE_08" DOUBLE,
    "FEATURE_09" DOUBLE,
    "FEATURE_10" DOUBLE
);
INSERT INTO PAL_BSTS_FORECAST_DATA_TBL VALUES
(50,0.471,-0.66,-0.086,-0.421,0.082,-0.607,-1.107,-0.559,-1.404,-1.646);
INSERT INTO PAL_BSTS_FORECAST_DATA_TBL VALUES
(51,0.872,0.062,0.481,0.931,0.901,0.192,-0.729,0.894,-0.754,1.107);
INSERT INTO PAL_BSTS_FORECAST_DATA_TBL VALUES
(52,0.976,-0.003,0.824,-0.089,1.725,1.157,-0.589,0.133,0.007,-0.115);
INSERT INTO PAL_BSTS_FORECAST_DATA_TBL VALUES
(53,0.446,0.231,0.098,0.241,-0.445,0.71,-0.014,0.182,-0.465,-1.062);
INSERT INTO PAL_BSTS_FORECAST_DATA_TBL VALUES
(54,0.248,-0.142,0.174,-0.558,-0.123,0.464,-0.38,1.236,-0.552,-1.051);
INSERT INTO PAL_BSTS_FORECAST_DATA_TBL VALUES
(55,-0.319,-0.867,0.334,-0.348,0.285,0.942,-0.16,-0.488,-0.65,-0.769);
INSERT INTO PAL_BSTS_FORECAST_DATA_TBL VALUES
(56,-0.194,-0.822,0.523,-0.297,-0.198,0.566,-0.566,-0.289,-0.596,-0.559);
INSERT INTO PAL_BSTS_FORECAST_DATA_TBL VALUES
(57,-0.357,-0.564,-0.391,-0.318,-0.531,-0.524,-0.98,0.578,-0.948,-0.87);
INSERT INTO PAL_BSTS_FORECAST_DATA_TBL VALUES
(58,-0.76,-1.113,-0.178,-0.882,-0.446,-0.228,-0.477,-0.705,-1.199,-0.517);
INSERT INTO PAL_BSTS_FORECAST_DATA_TBL VALUES
(59,-0.611,-1.163,0.186,-0.325,-0.43,-0.4,-0.976,-0.576,-0.927,-1.577);
TRUNCATE TABLE PAL_PARAMETER_TBL;
CALL _SYS_AFL.PAL_BSTS_PREDICT (PAL_BSTS_FORECAST_DATA_TBL, PAL_BSTS_MODEL_TBL,
PAL_PARAMETER_TBL, ?, ?);
```

### Expected Result

FORECAST table:

| | TIME_STAMP | FORECAST | SE | LOWER_80 | UPPER_80 | LOWER_95 | UPPER_95 |
|---|---|---|---|---|---|---|---|
| 1 | 50 | 0.14315112958561305 | 0.5912309488476427 | -0.6145420753719811 | 0.9008443345432072 | -1.0156402458416083 | 1.3019425050128342 |
| 2 | 51 | 0.46940549865082326 | 0.7655577804690683 | -0.5116966060248722 | 1.4505076033265187 | -1.0310601909884536 | 1.9698711882901003 |
| 3 | 52 | 0.1558133127902846 | 1.004785992407843 | -1.1318721853519715 | 1.4434988109325406 | -1.8135310600333612 | 2.1251576856139303 |
| 4 | 53 | 0.055188066921518925 | 1.1606546177179766 | -1.4322511797241895 | 1.5426273135672275 | -2.2196532002394775 | 2.330029334082515 |
| 5 | 54 | 0.06448119262215338 | 1.3850783196110426 | -1.7105686980320174 | 1.839531083276324 | -2.650222450995984 | 2.779184836240291 |
| 6 | 55 | 0.04584379546416252 | 1.6608935242015443 | -2.082677622263375 | 2.1743652131917 | -3.2094477198039932 | 3.3011353107323185 |
| 7 | 56 | -0.039227003739097537 | 1.9051154375307457 | -2.4807315029374997 | 2.402277495459305 | -3.773184677143608 | 3.694730669665413 |
| 8 | 57 | 0.12408405306265814 | 2.1931566701581264 | -2.6865602638918293 | 2.9347283700171456 | -4.174424066807144 | 4.42259217293246 |
| 9 | 58 | -0.20058824209869436 | 2.479857629049099 | -3.3786547463218253 | 2.977478262124437 | -5.061019920160283 | 4.659843435962894 |
| 10 | 59 | 0.3391817113472171 | 2.763763762918892 | -3.202725266549015 | 3.881088689243449 | -5.077695768478346 | 5.756059191172781 |

DECOMPOSE table:

| | TIME_STAMP | TREND | SEASONAL | |
|---|---|---|---|---|
| 1 | 50 | 0.06556105280054766 | 0.13490728841441008 | [{"attr":"FEATURE_01","val":0.08300399954920236,"pct":0.35433526093504477},{"attr":"FEATURE_07","" |
| 2 | 51 | 0.08441543869586482 | 0.18082231231945636 | [{"attr":"FEATURE_01","val":0.1536719482099882 3,"pct":0.5126875245632561},{"attr":"FEATURE_08","va |
| 3 | 52 | 0.11330635805610623 | -0.12635991081849976 | [{"attr":"FEATURE_01","val":0.171999795244207,"pct":0.7571347114567715},{"attr":"FEATURE_07","val": |
| 4 | 53 | 0.12878705419996342 | -0.17496579438767457 | [{"attr":"FEATURE_01","val":0.0785926708905361,"pct":0.6560008707233602},{"attr":"FEATURE_08","va |
| 5 | 54 | 0.13771673523247663 | -0.19178393167769292 | [{"attr":"FEATURE_08","val":0.10455100653635227,"pct":0.5386498180806059},{"attr":"FEATURE_01","va |
| 6 | 55 | 0.15022646291976477 | 0.02075369828832844 | [{"attr":"FEATURE_01","val":-0.05621714619149799,"pct":0.3753729223083 0068},{"attr":"FEATURE_08", |
| 7 | 56 | 0.17831560613908462 | -0.11026797580321109 | [{"attr":"FEATURE_01","val":-0.03418848389075426,"pct":0.2768040064399312},{"attr":"FEATURE_07","\ |
| 8 | 57 | 0.20790493884007602 | 0.01781692760407656 | [{"attr":"FEATURE_01","val":-0.06291385953092409,"pct":0.3178538704451703},{"attr":"FEATURE_08","\ |
| 9 | 58 | 0.22125003097056178 | -0.1652867132154723 | [{"attr":"FEATURE_01","val":-0.13393426678852186,"pct":0.4803282689943891 5},{"attr":"FEATURE_08", |
| 10 | 59 | 0.24386208227974884 | 0.3540682179503503 | [{"attr":"FEATURE_01","val":-0.10767610132603534,"pct":0.4264159172109062 9},{"attr":"FEATURE_08", |

## 5.5.7 Brown Exponential Smoothing

The brown exponential smoothing model is suitable to model the time series with trend but without seasonality. In PAL, both non-adaptive and adaptive brown linear exponential smoothing are provided.

For non-adaptive brown exponential smoothing, let $S_t$ and $T_t$ be the simply smoothed value and doubly smoothed value for the (t+1)-th time period, respectively. Let $a_t$ and $b_t$ be the intercept and the slope. The procedure is as follows:

1. Initialization:
   $S_0 = x_0$
   $T_0 = x_0$
   $a_0 = 2S_0 - T_0$
   $b_0 = \frac{\alpha}{1-\alpha} \times (S_0 - T_0)$
   $F_1 = a_0 + b_0$
2. Calculation:
   $S_t = \alpha x_t + (1 - \alpha)S_{t-1}$
   $T_t = \alpha S_t + (1 - \alpha) T_{t-1}$
   $a_t = 2S_t - T_t$
   $b_t = \frac{\alpha}{1-\alpha} \times (S_t - T_t)$
   $F_{t+1} = a_t + b_t$

For adaptive brown exponential smoothing, you need to update the parameter of α for every forecasting. The following rules must be satisfied.

1. Initialization:
   $S_0 = x_0$

$T_0 = x_0$
$a_0 = 2S_0 - T_0$
$b_0 = \frac{\alpha}{1-\alpha} \times (S_0 - T_0)$
$F_1 = a_0 + b_0$
$A_0 = M_0 = 0$
$\alpha_1 = \alpha_2 = \alpha_3 = \delta = 0.2$

2. Calculation:
   $E_t = x_t - F_t$
   $A_t = \delta E_t + (1 - \delta)A_{t-1}$
   $M_t = \delta|E_t| + (1 - \delta)M_{t-1}$
   $\alpha_t = |\frac{At}{Mt}| \quad (t \geq 4)$
   $St = \alpha_t x_t + (1 - \alpha_t)S_{t-1}$
   $T_t = \alpha_t S_t + (1 - \alpha_t)T_{t-1}$
   $a_t = 2S_t - T_t$
   $b_t = \frac{\alpha}{1-\alpha} \times (S_t - T_t)$
   $F_{t+1} = a_t + b_t$

Where α, δ ∈(0,1) are two user specified parameters. The model can be viewed as two coupled single exponential smoothing models, and thus forecast can be made by the following equation:

$F_{T+m} = a_T + mb_T$

> **Note**
>
> $F_0$ is not defined because there is no estimation for the time slot 0. According to the definition, you can get $F_1 = a_0 + b_0$ and so on.

## Prerequisites

- No missing or null data in the inputs.
- The data is numeric, not categorical.
- The minimum number of valid historical data is two.

## _SYS_AFL.PAL_BROWN_EXPSMOOTH_INTERVAL

This is a brown exponential smoothing function.

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |

| Position | Table Name | Parameter Type |
|---|---|---|
| 2 | `<PARAMETER table>` | IN |
| 3 | `<FORECAST table>` | OUT |
| 4 | `<STATISTICS table>` | OUT |

### Input Table

| Table | Column | Data Type | Description |
|---|---|---|---|
| DATA | 1st column | INTEGER | ID |
| | 2nd column | INTEGER, DOUBLE, or DECIMAL(p, s) | Raw data |

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| ALPHA | DOUBLE | 0.1 (Brown exponential smoothing)<br>0.2 (Adaptive brown exponential smoothing) | The smoothing constant alpha for brown exponential smoothing or the initialization value for adaptive brown exponential smoothing ($0 < \alpha < 1$). | |
| DELTA | DOUBLE | 0.2 | Value of weighted for $A_t$ and $M_t$. | Only valid when ADAPTIVE_METHOD is 1. |
| FORECAST_NUM | INTEGER | 0 | Number of values to be forecast. | |
| ADAPTIVE_METHOD | INTEGER | 0 | • 0: Brown exponential smoothing<br>• 1: Adaptive brown exponential smoothing | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| MEASURE_NAME | VARCHAR | No default value | Measure name. Supported measures are MPE, MSE, RMSE, ET, MAD, MASE, WMAPE, SMAPE, and MAPE.<br>For detailed information on measures, see *Forecast Accuracy Measures*. | |
| IGNORE_ZERO | INTEGER | 0 | • 0: Uses zero values in the input dataset when calculating MPE or MAPE.<br>• 1: Ignores zero values in the input dataset when calculating MPE or MAPE. | Only valid when MEASURE_NAME is MPE or MAPE. |
| EXPOST_FLAG | INTEGER | 1 | • 0: Does not output the expost forecast, and just outputs the forecast values.<br>• 1: Outputs the expost forecast and the forecast values. | |

### Output Tables

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| FORECAST | 1st column | INTEGER | TIMESTAMP | ID |
| | 2nd column | DOUBLE | VALUE | Output result |
| STATISTICS | 1st column | NVARCHAR(100) | STAT_NAME | Name of statistics |
| | 2nd column | DOUBLE | STAT_VALUE | Value of statistics |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;

- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA DM_PAL;
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME " VARCHAR(100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (100)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('ALPHA', NULL,0.1, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('DELTA',NULL, 0.2, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FORECAST_NUM',6, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('EXPOST_FLAG',1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('ADAPTIVE_METHOD',0, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MEASURE_NAME', NULL, NULL, 'MSE');
DROP TABLE PAL_BROWNSMOOTH_DATA_TBL;
CREATE COLUMN TABLE PAL_BROWNSMOOTH_DATA_TBL ("ID" INT, "RAWDATA" DOUBLE);
INSERT INTO PAL_BROWNSMOOTH_DATA_TBL VALUES (1,143.0);
INSERT INTO PAL_BROWNSMOOTH_DATA_TBL VALUES (2,152.0);
INSERT INTO PAL_BROWNSMOOTH_DATA_TBL VALUES (3,161.0);
INSERT INTO PAL_BROWNSMOOTH_DATA_TBL VALUES (4,139.0);
INSERT INTO PAL_BROWNSMOOTH_DATA_TBL VALUES (5,137.0);
INSERT INTO PAL_BROWNSMOOTH_DATA_TBL VALUES (6,174.0);
INSERT INTO PAL_BROWNSMOOTH_DATA_TBL VALUES (7,142.0);
INSERT INTO PAL_BROWNSMOOTH_DATA_TBL VALUES (8,141.0);
INSERT INTO PAL_BROWNSMOOTH_DATA_TBL VALUES (9,162.0);
INSERT INTO PAL_BROWNSMOOTH_DATA_TBL VALUES (10,180.0);
INSERT INTO PAL_BROWNSMOOTH_DATA_TBL VALUES (11,164.0);
INSERT INTO PAL_BROWNSMOOTH_DATA_TBL VALUES (12,171.0);
INSERT INTO PAL_BROWNSMOOTH_DATA_TBL VALUES (13,206.0);
INSERT INTO PAL_BROWNSMOOTH_DATA_TBL VALUES (14,193.0);
INSERT INTO PAL_BROWNSMOOTH_DATA_TBL VALUES (15,207.0);
INSERT INTO PAL_BROWNSMOOTH_DATA_TBL VALUES (16,218.0);
INSERT INTO PAL_BROWNSMOOTH_DATA_TBL VALUES (17,229.0);
INSERT INTO PAL_BROWNSMOOTH_DATA_TBL VALUES (18,225.0);
INSERT INTO PAL_BROWNSMOOTH_DATA_TBL VALUES (19,204.0);
INSERT INTO PAL_BROWNSMOOTH_DATA_TBL VALUES (20,227.0);
INSERT INTO PAL_BROWNSMOOTH_DATA_TBL VALUES (21,223.0);
INSERT INTO PAL_BROWNSMOOTH_DATA_TBL VALUES (22,242.0);
INSERT INTO PAL_BROWNSMOOTH_DATA_TBL VALUES (23,239.0);
INSERT INTO PAL_BROWNSMOOTH_DATA_TBL VALUES (24,266.0);
CALL _SYS_AFL.PAL_BROWN_EXPSMOOTH_INTERVAL(PAL_BROWNSMOOTH_DATA_TBL,
#PAL_PARAMETER_TBL, ?,?);
```

### Expected

FORECAST:

| | TIMESTAMP | VALUE | PI1_LOWER | PI1_UPPER | PI2_LOWER | PI2_UPPER |
|---|---|---|---|---|---|---|
| 1 | 2 | 143 | ? | ? | ? | ? |
| 2 | 3 | 144.79999999999998 | ? | ? | ? | ? |
| 3 | 4 | 148.13 | ? | ? | ? | ? |
| 4 | 5 | 146.55599999999998 | ? | ? | ? | ? |
| 5 | 6 | 144.80549999999994 | ? | ? | ? | ? |
| 6 | 7 | 150.70953999999995 | ? | ? | ? | ? |
| 7 | 8 | 149.32471699999996 | ? | ? | ? | ? |
| 8 | 9 | 147.9297631999999 | ? | ? | ? | ? |
| 9 | 10 | 150.9305529899999 | ? | ? | ? | ? |
| 10 | 11 | 157.0718871899999 | ? | ? | ? | ? |
| 11 | 12 | 159.07564902009992 | ? | ? | ? | ? |
| 12 | 13 | 162.14793961227988 | ? | ? | ? | ? |
| 13 | 14 | 171.72501559582292 | ? | ? | ? | ? |
| 14 | 15 | 177.2251969865346 | ? | ? | ? | ? |
| 15 | 16 | 184.6380919431457 | ? | ? | ? | ? |
| 16 | 17 | 193.0661559385692 | ? | ? | ? | ? |
| 17 | 18 | 202.34222621547653 | ? | ? | ? | ? |
| 18 | 19 | 209.3224208776167 | ? | ? | ? | ? |
| 19 | 20 | 210.9331543451741 | ? | ? | ? | ? |
| 20 | 21 | 216.7685169104438 | ? | ? | ? | ? |
| 21 | 22 | 220.79747541920784 | ? | ? | ? | ? |
| 22 | 23 | 227.8829570571146 | ? | ? | ? | ? |
| 23 | 24 | 233.163367613248 | ? | ? | ? | ? |
| 24 | 25 | 242.89886648758352 | 222.4711737996572 | 263.32655917550983 | 211.65740650109154 | 274.1403264740755 |
| 25 | 26 | 246.39540520843616 | 225.56316448175357 | 267.22764593511874 | 214.535242387586 | 278.25556802928634 |
| 26 | 27 | 249.89194392928883 | 228.62260518983547 | 271.1612826687422 | 217.3632973846638 | 282.4205904739138 |
| 27 | 28 | 253.38848265014147 | 231.64953978634182 | 275.12742551394115 | 220.14163857413655 | 286.6353267261464 |
| 28 | 29 | 256.88502137099414 | 234.6441510848303 | 279.125891657158 | 222.87054554521347 | 290.8994971967748 |
| 29 | 30 | 260.3815600918468 | 237.60674392814127 | 283.1563762555523 | 225.55048451478683 | 295.2126356689067 |

STATISTICS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | MSE | 474.1420040776227 |

> **Note**
>
> The wrapper procedure generation approach to call PAL functions is still supported. For more information, refer to the older versions (prior to SAP HANA Platform 2.0 SPS 02) of this document.

## Related Information

Forecast Accuracy Measures [page 762]

## 5.5.8 Change-Point Detection

Change-point detection (CPDetection) methods aim at detecting multiple abrupt changes such as change in mean, variance or distribution in an observed time-series data.

Some consider the common approach of detecting changepoints through minimizing a cost function over both the number of changepoints and their locations. However, minimizing the log-likelihood function for a segmentation directly will lead to over-fitting. This issue can be solved by either introducing a penalty term or fixing the total number of changepoints, which are the so-called penalised and constrained optimisation method.

First, the initial dynamic programming (DP) approach, Optimal Partitioning (Opt), was implemented. It consists in recursively solving the sub-problems of minimisation over the position of the last changepoint. With a linear penalty term β, the minimum value of the penalised cost F(t), could be written as,

$$F(\tau)=\min_{\tau<t}\{F(\tau)+C_{y_{\tau+1:t}}+\beta\}$$

Where $C_{y_{\tau+1:t}}$ denotes the cost for the segment from $y_{\tau+1}$ to $y_t$. The computation takes $O(n^2)$ time.

To speed up, a first type of pruning method was applied. It is proved that if the following relation

$$C_{y_{s+1:t}}+C_{y_{t+1:n}}+\kappa\leq C_{y_{s+1:n}},\quad s<t<n$$

is true for some constant $\kappa O(n)$.

A difficulty of using Pelt in practice is to choose a good value of β. Too small values of penalty make Pelt sensitive even to a tiny fluctuate while too large values forbidden most of changes. Due to the discrete nature of change points, there are almost no penalties that can be suited to all cases. under some circumstances, taking the number of samples as the value of the penalty is enough, while under others, the distance between neighbouring change points makes sense. Thus, as an extension, the Adapted Pelt (AdpPelt) method was introduced to automatically optimize the penalty function based on the nonlinear relation between the total loss and the number of change points. For a range of penalties,[, then the point t can be discarded from the set of potential change points. This is the famous Pelt (Pruned exact linear time) method. Under the assumption that regime lengths are randomly drawn from a uniform distribution, the Complexity of Pelt is of order $\beta_0$,$\beta_1$ ], the next candidate point will be given as,

$$\beta_{int}=\frac{l^*_{m(\beta_1)}-l^*_{m(\beta_0)}}{m(\beta_0)-m(\beta_1)},$$

Where $m$ and $l^*_m$, then the point t can be discarded from the are the number of changepoints and the penalised total loss at m respectively.

The prunedDP algorithm is based on the minimization of the cost function that depends on some parameter, which on the other hand is affected by the changepoints. Introduce the parameter of the last segment and the cost of the best partition in k regions up to $t$,

$$H_{k,t}(\theta)=\min_{k-1<\tau\leq t}\{C_{k-1,\tau}+c([\tau+1,t],\theta)\},$$

Where τ is the last change-point position and $c(\wp\theta)$ defines the point-additive loss of a segment with parameter θ. The minimal loss $C_{k,T}$ is then obtained as $\min_{\theta}H_{k,t}(\theta)$. For each total number of regions k, from 2 to some maximal K, the PrunedDP works on a set of splitting points. Once it has been determined that when adding a new point, the values of θ changes but not the best candidates, the point will be removed and can never become optimal again. Its average computational complexity is $O(KNlog(N)))$.

## Prerequisite

- The length of the input time-series data should not be less than five.
- The input data must not have missing value.

## _AFL_SYS.PAL_CPDetection

This function automatically detects multiple changepoints.

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<RESULT table>` | OUT |
| 4 | `<STATISTICS table>` | OUT |

### Signature

### Input Table

| Table | Column | Data Type | Description | Constraint |
|---|---|---|---|---|
| DATA | 1st column | INTEGER, DOUBLE, DECIMAL(p, s), VARCHAR, or NVARCHAR | Time stamp | ID column. |
| | 2nd column | INTEGER, DOUBLE, or DECIMAL(p, s) | Raw data | |
| | Other columns | INTEGER, DOUBLE, or DECIMAL(p, s) | Covariate data | If one has. |

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| SOLVER | VARCHAR or NVARCHAR | 'Pelt' | Search methods:<br>• 'Pelt': Pruned exactly linear time<br>• 'Opt': Optimal partitioning<br>• 'AdpPelt': adapted Pelt, finding non-linear optimized penalty function automatically with Pelt method.<br>• 'PrunedDP': Pruned Dynamic Programming | |
| PENALTY | VARCHAR or NVARCHAR | 'AIC' | Penalties for penalised method.<br>• 'AIC': Akaike information criterion<br>• 'BIC': Bayesian information criterion<br>• 'mBIC': modified BIC, taking distance among points into account<br>• 'Oracle': penalty satisfying oracle-type inequality<br>• 'Custom': user-defined penalty. Penalty value should be specified at the same time. An input example: `Values ('Custom', NULL, <penalty-value> , NULL)`. | PrunedDP supports all the penalty type, while Pelt and Opt only support type of AIC, BIC and Custom. When AdpPelt is selected, Custom will be the only choice. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| COSTFUNCTION | VARCHAR or NVARCHAR | 'Normal_MSE' | Cost function.<br>• 'Normal_MSE': quadratic error loss for normal mean-shifts<br>• 'Normal_RBF': radical basis function for normal mean-shifts<br>• 'Normal_MHLB': Mahalanobis-type metric for normal mean-shifts<br>• 'Normal_MV': for normal mean-shifts and scale-shifts<br>• 'Linear': for linear mean-shifts<br>• 'Gamma': for Gamma distributed mean-shifts<br>• 'Poisson': for Poisson distributed mean-shifts<br>• 'Exponential': for exponential distributed mean-shifts<br>• 'Normal_M': for normal distributed mean-shifts<br>• 'NegBinomial': for negative-binomial distributed mean-shifts | The first eight cost functions are available in Pelt, Opt and AdpPelt, but only the last four available in PrunedDP. |
| PENALISATION_FACTOR | DOUBLE | 0.02 | Trade-off between speed and accuracy of the running. A small value (generally less than 0.1) will dramatically improve the efficiency. | Only valid in Pelt and AdpPelt. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| MIN_SIZE | INTEGER | 2 | Minimal length from the very beginning within which the change would not happen. | Only valid for penalised methods (Opt, Pelt, and AdpPelt). |
| MIN_SEP | INTEGER | 1 | Minimal length of separation from the last change point between which the change would not happen. | Only valid for penalised methods. |
| MaxK | INTEGER | 0 | Maximal number of change points, if the give value is less than 1, it would be determined atomically according to the input data. | Only valid for PrunedDP. |
| DISPERSION | DOUBLE | 1.0 | Dispersion coefficient for Gamma and negative binomial distribution. | |
| RANGED_PENALITIES | VARCHAR or NVARCHAR | '[0.01,100]' | '[$\beta_{min}$, $\beta_{max}$]', range of penalties. | Only valid for AdpPelt. |
| MAX_ITERATION | INTEGER | 40 | Maximal iteration of searching best penalty. | Only valid for AdpPelt. |

### Output Tables

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| RESULT | 1st column | VARCHAR, NVARCHAR, INTEGER, or DOUBLE | TIME_STAMP | Time stamp of change points. |
| STATS | 1st column | NVARCHAR(1000) | STAT_NAME | Statistics name. |
| | 2nd column | NVARCHAR(1000) | STAT_VALUE | Statistics value. |

### Examples

Assume that:

- DM_PAL is a schema belonging to USER1;
- USER1 has been assigned the AFL__SYS_AFL_AFLPAL_EXECUTE or AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION role.

**Example 1: Pelt with AIC penalty**

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_CPDETECTION_DATA_TBL;
CREATE COLUMN TABLE PAL_CPDETECTION_DATA_TBL (
    "TIME_STAMP" VARCHAR(10),
    "SERIES" DOUBLE
);
INSERT INTO PAL_CPDETECTION_DATA_TBL VALUES ('1-1', -5.36);
INSERT INTO PAL_CPDETECTION_DATA_TBL VALUES ('1-2', -5.14);
INSERT INTO PAL_CPDETECTION_DATA_TBL VALUES ('1-3', -4.94);
INSERT INTO PAL_CPDETECTION_DATA_TBL VALUES ('2-1', -5.15);
INSERT INTO PAL_CPDETECTION_DATA_TBL VALUES ('2-2', -4.95);
INSERT INTO PAL_CPDETECTION_DATA_TBL VALUES ('2-3', 0.55);
INSERT INTO PAL_CPDETECTION_DATA_TBL VALUES ('2-4', 0.88);
INSERT INTO PAL_CPDETECTION_DATA_TBL VALUES ('3-1', 0.95);
INSERT INTO PAL_CPDETECTION_DATA_TBL VALUES ('3-2', 0.68);
INSERT INTO PAL_CPDETECTION_DATA_TBL VALUES ('3-3', 0.86);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR(100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR(100)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SOLVER', NULL, NULL, 'Pelt');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('COSTFUNCTION', NULL, NULL, 'Normal_MSE');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PENALTY', NULL, NULL, 'AIC');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PENALISATION_FACTOR', NULL, 0.02, NULL);
CALL
_SYS_AFL.PAL_CPDETECTION(PAL_CPDETECTION_DATA_TBL,"#PAL_PARAMETER_TBL", ?, ?);
```

**Expected Result**

RESULT:

| | TIME_STAMP |
|---|---|
| 1 | 2-2 |

STATS:

| | STAT_NAME | STAT_VAL |
|---|---|---|
| 1 | solver | Pelt |
| 2 | cost function | Normal_MSE |
| 3 | penalty type | AIC |
| 4 | total loss | 4.2276 |
| 5 | penalisatio... | 0.02 |

**Example 2: AdpPelt with Custom penalty**

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_CPDETECTION_DATA_TBL;
CREATE COLUMN TABLE PAL_CPDETECTION_DATA_TBL (
```

```
    "TIME_STAMP" VARCHAR(10),
    "SERIES" DOUBLE
);
INSERT INTO PAL_CPDETECTION_DATA_TBL VALUES ('1-1', -5.36);
INSERT INTO PAL_CPDETECTION_DATA_TBL VALUES ('1-2', -5.14);
INSERT INTO PAL_CPDETECTION_DATA_TBL VALUES ('1-3', -4.94);
INSERT INTO PAL_CPDETECTION_DATA_TBL VALUES ('2-1', -5.15);
INSERT INTO PAL_CPDETECTION_DATA_TBL VALUES ('2-2', -4.95);
INSERT INTO PAL_CPDETECTION_DATA_TBL VALUES ('2-3', 0.55);
INSERT INTO PAL_CPDETECTION_DATA_TBL VALUES ('2-4', 0.88);
INSERT INTO PAL_CPDETECTION_DATA_TBL VALUES ('3-1', 0.95);
INSERT INTO PAL_CPDETECTION_DATA_TBL VALUES ('3-2', 0.68);
INSERT INTO PAL_CPDETECTION_DATA_TBL VALUES ('3-3', 0.86);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR(100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR(100)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SOLVER', NULL, NULL, 'AdpPelt');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('COSTFUNCTION', NULL, NULL, 'Normal_MV');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('RANGED_PENALITIES', NULL, NULL,
'[0.01,100]');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PENALISATION_FACTOR', NULL, 0.02, NULL);
CALL
_SYS_AFL.PAL_CPDETECTION(PAL_CPDETECTION_DATA_TBL,"#PAL_PARAMETER_TBL", ?, ?);
```

**Expected Result**

RESULT:

| | TIME_STAMP |
|---|---|
| 1 | 2-2 |

STATS:

| | STAT_NAME | STAT_VAL |
|---|---|---|
| 1 | solver | AdpPelt |
| 2 | cost function | Normal_MV |
| 3 | penalty type | Custom |
| 4 | total loss | -27.8209 |
| 5 | penalisatio... | 0.02 |
| 6 | iteration | 3 |
| 7 | optimal pe... | 5.00949 |

**Example 3: PrunedDP with Oracle penalty**

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_CPDETECTION_DATA_TBL;
CREATE COLUMN TABLE PAL_CPDETECTION_DATA_TBL (
    "TIME_STAMP" VARCHAR(10),
    "SERIES" DOUBLE
);
INSERT INTO PAL_CPDETECTION_DATA_TBL VALUES ('1-1', -5.36);
INSERT INTO PAL_CPDETECTION_DATA_TBL VALUES ('1-2', -5.14);
INSERT INTO PAL_CPDETECTION_DATA_TBL VALUES ('1-3', -4.94);
```

```
INSERT INTO PAL_CPDETECTION_DATA_TBL VALUES ('2-1', -5.15);
INSERT INTO PAL_CPDETECTION_DATA_TBL VALUES ('2-2', -4.95);
INSERT INTO PAL_CPDETECTION_DATA_TBL VALUES ('2-3', 0.55);
INSERT INTO PAL_CPDETECTION_DATA_TBL VALUES ('2-4', 0.88);
INSERT INTO PAL_CPDETECTION_DATA_TBL VALUES ('3-1', 0.95);
INSERT INTO PAL_CPDETECTION_DATA_TBL VALUES ('3-2', 0.68);
INSERT INTO PAL_CPDETECTION_DATA_TBL VALUES ('3-3', 0.86);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR(100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR(100)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SOLVER', NULL, NULL, 'PrunedDP');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('COSTFUNCTION', NULL, NULL, 'Normal_M');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PENALTY', NULL, NULL, 'Oracle');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MaxK', 3, NULL, NULL);
CALL
_SYS_AFL.PAL_CPDETECTION(PAL_CPDETECTION_DATA_TBL,"#PAL_PARAMETER_TBL", ?, ?);
```

**Expected Result**

RESULT:

| | TIME_STAMP |
|---|---|
| 1 | 2-2 |

STATS:

| | STAT_NAME | STAT_VAL |
|---|---|---|
| 1 | solver | PrunedDP |
| 2 | cost function | Normal_M |
| 3 | penalty type | Oracle |
| 4 | total loss | 0.000981... |
| 5 | kMax | 3 |

## 5.5.9 Croston's Method

The Croston's method is a forecast strategy for products with intermittent demand.

The Croston's method consists of two steps. First, separate exponential smoothing estimates are made of the average size of a demand. Second, the average interval between demands is calculated. This is then used in a form of the constant model to predict the future demand.

**Initialization**

The system estimates the demand volume and interval between demands based on historical demands and intervals. It also takes smoothing factor into consideration.

```
V(t) = Historical value
```

```
P(t) = Forecasted value
```

```
q = Interval between last two periods with demand

α = Smoothing factor for the estimates

Z = Estimate of demand volume

X = Estimate of intervals between demand

Z(0) = Average of Non-zero Demands

X(0) = Average Interval between Non-Zero Demands
```

**Calculation of Forecast Parameters**

The forecast is made using a modified constant model. The forecast parameters P and X are determined as follows:

```
If V(t) = 0
q = q + 1
Else
Z(t) = Z(t−1) + α[V(t) − Z(t−1)]
X(t) = X(t−1) + α[q − X(t−1)]
Endif
```

In the last iteration, the parameters Z(f) and X(f) will be delivered for the forecast. The Croston's method supports two different mode to output: constant mode and sporadic mode:

* Constant mode: average demand between two non-zero demands, which means Z(t)=Z(t-1) if V(t)=0, and then Z(t) and X(t) will be delivered for the forecast;

* Sporadic mode: predicts the demand at the non-zero demand time, which means Z(t)=0 if V(t)=0, and then Z(t) will be delivered for the forecast with X(t) interval time.

## Prerequisites

- No missing or null data in the inputs.
- The data is numeric, not categorical.

## _SYS_AFL.PAL_CROSTON

This is a Croston's method function.

**Overview of all Tables**

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |

| Position | Table Name | Parameter Type |
|---|---|---|
| 3 | `<RESULT table>` | OUT |
| 4 | `<STATISTICS table>` | OUT |

### Input Table

| Table | Column | Reference for Output Table | Data Type | Description |
|---|---|---|---|---|
| DATA | 1st column | TS_ID | INTEGER | ID |
| | 2nd column | TS_DATA | INTEGER, DOUBLE, or DECIMAL(p, s) | Raw data |

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| ALPHA | DOUBLE | 0.1 | Value of the smoothing constant alpha ($0 < \alpha < 1$). | |
| FORECAST_NUM | INTEGER | 0 | Number of values to be forecast. When it is set to 1, the algorithm only forecasts one value. | |
| METHOD | INTEGER | 0 | • 0: Uses the sporadic method<br>• 1: Uses the constant method | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| MEASURE_NAME | VARCHAR | No default value | Measure name. Supported measures are MPE, MSE, RMSE, ET, MAD, MASE, WMAPE, SMAPE, and MAPE.<br>For detailed information on measures, see *Forecast Accuracy Measures*. | |
| EXPOST_FLAG | INTEGER | 1 | • 0: Does not output the expost forecast, and just outputs the forecast values.<br>• 1: Outputs the expost forecast and the forecast values. | |
| IGNORE_ZERO | INTEGER | 0 | • 0: Uses zero values in the input dataset when calculating MPE or MAPE.<br>• 1: Ignores zero values in the input dataset when calculating MPE or MAPE. | Only valid when MEASURE_NAME is MPE or MAPE. |

### Output Tables

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| RESULT | 1st column | TS_ID (in input table) data type | TS_ID (in input table) column name | ID of data |
| | 2nd column | TS_DATA (in input table) data type | TS_DATA (in input table) column name | Output result |
| STATISTICS | 1st column | NVARCHAR | STAT_NAME | Name of statistics |
| | 2nd column | DOUBLE | STAT_VALUE | Value of statistics |

### Example

Assume that:

- DM_PAL is a schema belonging to USER1;
- USER1 has been assigned the AFL__SYS_AFL_AFLPAL_EXECUTE or AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION role.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_CROSTON_DATA_TBL;
CREATE COLUMN TABLE PAL_CROSTON_DATA_TBL (
    "ID" INT,
    "RAWDATA" DOUBLE
);
INSERT INTO PAL_CROSTON_DATA_TBL VALUES (0, 0.0);
INSERT INTO PAL_CROSTON_DATA_TBL VALUES (1, 1.0);
INSERT INTO PAL_CROSTON_DATA_TBL VALUES (2, 4.0);
INSERT INTO PAL_CROSTON_DATA_TBL VALUES (3, 0.0);
INSERT INTO PAL_CROSTON_DATA_TBL VALUES (4, 0.0);
INSERT INTO PAL_CROSTON_DATA_TBL VALUES (5, 0.0);
INSERT INTO PAL_CROSTON_DATA_TBL VALUES (6, 5.0);
INSERT INTO PAL_CROSTON_DATA_TBL VALUES (7, 3.0);
INSERT INTO PAL_CROSTON_DATA_TBL VALUES (8, 0.0);
INSERT INTO PAL_CROSTON_DATA_TBL VALUES (9, 0.0);
INSERT INTO PAL_CROSTON_DATA_TBL VALUES (10, 0.0);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('ALPHA', NULL,0.1,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FORECAST_NUM',1, NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('METHOD',0, NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES('MEASURE_NAME',NULL,NULL,'MAPE');
CALL "_SYS_AFL"."PAL_CROSTON"(PAL_CROSTON_DATA_TBL, #PAL_PARAMETER_TBL, ?, ?);
```

**Expected Results**

RESULT:

| | ID | RAWDATA |
|---|---|---|
| 1 | 1 | 0 |
| 2 | 2 | 3.025 |
| 3 | 3 | 0 |
| 4 | 4 | 0 |
| 5 | 5 | 0 |
| 6 | 6 | 3.1225 |
| 7 | 7 | 3.31025 |
| 8 | 8 | 0 |
| 9 | 9 | 0 |
| 10 | 10 | 0 |
| 11 | 11 | 3.279225 |

STATISTICS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | MAPE | 0.17226666666666665 |

> **Note**
>
> The wrapper procedure generation approach to call PAL functions is still supported. For more information, refer to the older versions (prior to SAP HANA Platform 2.0 SPS 02) of this document.

### Related Information

Forecast Accuracy Measures [page 762]

## 5.5.10 Croston TSB Method

The Croston TSB method (for Teunter, Syntetos & Babai) is a forecast strategy for products with intermittent demand. It is a modification of Croston's method.

It replaces the demand interval in Croston's method by demand probability which is updated every period. Compared to Croston's method, the forecast of TSB method is not biased and its probability forecast can be used to estimate the risk of obsolescence.

The TSB model uses separate single exponential smoothing for demand probability and demand size. The estimation of prbobability is updated for every period, while the estimation of demand size is only updated when the demand is non-zero. We can get the demand interval by taking the reciprocal of demand probabilty. The TSB model can be expressed as the following formulas.

If $x_t = 0$:

$$\hat{x}_{t+1} = \hat{x}_t,$$

$$\hat{p}_{t+1} = (1-\beta)\hat{p}_t,$$

else:

$$\hat{x}_{t+1} = \alpha x_t + (1-\alpha)\hat{x}_t,$$

$$\hat{p}_{t+1} = \beta + (1-\beta)\hat{p}_t.$$

Forecast equation is as follows:

$$\hat{y}_{t+m} = \hat{x}_{t+1}\hat{p}_{t+1}.$$

Here $m = 1,2,3,...,0 \le \alpha, \beta \le 1$,

$x_t$ is the original demand of the series in period $t$,

$\hat{x}_t$ is the estimation of mean demand size for period $t$,

$\hat{p}_t$ is the estimation of the probability of a demand occurrence for period $t$,

$\hat{y}_t$ is the estimation for the average size of demands,

and α and β are parameters.

The same as the Croston's method, TSB method also supports two different algorithm types to output: constant mode and sporadic mode.

- Constant mode: average demand between two non-zero demands, which means $\hat{x}_{t+1} = \hat{x}_t$ if $x_t = 0$, then $\hat{y}_{t+1} = \hat{x}_{t+1}\hat{p}_{t+1}$, as the formula is shown.
- Sporadic mode: predicts the demand at the non-zero demand time, which means $\hat{x}_t$= 0 if $x_t = 0$, then $\hat{y}_t = \hat{x}_t$ with $1/\hat{p}_t$ interval time.

## Prerequisites

- No missing or null data in the inputs.
- The data is numeric, not categorical.

## _SYS_AFL.PAL_CROSTONTSB

This is a Croston TSB method function.

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA Table>` | IN |
| 2 | `<PARAMETER Table>` | IN |
| 3 | `<RESULT Table>` | OUT |
| 4 | `<STATISTICS Table>` | OUT |
| 5 | `<METRICS Table>` | OUT |

### Input Tables

| Table | Column | Data Type | Description | Reference for Output Table |
|---|---|---|---|---|
| DATA | 1st column | INTEGER | ID of data | TS_ID |
| | 2nd column | INTEGER, DOUBLE, DECIMAL (p,s) | Raw data | |
| PARAMETER | 1st column | VARCHAR or NVARCHAR | Parameter name | |

| Table | Column | Data Type | Description | Reference for Output Table |
|---|---|---|---|---|
| | 2nd column | INTEGER | Int value | |
| | 3rd column | DOUBLE | Double value | |
| | 4th column | VARCHAR or NVARCHAR | String value | |

### Parameter Table

Mandatory Parameters

None

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| ALPHA | DOUBLE | 0.1 | Smoothing parameter for demand. | |
| BETA | DOUBLE | 0.1 | Smoothing parameter for probability. | |
| FORECAST_NUM | INTEGER | 0 | Number of values to be forecast. When it is set to 1, the algorithm only forecasts one value. | |
| METHOD | INTEGER | 0 | • 0: Uses the sporadic method.<br>• 1: Uses the constant method. | |
| MEASURE_NAME | VARCHAR | No default value | Measure name. Supported measures are MPE, MSE, RMSE, ET, MAD, MASE, WMAPE, SMAPE, and MAPE. For detailed description, see *Forecast Accuracy Measures*. | |
| EXPOST_FLAG | INTEGER | 1 | • 0: Does not output the expost forecast, and just outputs the forecast values;<br>• 1: Outputs the expost forecast and the forecast values. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| IGNORE_ZERO | INTEGER | 0 | • 0: Uses zero values in the input dataset when calculating MAPE;<br>• 1: Ignores zero values in the input dataset when calculating MAPE. | Only valid when MEASURE_NAME is MPE or MAPE. |
| REMOVE_LEADING_ZEROS | INTEGER | 0 | • 0: Uses leading zero values in the input dataset when smoothing the probability;<br>• 1: Ignores leading zero values in the input dataset when smoothing the probability. | When it is set to 1, the leading zeros are ignored for calculating measure. |

### Output Tables

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| RESULT | 1st column | TS_ID (in input table) data type | TS_ID (in input table) column name | ID of data |
| | 2nd column | DOUBLE | OUTPUT_VALUE | Output result |
| STATISTICS | 1st column | NVARCHAR | STAT_NAME | Name of statistics |
| | 2nd column | DOUBLE | STAT_VALUE | Value of statistics |
| METRICS | 1st column | NVARCHAR | NAME | Name of metrics (“DEMAND_FORECAST” or “PROBABILITY_FORECAST”) |
| | 2nd column | DOUBLE | VALUE | Value of metrics |

In the METRICS table, “DEMAND_FORECAST” and “PROBABILITY_FORECAST” are the forecasts of single exponential smoothing of demand and probability respectively. “DEMAND_FORECAST” is the estimation of mean demand size in the future, and “PROBABILITY_FORECAST” is the estimation of mean probability of demand occurrence in the future, which reciprocal is forecast of interval. “PROBABILITY_FORECAST” can be used to estimate the risk of obsolescence.

### Examples

Assume that:

- DM_PAL is a schema belonging to USER1;
- USER1 has been assigned the AFL__SYS_AFL_AFLPAL_EXECUTE or AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION role.

**Example 1**: Constant mode

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_CROSTONTSB_DATA_TBL;
CREATE COLUMN TABLE PAL_CROSTONTSB_DATA_TBL (
    "ID" INT,
    "RAWDATA" DOUBLE
);
INSERT INTO PAL_CROSTONTSB_DATA_TBL VALUES (0, 0.0);
INSERT INTO PAL_CROSTONTSB_DATA_TBL VALUES (1, 0.0);
INSERT INTO PAL_CROSTONTSB_DATA_TBL VALUES (2, 4.0);
INSERT INTO PAL_CROSTONTSB_DATA_TBL VALUES (3, 0.0);
INSERT INTO PAL_CROSTONTSB_DATA_TBL VALUES (4, 0.0);
INSERT INTO PAL_CROSTONTSB_DATA_TBL VALUES (5, 0.0);
INSERT INTO PAL_CROSTONTSB_DATA_TBL VALUES (6, 5.0);
INSERT INTO PAL_CROSTONTSB_DATA_TBL VALUES (7, 3.0);
INSERT INTO PAL_CROSTONTSB_DATA_TBL VALUES (8, 0.0);
INSERT INTO PAL_CROSTONTSB_DATA_TBL VALUES (9, 0.0);
INSERT INTO PAL_CROSTONTSB_DATA_TBL VALUES (10, 0.0);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('ALPHA', NULL,0.1,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('BETA', NULL,0.1,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FORECAST_NUM',10, NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('METHOD',1, NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES('MEASURE_NAME',NULL,NULL,'MAPE');
INSERT INTO #PAL_PARAMETER_TBL VALUES('EXPOST_FLAG',1,NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES('REMOVE_LEADING_ZEROS',0,NULL,NULL);
CALL "_SYS_AFL"."PAL_CROSTONTSB"(PAL_CROSTONTSB_DATA_TBL,
#PAL_PARAMETER_TBL, ?, ?, ?);
```

**Expected Result 1**: Constant mode

RESULT:

| | ID | OUTPUT_VALUE |
|---|---|---|
| 1 | 1 | 1.4400000000000002 |
| 2 | 2 | 1.2960000000000003 |
| 3 | 3 | 1.5664000000000002 |
| 4 | 4 | 1.4097600000000003 |
| 5 | 5 | 1.2687840000000004 |
| 6 | 6 | 1.1419056000000003 |
| 7 | 7 | 1.4634079160000002 |
| 8 | 8 | 1.6807311771600004 |
| 9 | 9 | 1.5126580594440002 |
| 10 | 10 | 1.3613922534996004 |
| 11 | 11 | 1.2252530281496403 |
| 12 | 12 | 1.2252530281496403 |
| 13 | 13 | 1.2252530281496403 |
| 14 | 14 | 1.2252530281496403 |
| 15 | 15 | 1.2252530281496403 |
| 16 | 16 | 1.2252530281496403 |
| 17 | 17 | 1.2252530281496403 |
| 18 | 18 | 1.2252530281496403 |
| 19 | 19 | 1.2252530281496403 |
| 20 | 20 | 1.2252530281496403 |

STATISTICS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | MAPE | 0.8959816241333334 |

METRICS:

| | NAME | VALUE |
|---|---|---|
| 1 | DEMAND_FORECAST | 3.9899999999999998 |
| 2 | PROBABILITY_FORECAST | 0.3070809594360001 |

**Example 2**: Sporadic mode

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_CROSTONTSB_DATA_TBL;
CREATE COLUMN TABLE PAL_CROSTONTSB_DATA_TBL (
    "ID" INT,
    "RAWDATA" DOUBLE
);
```

```
INSERT INTO PAL_CROSTONTSB_DATA_TBL VALUES (0, 0.0);
INSERT INTO PAL_CROSTONTSB_DATA_TBL VALUES (1, 0.0);
INSERT INTO PAL_CROSTONTSB_DATA_TBL VALUES (2, 4.0);
INSERT INTO PAL_CROSTONTSB_DATA_TBL VALUES (3, 0.0);
INSERT INTO PAL_CROSTONTSB_DATA_TBL VALUES (4, 0.0);
INSERT INTO PAL_CROSTONTSB_DATA_TBL VALUES (5, 0.0);
INSERT INTO PAL_CROSTONTSB_DATA_TBL VALUES (6, 5.0);
INSERT INTO PAL_CROSTONTSB_DATA_TBL VALUES (7, 3.0);
INSERT INTO PAL_CROSTONTSB_DATA_TBL VALUES (8, 0.0);
INSERT INTO PAL_CROSTONTSB_DATA_TBL VALUES (9, 0.0);
INSERT INTO PAL_CROSTONTSB_DATA_TBL VALUES (10, 0.0);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('ALPHA', NULL,0.1,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('BETA', NULL,0.1,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FORECAST_NUM',10, NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('METHOD',0, NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES('MEASURE_NAME',NULL,NULL,'MAPE');
INSERT INTO #PAL_PARAMETER_TBL VALUES('EXPOST_FLAG',1,NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES('REMOVE_LEADING_ZEROS',0,NULL,NULL);
CALL "_SYS_AFL"."PAL_CROSTONTSB"(PAL_CROSTONTSB_DATA_TBL,
#PAL_PARAMETER_TBL, ?, ?, ?);
```

**Expected Result 2**: Sporadic mode

RESULT:

| | ID | OUTPUT_VALUE |
|---|---|---|
| 1 | 1 | 0 |
| 2 | 2 | 0 |
| 3 | 3 | 0 |
| 4 | 4 | 0 |
| 5 | 5 | 0 |
| 6 | 6 | 4 |
| 7 | 7 | 4.1 |
| 8 | 8 | 0 |
| 9 | 9 | 0 |
| 10 | 10 | 0 |
| 11 | 11 | 3.9899999999... |
| 12 | 12 | 0 |
| 13 | 13 | 0 |
| 14 | 14 | 3.9899999999... |
| 15 | 15 | 0 |
| 16 | 16 | 0 |
| 17 | 17 | 0 |
| 18 | 18 | 3.9899999999... |
| 19 | 19 | 0 |
| 20 | 20 | 0 |

STATISTICS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | MAPE | 0.15666666666666665 |

METRICS:

| | NAME | VALUE |
|---|---|---|
| 1 | DEMAND_FORECAST | 3.9899999999999998 |
| 2 | PROBABILITY_FORECAST | 0.3070809594360001 |

## _SYS_AFL.PAL_MASSIVE_CROSTONTSB

This is a Massive Croston TSB method function.

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA Table>` | IN |
| 2 | `<PARAMETER Table>` | IN |
| 3 | `<RESULT Table>` | OUT |
| 4 | `<STATISTICS Table>` | OUT |
| 5 | `<METRICS Table>` | OUT |
| 6 | `<ERRORMSG Table>` | OUT |

### Input Tables

| Table | Column | Data Type | Description | Reference for Output Table |
|---|---|---|---|---|
| DATA | 1st column | INTEGER, VRACHAR or NVARCHAR | Group ID | |
| | 2nd column | INTEGER | ID of data | TS_ID |
| | 3rd column | INTEGER, DOUBLE, DECIMAL (p,s) | Raw data | |
| PARAMETER | 1st column | INTEGER, VRACHAR or NVARCHAR | Group ID | |
| | 2nd column | VARCHAR or NVARCHAR | Parameter name | |
| | 3rd column | INTEGER | Int value | |
| | 4th column | DOUBLE | Double value | |
| | 5th column | VARCHAR or NVARCHAR | String value | |

### Parameter Table

See Parameter Table in _SYS_AFL.PAL_CROSTONTSB.

### Output Tables

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| RESULT | 1st column | NVARCHAR | GROUP_ID | Group ID |
| | 2nd column | TS_ID (in input table) data type | TS_ID (in input table) column name | ID of data |
| | 3rd column | DOUBLE | OUTPUT_VALUE | Output result |
| STATISTICS | 1st column | NVARCHAR | GROUP_ID | Group ID |
| | 2nd column | NVARCHAR | STAT_NAME | Name of statistics |
| | 3rd column | DOUBLE | STAT_VALUE | Value of statistics |
| METRICS | 1st column | NVARCHAR | GROUP_ID | Group ID |

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| | 2nd column | NVARCHAR | NAME | Name of metrics (“DEMAND_FORECAST” or “PROBABILITY_FORECAST”) |
| | 3rd column | DOUBLE | VALUE | Value of metrics |
| ERRORMSG | 1st column | NVARCHAR | GROUP_ID | Group ID |
| | 2nd column | NVARCHAR | ERROR_TIMESTAMP | Error ID |
| | 3rd column | INTEGER | ERRORCODE | Error code |
| | 4th column | NVARCHAR | MESSAGE | Error message |

### Examples

Assume that:

- DM_PAL is a schema belonging to USER1;
- USER1 has been assigned the AFL__SYS_AFL_AFLPAL_EXECUTE or AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION role.

**Example 1**: Two groups use different parameter tables.

```
SET SCHEMA DM_PAL;

DROP TABLE PAL_MASSIVE_CROSTONTSB_TBL;
CREATE COLUMN TABLE PAL_MASSIVE_CROSTONTSB_TBL (
    "GROUPID" VARCHAR(20),
    "ID" INT,
    "ts" DOUBLE
);

DROP TABLE PAL_MASSIVE_CROSTONTSB_CONTROL_TBL;
CREATE COLUMN TABLE PAL_MASSIVE_CROSTONTSB_CONTROL_TBL (
    "GROUPID" VARCHAR(100),
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO PAL_MASSIVE_CROSTONTSB_TBL VALUES ('100',1, 3);
INSERT INTO PAL_MASSIVE_CROSTONTSB_TBL VALUES ('100',2, 6);
INSERT INTO PAL_MASSIVE_CROSTONTSB_TBL VALUES ('100',3, 4);
INSERT INTO PAL_MASSIVE_CROSTONTSB_TBL VALUES ('100',4, 0);
INSERT INTO PAL_MASSIVE_CROSTONTSB_TBL VALUES ('100',5, 0);
INSERT INTO PAL_MASSIVE_CROSTONTSB_TBL VALUES ('100',6, 0);
INSERT INTO PAL_MASSIVE_CROSTONTSB_TBL VALUES ('100',7, 0);
INSERT INTO PAL_MASSIVE_CROSTONTSB_TBL VALUES ('100',8, 0);
INSERT INTO PAL_MASSIVE_CROSTONTSB_TBL VALUES ('100',9, 0);
INSERT INTO PAL_MASSIVE_CROSTONTSB_TBL VALUES ('100',10, 16);
INSERT INTO PAL_MASSIVE_CROSTONTSB_TBL VALUES ('200',1, 5);
INSERT INTO PAL_MASSIVE_CROSTONTSB_TBL VALUES ('200',2, 0);
INSERT INTO PAL_MASSIVE_CROSTONTSB_TBL VALUES ('200',3, 0);
INSERT INTO PAL_MASSIVE_CROSTONTSB_TBL VALUES ('200',4, 3);
INSERT INTO PAL_MASSIVE_CROSTONTSB_TBL VALUES ('200',5, 0);
INSERT INTO PAL_MASSIVE_CROSTONTSB_TBL VALUES ('200',6, 0);
INSERT INTO PAL_MASSIVE_CROSTONTSB_TBL VALUES ('200',7, 0);
INSERT INTO PAL_MASSIVE_CROSTONTSB_TBL VALUES ('200',8, 0);
INSERT INTO PAL_MASSIVE_CROSTONTSB_TBL VALUES ('200',9, 8);
INSERT INTO PAL_MASSIVE_CROSTONTSB_TBL VALUES ('200',10, 0);
```

```
INSERT INTO PAL_MASSIVE_CROSTONTSB_CONTROL_TBL VALUES ('100','METHOD', 0, NULL,
NULL);
INSERT INTO PAL_MASSIVE_CROSTONTSB_CONTROL_TBL VALUES ('100','FORECAST_NUM', 10,
NULL, NULL);
INSERT INTO PAL_MASSIVE_CROSTONTSB_CONTROL_TBL VALUES ('100','ALPHA', NULL, 0.3,
NULL);
INSERT INTO PAL_MASSIVE_CROSTONTSB_CONTROL_TBL VALUES ('100','MEASURE_NAME',
NULL, NULL, 'MAPE');
INSERT INTO PAL_MASSIVE_CROSTONTSB_CONTROL_TBL VALUES ('200','METHOD', 1, NULL,
NULL);
INSERT INTO PAL_MASSIVE_CROSTONTSB_CONTROL_TBL VALUES ('200','FORECAST_NUM', 10,
NULL, NULL);
INSERT INTO PAL_MASSIVE_CROSTONTSB_CONTROL_TBL VALUES ('200','ALPHA', NULL, 0.5,
NULL);
INSERT INTO PAL_MASSIVE_CROSTONTSB_CONTROL_TBL VALUES ('200','MEASURE_NAME',
NULL, NULL, 'MAPE');
CALL "_SYS_AFL"."PAL_MASSIVE_CROSTONTSB"(PAL_MASSIVE_CROSTONTSB_TBL,
PAL_MASSIVE_CROSTONTSB_CONTROL_TBL, ?, ?, ?, ?);
```

**Expected Result 1**: Two groups use different parameter tables.

RESULT:

| | GROUP_ID | ID | OUTPUT_VALUE |
|---|---|---|---|
| 1 | 100 | 2 | 5.975 |
| 2 | 100 | 3 | 5.9825 |
| 3 | 100 | 4 | 0 |
| 4 | 100 | 5 | 0 |
| 5 | 100 | 6 | 0 |
| 6 | 100 | 7 | 0 |
| 7 | 100 | 8 | 0 |
| 8 | 100 | 9 | 0 |
| 9 | 100 | 10 | 5.38775 |
| 10 | 100 | 11 | 0 |
| 11 | 100 | 12 | 0 |
| 12 | 100 | 13 | 8.571425 |
| 13 | 100 | 14 | 0 |
| 14 | 100 | 15 | 0 |
| 15 | 100 | 16 | 8.571425 |
| 16 | 100 | 17 | 0 |
| 17 | 100 | 18 | 0 |
| 18 | 100 | 19 | 8.571425 |
| 19 | 100 | 20 | 0 |
| 20 | 200 | 2 | 1.6791666666... |
| 21 | 200 | 3 | 1.51125 |
| 22 | 200 | 4 | 1.360125 |
| 23 | 200 | 5 | 1.3757770833... |
| 24 | 200 | 6 | 1.238199375 |
| 25 | 200 | 7 | 1.1143794375 |
| 26 | 200 | 8 | 1.00294149375 |
| 27 | 200 | 9 | 0.9026473443... |
| 28 | 200 | 10 | 1.8061613446... |
| 29 | 200 | 11 | 1.6255452101... |
| 30 | 200 | 12 | 1.6255452101... |
| 31 | 200 | 13 | 1.6255452101... |
| 32 | 200 | 14 | 1.6255452101... |
| 33 | 200 | 15 | 1.6255452101... |
| 34 | 200 | 16 | 1.6255452101... |
| 35 | 200 | 17 | 1.6255452101... |
| 36 | 200 | 18 | 1.6255452101... |
| 37 | 200 | 19 | 1.6255452101... |
| 38 | 200 | 20 | 1.6255452101... |

STATISTICS:

| | GROUP_ID | STAT_NAME | STAT_VALUE |
|---|---|---|---|
| 1 | 100 | MAPE | 0.12922858796296297 |
| 2 | 200 | MAPE | 0.9370882313281251 |

METRICS:

| | GROUP_ID | NAME | VALUE |
|---|---|---|---|
| 1 | 100 | DEMAND_FORECAST | 8.571425 |
| 2 | 100 | PROBABILITY_FORECAST | 0.3458446066000001 |
| 3 | 200 | DEMAND_FORECAST | 6.041666666666666 |
| 4 | 200 | PROBABILITY_FORECAST | 0.2690557589250001 |

ERRORMSG:

| | GROUP_ID | ERROR_TIMESTAMP | ERRORCODE | MESSAGE |
|---|---|---|---|---|
| | | | | |
| | | | | |
| | | | | |

**Example 2**: Two groups share the same parameter table.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_MASSIVE_CROSTONTSB_TBL;
CREATE COLUMN TABLE PAL_MASSIVE_CROSTONTSB_TBL (
    "GROUPID" VARCHAR(20),
    "ID" INT,
    "ts" DOUBLE
);
DROP TABLE PAL_MASSIVE_CROSTONTSB_CONTROL_TBL;
CREATE COLUMN TABLE PAL_MASSIVE_CROSTONTSB_CONTROL_TBL (
    "GROUPID" VARCHAR(100),
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO PAL_MASSIVE_CROSTONTSB_TBL VALUES ('100',1, 3);
INSERT INTO PAL_MASSIVE_CROSTONTSB_TBL VALUES ('100',2, 6);
INSERT INTO PAL_MASSIVE_CROSTONTSB_TBL VALUES ('100',3, 4);
INSERT INTO PAL_MASSIVE_CROSTONTSB_TBL VALUES ('100',4, 0);
INSERT INTO PAL_MASSIVE_CROSTONTSB_TBL VALUES ('100',5, 0);
INSERT INTO PAL_MASSIVE_CROSTONTSB_TBL VALUES ('100',6, 0);
INSERT INTO PAL_MASSIVE_CROSTONTSB_TBL VALUES ('100',7, 0);
INSERT INTO PAL_MASSIVE_CROSTONTSB_TBL VALUES ('100',8, 0);
INSERT INTO PAL_MASSIVE_CROSTONTSB_TBL VALUES ('100',9, 0);
INSERT INTO PAL_MASSIVE_CROSTONTSB_TBL VALUES ('100',10, 16);
INSERT INTO PAL_MASSIVE_CROSTONTSB_TBL VALUES ('200',1, 5);
INSERT INTO PAL_MASSIVE_CROSTONTSB_TBL VALUES ('200',2, 0);
INSERT INTO PAL_MASSIVE_CROSTONTSB_TBL VALUES ('200',3, 0);
INSERT INTO PAL_MASSIVE_CROSTONTSB_TBL VALUES ('200',4, 3);
INSERT INTO PAL_MASSIVE_CROSTONTSB_TBL VALUES ('200',5, 0);
INSERT INTO PAL_MASSIVE_CROSTONTSB_TBL VALUES ('200',6, 0);
INSERT INTO PAL_MASSIVE_CROSTONTSB_TBL VALUES ('200',7, 0);
INSERT INTO PAL_MASSIVE_CROSTONTSB_TBL VALUES ('200',8, 0);
```

```
INSERT INTO PAL_MASSIVE_CROSTONTSB_TBL VALUES ('200',9, 8);
INSERT INTO PAL_MASSIVE_CROSTONTSB_TBL VALUES ('200',10, 0);
INSERT INTO PAL_MASSIVE_CROSTONTSB_CONTROL_TBL VALUES
('PAL_MASSIVE_PROCESSING_SPECIAL_GROUP_ID','METHOD', 0, NULL, NULL);
INSERT INTO PAL_MASSIVE_CROSTONTSB_CONTROL_TBL VALUES
('PAL_MASSIVE_PROCESSING_SPECIAL_GROUP_ID','FORECAST_NUM', 10, NULL, NULL);
INSERT INTO PAL_MASSIVE_CROSTONTSB_CONTROL_TBL VALUES
('PAL_MASSIVE_PROCESSING_SPECIAL_GROUP_ID','ALPHA', NULL, 0.3, NULL);
INSERT INTO PAL_MASSIVE_CROSTONTSB_CONTROL_TBL VALUES
('PAL_MASSIVE_PROCESSING_SPECIAL_GROUP_ID','MEASURE_NAME', NULL, NULL, 'MAPE');
CALL "_SYS_AFL"."PAL_MASSIVE_CROSTONTSB"(PAL_MASSIVE_CROSTONTSB_TBL,
PAL_MASSIVE_CROSTONTSB_CONTROL_TBL, ?, ?, ?, ?);
```

**Expected Result 2**: Two groups share the same parameter table.

RESULT:

| | GROUP_ID | ID | OUTPUT_VALUE |
|---|---|---|---|
| 1 | 100 | 2 | 5.975 |
| 2 | 100 | 3 | 5.9825 |
| 3 | 100 | 4 | 0 |
| 4 | 100 | 5 | 0 |
| 5 | 100 | 6 | 0 |
| 6 | 100 | 7 | 0 |
| 7 | 100 | 8 | 0 |
| 8 | 100 | 9 | 0 |
| 9 | 100 | 10 | 5.38775 |
| 10 | 100 | 11 | 0 |
| 11 | 100 | 12 | 0 |
| 12 | 100 | 13 | 8.571425 |
| 13 | 100 | 14 | 0 |
| 14 | 100 | 15 | 0 |
| 15 | 100 | 16 | 8.571425 |
| 16 | 100 | 17 | 0 |
| 17 | 100 | 18 | 0 |
| 18 | 100 | 19 | 8.571425 |
| 19 | 100 | 20 | 0 |
| 20 | 200 | 2 | 0 |
| 21 | 200 | 3 | 0 |
| 22 | 200 | 4 | 5.2333333333... |
| 23 | 200 | 5 | 0 |
| 24 | 200 | 6 | 0 |
| 25 | 200 | 7 | 0 |
| 26 | 200 | 8 | 0 |
| 27 | 200 | 9 | 4.5633333333... |
| 28 | 200 | 10 | 0 |
| 29 | 200 | 11 | 0 |
| 30 | 200 | 12 | 0 |
| 31 | 200 | 13 | 5.5943333333... |
| 32 | 200 | 14 | 0 |
| 33 | 200 | 15 | 0 |
| 34 | 200 | 16 | 5.5943333333... |
| 35 | 200 | 17 | 0 |
| 36 | 200 | 18 | 0 |
| 37 | 200 | 19 | 0 |
| 38 | 200 | 20 | 5.5943333333... |

STATISTICS:

| | GROUP_ID | STAT_NAME | STAT_VALUE |
|---|---|---|---|
| 1 | 100 | MAPE | 0.12922858796296297 |
| 2 | 200 | MAPE | 0.13044753086419755 |

METRICS:

| | GROUP_ID | NAME | VALUE |
|---|---|---|---|
| 1 | 100 | DEMAND_FORECAST | 8.571425 |
| 2 | 100 | PROBABILITY_FORECAST | 0.3458446066000001 |
| 3 | 200 | DEMAND_FORECAST | 5.594333333333333 |
| 4 | 200 | PROBABILITY_FORECAST | 0.2690557589250001 |

ERRORMSG:

| | GROUP_ID | ERROR_TIMESTAMP | ERRORCODE | MESSAGE |
|---|---|---|---|---|
| | | | | |
| | | | | |

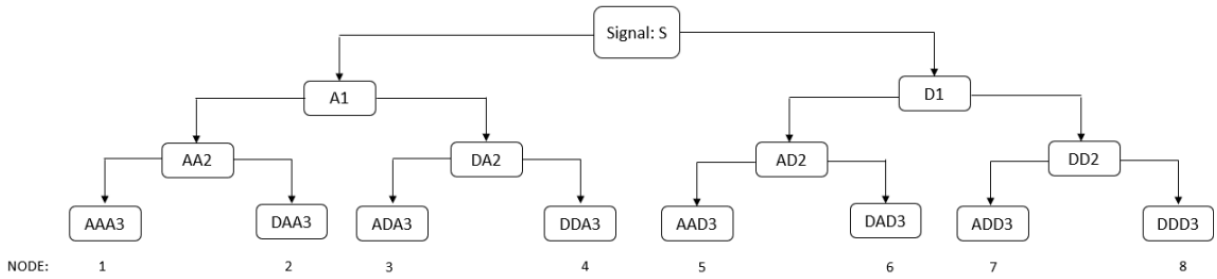
## Related Information

Croston's Method [page 702]
Forecast Accuracy Measures [page 762]

# 5.5.11 Discrete Wavelet Packet Transform

Discrete wavelet packet transform (DWPT) decomposes both the approximation part and the detail part instead of only decomposing the approximation part that is used in discrete wavelet transform (DWT). DWPT performs the transform step to both the low pass and the high pass result. The coefficients sets at the subsequent level are obtained by recursively applying the discrete wavelet transform step to the approximation results (low-pass filter results) and detail results (high-pass filter results) at the prior level.

The diagram for wavelet packet transform is shown as follows. In this figure, "A" means approximation, "D" means detail, and the digit represents the level of decomposition.




# _SYS_AFL.PAL_WAVELET_PACKET_TRANSFORM

## Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<RESULT table>` | OUT |
| 3 | `<STATISTIC table>` | OUT |

## Input Table

| Table | Column | Data Type | Description | Constraint |
|---|---|---|---|---|
| DATA | 1st Column | INTEGER | Time stamp. Time stamp does not need to be in order, but must be equal sampling | |
| | 2nd Column | DOUBLE, or DECIMAL(p,s) | Signal values | |

## Parameter Table

Mandatory Parameters

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| FILTER_TYPE | VARCHAR | No default value | The filter used for the discrete wavelet transform.<br>• db family: 'db1'–'db20'<br>• bior family: 'bior1.1', 'bior1.3', 'bior1.5', 'bior2.2', 'bior2.4', 'bior2.6', 'bior2.8', 'bior3.1', 'bior3.3', 'bior3.5', 'bior3.7', 'bior3.9', 'bior4.4', 'bior5.5', 'bior6.8'<br>• rbio family: 'rbio1.1', 'rbio1.3', 'rbio1.5', 'rbio2.2', 'rbio2.4', 'rbio2.6', 'rbio2.8', 'rbio3.1', 'rbio3.3', 'rbio3.5', 'rbio3.7', 'rbio3.9', 'rbio4.4', 'rbio5.5', 'rbio6.8'<br>• coif family: 'coif1'~'coif5'<br>• sym family: 'sym2'~'sym20' | |

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| PADDING_TYPE | INTEGER | 0 | • 0: zero padding<br>• 1: symmetric padding<br>• 2: periodic padding<br>• 3: reflect padding<br>• 4: smooth padding<br>• 5: constant padding | |
| LEVEL | INTEGER | 1 | the decompose level for the wavelet transform. | |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| OUT_TYPE | INTEGER | 0 | Control the node output order in the result table.<br>• 0: ordered by the node index.<br>• 1: ordered by the frequency. | |
| COMPRESSION | INTEGER | 0 | • 0: no compression<br>• 1: compression: all zero values are discarded. | |
| THRESHOLD_METHOD | INTEGER | 0 | • 0: no thresholding<br>• 1: soft thresholding<br>• 2: hard thresholding | |
| THRESHOLD | DOUBLE | 1e-9 | the value for results thresholding. | |

| Table | Column | Data Type | Column Name | Description | Constraint |
|---|---|---|---|---|---|
| RESULT | 1st Column | INTEGER | NODE | The index of the node | |
| | 2nd Column | INTEGER | ID | | |
| | 3rd Column | DOUBLE | COEFFICIENTS | | |
| STATISTIC | 1st Column | NVARCHAR(100) | NAME | | |
| | 2nd Column | NVARCHAR(5000) | VAL | | |

### Examples

Assume that:

- DM_PAL is a schema belonging to USER1;
- USER1 has been assigned the AFL__SYS_AFL_AFLPAL_EXECUTE or AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION role.

### Example 1: Wavelet Packet Transform

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_WPT_DATA_TBL;
CREATE COLUMN TABLE PAL_WPT_DATA_TBL (
    "ID" INTEGER,
    "VAL" DOUBLE
);
INSERT INTO PAL_WPT_DATA_TBL VALUES (1, 266.0);
INSERT INTO PAL_WPT_DATA_TBL VALUES (2, 145.9);
INSERT INTO PAL_WPT_DATA_TBL VALUES (3, 181.3);
```

```
INSERT INTO PAL_WPT_DATA_TBL VALUES (4, 119.3);
INSERT INTO PAL_WPT_DATA_TBL VALUES (5, 180.3);
INSERT INTO PAL_WPT_DATA_TBL VALUES (6, 168.5);
INSERT INTO PAL_WPT_DATA_TBL VALUES (7, 231.8);
INSERT INTO PAL_WPT_DATA_TBL VALUES (8, 224.5);
INSERT INTO PAL_WPT_DATA_TBL VALUES (9, 192.8);
INSERT INTO PAL_WPT_DATA_TBL VALUES (10, 122.9);
INSERT INTO PAL_WPT_DATA_TBL VALUES (11, 336.5);
INSERT INTO PAL_WPT_DATA_TBL VALUES (12, 185.9);
INSERT INTO PAL_WPT_DATA_TBL VALUES (13, 194.3);
INSERT INTO PAL_WPT_DATA_TBL VALUES (14, 149.5);
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR(100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR(100)
);
INSERT INTO PAL_PARAMETER_TBL VALUES ('PADDING_TYPE', 1, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('LEVEL', 2, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('FILTER_TYPE', NULL, NULL, 'db2');
INSERT INTO PAL_PARAMETER_TBL VALUES ('OUT_TYPE', 1, NULL, NULL);
DROP TABLE PAL_COEFF_TBL;
CREATE COLUMN TABLE  PAL_COEFF_TBL(
    "NODE" INTEGER,
    "ID" INTEGER,
    "COEFF" DOUBLE
);
DROP TABLE PAL_STAT_TBL;
CREATE COLUMN TABLE  PAL_STAT_TBL(
    "NAME" NVARCHAR(100),
    "VAL" NVARCHAR(5000)
);
DO BEGIN
    lt_data = SELECT * FROM PAL_WPT_DATA_TBL;
    lt_ctrl = SELECT * FROM PAL_PARAMETER_TBL;
    CALL _SYS_AFL.PAL_WAVELET_PACKET_TRANSFORM(:lt_data, :lt_ctrl, lt_coeff,
lt_stat);
    INSERT INTO PAL_COEFF_TBL SELECT * FROM :lt_coeff;
    INSERT INTO PAL_STAT_TBL SELECT * FROM :lt_stat;
END;
SELECT * FROM PAL_COEFF_TBL;
SELECT * FROM PAL_STAT_TBL;
```

### Expected Result

RESULT:

| | NODE | ID | COEFF |
|---|---|---|---|
| 1 | 0 | 0 | 451.44232843015095 |
| 2 | 0 | 1 | 405.50609060655347 |
| 3 | 0 | 2 | 350.69164447275665 |
| 4 | 0 | 3 | 412.11840593506145 |
| 5 | 0 | 4 | 362.0465171394876 |
| 6 | 1 | 0 | 35.52032910391424 |
| 7 | 1 | 1 | -53.88535808292276 |
| 8 | 1 | 2 | 71.25886175245088 |
| 9 | 1 | 3 | 78.7675135553637 |
| 10 | 1 | 4 | -70.4018328363117 |
| 11 | 3 | 0 | 28.554021922864422 |
| 12 | 3 | 1 | -11.2283177673482... |
| 13 | 3 | 2 | -57.11207423513325 |
| 14 | 3 | 3 | -16.4680033012117... |
| 15 | 3 | 4 | -19.93382383538225 |
| 16 | 2 | 0 | 87.5239787508988 |
| 17 | 2 | 1 | 59.19504270044543 |
| 18 | 2 | 2 | 16.51461709275206 |
| 19 | 2 | 3 | 131.58192594457506 |
| 20 | 2 | 4 | -27.2891395322063... |

STATISTIC:

| | NAME | VAL |
|---|---|---|
| 1 | LEVEL_COEFF_SIZE | {"coeffSize":[14,8,5]} |

**Example 2: Compression and Thresholding**

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_WPT_DATA_TBL;
CREATE COLUMN TABLE PAL_WPT_DATA_TBL (
    "ID" INTEGER,
    "VAL" DOUBLE
);
INSERT INTO PAL_WPT_DATA_TBL VALUES (1, 266.0);
INSERT INTO PAL_WPT_DATA_TBL VALUES (2, 145.9);
INSERT INTO PAL_WPT_DATA_TBL VALUES (3, 181.3);
INSERT INTO PAL_WPT_DATA_TBL VALUES (4, 119.3);
INSERT INTO PAL_WPT_DATA_TBL VALUES (5, 180.3);
INSERT INTO PAL_WPT_DATA_TBL VALUES (6, 168.5);
INSERT INTO PAL_WPT_DATA_TBL VALUES (7, 231.8);
INSERT INTO PAL_WPT_DATA_TBL VALUES (8, 224.5);
INSERT INTO PAL_WPT_DATA_TBL VALUES (9, 192.8);
INSERT INTO PAL_WPT_DATA_TBL VALUES (10, 122.9);
INSERT INTO PAL_WPT_DATA_TBL VALUES (11, 336.5);
INSERT INTO PAL_WPT_DATA_TBL VALUES (12, 185.9);
INSERT INTO PAL_WPT_DATA_TBL VALUES (13, 194.3);
INSERT INTO PAL_WPT_DATA_TBL VALUES (14, 149.5);
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR(100),
```

```
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR(100)
);
INSERT INTO PAL_PARAMETER_TBL VALUES ('PADDING_TYPE', 1, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('LEVEL', 2, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('FILTER_TYPE', NULL, NULL, 'db2');
INSERT INTO PAL_PARAMETER_TBL VALUES ('OUT_TYPE', 1, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('COMPRESSION', 1, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('THRESHOLD_METHOD', 1, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('THRESHOLD', NULL, 50.0, NULL);
DROP TABLE PAL_COEFF_TBL;
CREATE COLUMN TABLE  PAL_COEFF_TBL(
    "NODE" INTEGER,
    "ID" INTEGER,
    "COEFF" DOUBLE
);
DROP TABLE PAL_STAT_TBL;
CREATE COLUMN TABLE  PAL_STAT_TBL(
    "NAME" NVARCHAR(100),
    "VAL" NVARCHAR(5000)
);
DO BEGIN
    lt_data = SELECT * FROM PAL_WPT_DATA_TBL;
    lt_ctrl = SELECT * FROM PAL_PARAMETER_TBL;
    CALL _SYS_AFL.PAL_WAVELET_PACKET_TRANSFORM(:lt_data, :lt_ctrl, lt_coeff,
lt_stat);
    INSERT INTO PAL_COEFF_TBL SELECT * FROM :lt_coeff;
    INSERT INTO PAL_STAT_TBL SELECT * FROM :lt_stat;
END;
SELECT * FROM PAL_COEFF_TBL;
SELECT * FROM PAL_STAT_TBL;
```

### Expected Result

RESULT:

| | NODE | ID | COEFF |
|---|---|---|---|
| 1 | 0 | 0 | 401.44232843015095 |
| 2 | 0 | 1 | 355.50609060655347 |
| 3 | 0 | 2 | 300.69164447275665 |
| 4 | 0 | 3 | 362.11840593506145 |
| 5 | 0 | 4 | 312.0465171394876 |
| 6 | 1 | 1 | -3.88535808292275... |
| 7 | 1 | 2 | 21.258861752450883 |
| 8 | 1 | 3 | 28.767513555363706 |
| 9 | 1 | 4 | -20.4018328363116... |
| 10 | 3 | 2 | -7.11207423513324... |
| 11 | 2 | 0 | 37.523978750898806 |
| 12 | 2 | 1 | 9.195042700445427 |
| 13 | 2 | 3 | 81.58192594457506 |

## _SYS_AFL.PAL_WAVELET_PACKET_TRANSFORM_INVERSE

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<STATISTIC table>` | IN |
| 3 | `<PARAMETER table>` | IN |
| 4 | `<RESULT table>` | OUT |

### Input Table

| Table | Column | Data Type | Description | Constraint |
|---|---|---|---|---|
| DATA | 1st Column | INTEGER | the node index. | |
| | 2nd Column | INTEGER | the index in the node | |
| | 3rd Column | DOUBLE, or DECIMAL(p,s) | the coefficient. | |

### Statistic Table

| Table | Column | Data Type | Description | Constraint |
|---|---|---|---|---|
| STATISTIC | 1st Column | NVARCHAR(100) | name of statistics | |
| | 2nd Column | NVARCHAR(5000) | values | |

### Parameter Table

Mandatory Parameters

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| FILTER_TYPE | VARCHAR | No default value | The filter used for the discrete wavelet transform.<br>• db family: 'db1'<br>• bior family: 'bior1.1', 'bior1.3', 'bior1.5', 'bior2.2', 'bior2.4', 'bior2.6', 'bior2.8', 'bior3.1', 'bior3.3', 'bior3.5', 'bior3.7', 'bior3.9', 'bior4.4', 'bior5.5', 'bior6.8'<br>• rbio family: 'rbio1.1', 'rbio1.3', 'rbio1.5', 'rbio2.2', 'rbio2.4', 'rbio2.6', 'rbio2.8', 'rbio3.1', 'rbio3.3', 'rbio3.5', 'rbio3.7', 'rbio3.9', 'rbio4.4', 'rbio5.5', 'rbio6.8'<br>• coif family: 'coif1'~'coif5'<br>• sym family: 'sym2'~'sym20' | |

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| PADDING_TYPE | INTEGER | 0 | • 0: zero padding<br>• 1: symmetric padding<br>• 2: periodic padding<br>• 3: reflect padding<br>• 4: smooth padding<br>• 5: constant padding | |

### Output Table

| Table | Column | Data Type | Column Name | Description | Constraint |
|---|---|---|---|---|---|
| RESULT | 1st Column | INTEGER | ID | | |

| Table | Column | Data Type | Column Name | Description | Constraint |
|---|---|---|---|---|---|
| | 2nd Column | DOUBLE | VAL | The reconstruction of signal from given coefficients. | |

### Examples

Assume that:

- DM_PAL is a schema belonging to USER1;
- USER1 has been assigned the AFL__SYS_AFL_AFLPAL_EXECUTE or AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION role.

### Example: Wavelet Reconstruction

```
SET SCHEMA DM_PAL;
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR(100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR(100)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PADDING_TYPE', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FILTER_TYPE', NULL, NULL, 'db2');
CALL _SYS_AFL.PAL_WAVELET_PACKET_TRANSFORM_INVERSE(PAL_COEFF_TBL, PAL_STAT_TBL,
"#PAL_PARAMETER_TBL", ?);
```

### Expected Result

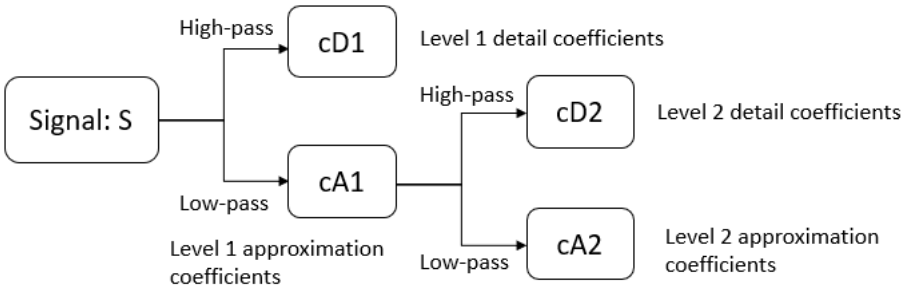
RESULT:

| | ID | VALUE |
|---|---|---|
| 1 | 1 | 266 |
| 2 | 2 | 145.9 |
| 3 | 3 | 181.3 |
| 4 | 4 | 119.30000000000001 |
| 5 | 5 | 180.30000000000007 |
| 6 | 6 | 168.50000000000009 |
| 7 | 7 | 231.80000000000013 |
| 8 | 8 | 224.50000000000009 |
| 9 | 9 | 192.80000000000007 |
| 10 | 10 | 122.90000000000006 |
| 11 | 11 | 336.5000000000001 |
| 12 | 12 | 185.90000000000006 |
| 13 | 13 | 194.3 |
| 14 | 14 | 149.49999999999997 |

## 5.5.12 Discrete Wavelet Transform

Discrete wavelet transform (DWT) was introduced in the late 1980s. DWT transform decomposes a given signal into a number of sets that contain an approximation coefficients set (cA) and multiple detail coefficient sets (cD). These cofficients sets are obtained by convolving the signal with the low-pass filter and the high-pass filter over time.

In PAL implementation, single-level 1-D discrete wavelet transform is supported.
The following diagram shows the flow to decompose a signal in DWT.




## _SYS_AFL.PAL_DISCRETE_WAVELET_TRANSFORM

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<RESULT table>` | OUT |

### Input Table

| Table | Column | Data Type | Description | Constraint |
|---|---|---|---|---|
| DATA | 1st Column | INTEGER | Time stamp. Time stamp does not need to be in order, but must be equal sampling | |
| | 2nd Column | DOUBLE, or DECIMAL(p,s) | Singal values | |

### Parameter Table

Mandatory Parameters

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| FILTER_TYPE | VARCHAR | No defeult value | The filter used for the discrete wavelet transform.<br>• db family: 'db1'<br>• bior family: 'bior1.1', 'bior1.3', 'bior1.5', 'bior2.2', 'bior2.4', 'bior2.6', 'bior2.8', 'bior3.1', 'bior3.3', 'bior3.5', 'bior3.7', 'bior3.9', 'bior4.4', 'bior5.5', 'bior6.8'<br>• rbio family: 'rbio1.1', 'rbio1.3', 'rbio1.5', 'rbio2.2', 'rbio2.4', 'rbio2.6', 'rbio2.8', 'rbio3.1', 'rbio3.3', 'rbio3.5', 'rbio3.7', 'rbio3.9', 'rbio4.4', 'rbio5.5', 'rbio6.8'<br>• coif family: 'coif1'~'coif5'<br>• sym family: 'sym2'~'sym20' | |

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| PADDING_TYPE | INTEGER | 0 | • 0: zero padding<br>• 1: symmetric padding<br>• 2: periodic padding<br>• 3: reflect padding<br>• 4: smooth padding<br>• 5: constant padding | |
| LEVEL | INTEGER | 1 | the decompose level for the wavelet transform. | |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| COMPRESSION | INTEGER | 0 | • 0: no compression<br>• 1: compression: all zero values are discarded. If values in a level are all zeros, the last zero would be kept. | |
| THRESHOLD_METHOD | INTEGER | 0 | • 0: no thresholding<br>• 1: soft thresholding<br>• 2: hard thresholding | |
| THRESHOLD | DOUBLE | 1e-9 | the value for results thresholding. | |
| LEVEL_THRESHOLDS | VARCHAR | No default value | A string to specify the level specific threshold that will be processed. The pattern is for example "X1:Y1\|X2:Y2\|.....\|Xn:Yn", where Xi is the level, Yi is the threshold for the level. Only define the level that needs thresholding. | If both THRESHOLD and LEVEL_THRESHOLDS are specified, LEVEL_THRESHOLDS takes precedence. |

### Output Table

| Table | Column | Data Type | Column Name | Description | Constraint |
|---|---|---|---|---|---|
| RESULT | 1st Column | INTEGER | LEVEL | The approximation coefficients are marked as level 0 and the detail coefficients are marked by the respective level. | |
| | 2nd Column | INTEGER | ID | | |
| | 3rd Column | DOUBLE | COEFFICIENTS | | |

Examples

Assume that:

- DM_PAL is a schema belonging to USER1;
- USER1 has been assigned the AFL__SYS_AFL_AFLPAL_EXECUTE or AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION role.

### Example 1: Discrete Wavelet Transform

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_DWT_DATA_TBL;
CREATE COLUMN TABLE PAL_DWT_DATA_TBL (
    "ID" INTEGER,
    "VAL" DOUBLE
);
INSERT INTO PAL_DWT_DATA_TBL VALUES (1, 266.0);
INSERT INTO PAL_DWT_DATA_TBL VALUES (2, 145.9);
INSERT INTO PAL_DWT_DATA_TBL VALUES (3, 181.3);
INSERT INTO PAL_DWT_DATA_TBL VALUES (4, 119.3);
INSERT INTO PAL_DWT_DATA_TBL VALUES (5, 180.3);
INSERT INTO PAL_DWT_DATA_TBL VALUES (6, 168.5);
INSERT INTO PAL_DWT_DATA_TBL VALUES (7, 231.8);
INSERT INTO PAL_DWT_DATA_TBL VALUES (8, 224.5);
INSERT INTO PAL_DWT_DATA_TBL VALUES (9, 192.8);
INSERT INTO PAL_DWT_DATA_TBL VALUES (10, 122.9);
INSERT INTO PAL_DWT_DATA_TBL VALUES (11, 336.5);
INSERT INTO PAL_DWT_DATA_TBL VALUES (12, 185.9);
INSERT INTO PAL_DWT_DATA_TBL VALUES (13, 194.3);
INSERT INTO PAL_DWT_DATA_TBL VALUES (14, 149.5);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR(100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR(100)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PADDING_TYPE', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('LEVEL', 2, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FILTER_TYPE', NULL, NULL, 'db2');
CALL _SYS_AFL.PAL_DISCRETE_WAVELET_TRANSFORM(PAL_DWT_DATA_TBL,
"#PAL_PARAMETER_TBL", ?);
```

### Expected Result

RESULT:

| | LEVEL | ID | COEFFICIENTS |
|---|---|---|---|
| 1 | 0 | 0 | 451.44232843015095 |
| 2 | 0 | 1 | 405.50609060655347 |
| 3 | 0 | 2 | 350.69164447275665 |
| 4 | 0 | 3 | 412.11840593506145 |
| 5 | 0 | 4 | 362.0465171394876 |
| 6 | 1 | 0 | 73.5459295270649 |
| 7 | 1 | 1 | 26.917406983556653 |
| 8 | 1 | 2 | 19.24232880311706 |
| 9 | 1 | 3 | 24.37852652440448 |
| 10 | 1 | 4 | 21.606775632371967 |
| 11 | 1 | 5 | 139.20749332395707 |
| 12 | 1 | 6 | 5.117512893638839 |
| 13 | 1 | 7 | -27.4342851191715... |
| 14 | 2 | 0 | 35.52032910391424 |
| 15 | 2 | 1 | -53.88535808292276 |
| 16 | 2 | 2 | 71.25886175245088 |
| 17 | 2 | 3 | 78.7675135553637 |
| 18 | 2 | 4 | -70.4018328363117 |

**Example 2: Compression and Thresholding**

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_DWT_DATA_TBL;
CREATE COLUMN TABLE PAL_DWT_DATA_TBL (
    "ID" INTEGER,
    "VAL" DOUBLE
);
INSERT INTO PAL_DWT_DATA_TBL VALUES (1, 266.0);
INSERT INTO PAL_DWT_DATA_TBL VALUES (2, 145.9);
INSERT INTO PAL_DWT_DATA_TBL VALUES (3, 181.3);
INSERT INTO PAL_DWT_DATA_TBL VALUES (4, 119.3);
INSERT INTO PAL_DWT_DATA_TBL VALUES (5, 180.3);
INSERT INTO PAL_DWT_DATA_TBL VALUES (6, 168.5);
INSERT INTO PAL_DWT_DATA_TBL VALUES (7, 231.8);
INSERT INTO PAL_DWT_DATA_TBL VALUES (8, 224.5);
INSERT INTO PAL_DWT_DATA_TBL VALUES (9, 192.8);
INSERT INTO PAL_DWT_DATA_TBL VALUES (10, 122.9);
INSERT INTO PAL_DWT_DATA_TBL VALUES (11, 336.5);
INSERT INTO PAL_DWT_DATA_TBL VALUES (12, 185.9);
INSERT INTO PAL_DWT_DATA_TBL VALUES (13, 194.3);
INSERT INTO PAL_DWT_DATA_TBL VALUES (14, 149.5);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR(100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR(100)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PADDING_TYPE', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('LEVEL', 2, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FILTER_TYPE', NULL, NULL, 'db2');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('COMPRESSION', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('THRESHOLD_METHOD', 1, NULL, NULL);
```

```
INSERT INTO #PAL_PARAMETER_TBL VALUES ('LEVEL_THRESHOLDS', NULL, NULL, '0:400|
2:100');
CALL _SYS_AFL.PAL_DISCRETE_WAVELET_TRANSFORM(PAL_DWT_DATA_TBL,
"#PAL_PARAMETER_TBL", ?);
```

**Expected Result**

RESULT:

| | LEVEL | ID | COEFFICIENTS |
|---|---|---|---|
| 1 | 0 | 0 | 51.44232843015095 |
| 2 | 0 | 1 | 5.506090606553471 |
| 3 | 0 | 3 | 12.118405935061... |
| 4 | 0 | 4 | 0 |
| 5 | 1 | 0 | 73.5459295270649 |
| 6 | 1 | 1 | 26.917406983556... |
| 7 | 1 | 2 | 19.24232880311706 |
| 8 | 1 | 3 | 24.37852652440448 |
| 9 | 1 | 4 | 21.606775632371... |
| 10 | 1 | 5 | 139.20749332395... |
| 11 | 1 | 6 | 5.117512893638839 |
| 12 | 1 | 7 | -27.434285119171... |
| 13 | 2 | 4 | 0 |

## _SYS_AFL.PAL_DISCRETE_WAVELET_TRANSFORM_INVERSE

Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<RESULT table>` | OUT |

Input Table

| Table | Column | Data Type | Description | Constraint |
|---|---|---|---|---|
| DATA | 1st Column | INTEGER | the level. | |
| | 2nd Column | INTEGER | the index | |
| | 3rd Column | DOUBLE, or DECIMAL(p,s) | the coefficient. | |

Parameter Table

Mandatory Parameters

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| FILTER_TYPE | VARCHAR | No default value | The filter used for the discrete wavelet transform.<br>• db family: 'db1'<br>• bior family: 'bior1.1', 'bior1.3', 'bior1.5', 'bior2.2', 'bior2.4', 'bior2.6', 'bior2.8', 'bior3.1', 'bior3.3', 'bior3.5', 'bior3.7', 'bior3.9', 'bior4.4', 'bior5.5', 'bior6.8'<br>• rbio family: 'rbio1.1', 'rbio1.3', 'rbio1.5', 'rbio2.2', 'rbio2.4', 'rbio2.6', 'rbio2.8', 'rbio3.1', 'rbio3.3', 'rbio3.5', 'rbio3.7', 'rbio3.9', 'rbio4.4', 'rbio5.5', 'rbio6.8'<br>• coif family: 'coif1'~'coif5'<br>• sym family: 'sym2'~'sym20' | |

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| PADDING_TYPE | INTEGER | 0 | • 0: zero padding<br>• 1: symmetric padding<br>• 2: periodic padding<br>• 3: reflect padding<br>• 4: smooth padding<br>• 5: constant padding | |

Output Table

| Table | Column | Data Type | Column Name | Description | Constraint |
|---|---|---|---|---|---|
| RESULT | 1st Column | INTEGER | ID | | |

| Table | Column | Data Type | Column Name | Description | Constraint |
|---|---|---|---|---|---|
| | 2nd Column | DOUBLE | VAL | the reconstruction of signal from given coefficients. | |

Examples

Assume that:

- DM_PAL is a schema belonging to USER1;
- USER1 has been assigned the AFL__SYS_AFL_AFLPAL_EXECUTE or AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION role.

**Example: Wavelet Reconstruction**

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_IDWT_DATA_TBL;
CREATE COLUMN TABLE PAL_IDWT_DATA_TBL (
    "LEVEL" INTEGER,
    "ID" INTEGER,
    "COEFF" DOUBLE
);
INSERT INTO PAL_IDWT_DATA_TBL VALUES (0, 0, 451.44232843015095);
INSERT INTO PAL_IDWT_DATA_TBL VALUES (0, 1, 405.50609060655347);
INSERT INTO PAL_IDWT_DATA_TBL VALUES (0, 2, 350.69164447275665);
INSERT INTO PAL_IDWT_DATA_TBL VALUES (0, 3, 412.11840593506145);
INSERT INTO PAL_IDWT_DATA_TBL VALUES (0, 4, 362.0465171394876);
INSERT INTO PAL_IDWT_DATA_TBL VALUES (1, 0, 73.5459295270649);
INSERT INTO PAL_IDWT_DATA_TBL VALUES (1, 1, 26.917406983556653);
INSERT INTO PAL_IDWT_DATA_TBL VALUES (1, 2, 19.24232880311706);
INSERT INTO PAL_IDWT_DATA_TBL VALUES (1, 3, 24.37852652440448);
INSERT INTO PAL_IDWT_DATA_TBL VALUES (1, 4, 21.606775632371967);
INSERT INTO PAL_IDWT_DATA_TBL VALUES (1, 5, 139.20749332395707);
INSERT INTO PAL_IDWT_DATA_TBL VALUES (1, 6, 5.117512893638839);
INSERT INTO PAL_IDWT_DATA_TBL VALUES (1, 7, -27.434285119171594);
INSERT INTO PAL_IDWT_DATA_TBL VALUES (2, 0, 35.52032910391424 );
INSERT INTO PAL_IDWT_DATA_TBL VALUES (2, 1, -53.88535808292276);
INSERT INTO PAL_IDWT_DATA_TBL VALUES (2, 2, 71.25886175245088);
INSERT INTO PAL_IDWT_DATA_TBL VALUES (2, 3, 78.7675135553637);
INSERT INTO PAL_IDWT_DATA_TBL VALUES (2, 4, -70.4018328363117);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR(100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR(100)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PADDING_TYPE', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FILTER_TYPE', NULL, NULL, 'db2');
CALL _SYS_AFL.PAL_DISCRETE_WAVELET_TRANSFORM_INVERSE(PAL_IDWT_DATA_TBL,
"#PAL_PARAMETER_TBL", ?);
```

**Expected Result**

RESULT:

| | ID | VALUE |
|---|---|---|
| 1 | 0 | 266 |
| 2 | 1 | 145.90000000000003 |
| 3 | 2 | 181.3 |
| 4 | 3 | 119.30000000000001 |
| 5 | 4 | 180.30000000000007 |
| 6 | 5 | 168.50000000000009 |
| 7 | 6 | 231.80000000000013 |
| 8 | 7 | 224.50000000000009 |
| 9 | 8 | 192.80000000000007 |
| 10 | 9 | 122.90000000000005 |
| 11 | 10 | 336.5000000000001 |
| 12 | 11 | 185.90000000000006 |
| 13 | 12 | 194.3 |
| 14 | 13 | 149.49999999999997 |

## 5.5.13 Dynamic Time Warping

Dynamic time warping (DTW) calculates the distance or dissimilarity between two series. DTW makes one series match the other one as much as possible by stretching or compressing one or both two.

Given two series $X = (x_1, x_2, x_3, ..., x_N)$ and $Y = (y_1, y_2, y_3, \ldots, y_M)$, a match curve is defined by a series of pairs of subscripts

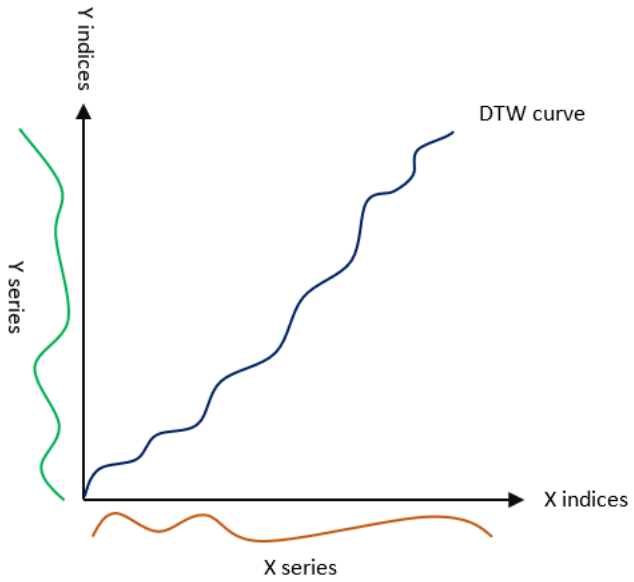
$C = \{(a_1, b_1), (a_2, b_2), (a_3, b_3), \ldots, (a_T, b_T)\}$

$a_i \in \{1, 2, 3, \ldots, N\}, b_i \in \{1, 2, 3, \ldots, M\}, i = 1, 2, 3, \ldots, T$

$a_i \geq a_{i-1}$

$b_i \geq b_{i-1}$

where $a_i$ and $b_i$ stand for subscript of X and Y respectively, and subjects to monotonicity constraint showed above. The curve C can be draw as below by treating components of C as coordinates:




Normally, we name X as query series and Y as reference series, and impose a constraint upon query series X that is $a_1 = 1, a_T = N$, which means query series X must engage fully in the match process.

Some other constraints can be specified:

1. Start-point constraint that is $b_1 = 1$. If there is no such constraint, it is called open beginning.
2. End-point constraint that is $b_t = M$. If there is no such constraint, it is called open end.
3. Window constraint that is $|a_i - b_i| \leq window, i = 1, 2, 3, \ldots, T$.

And if the match is neither open beginning nor open end, it is called closed.

The distance of such DTW curve C is defined as follows:

$$\Phi_C = \frac{\sum_{i=1}^{T} D(x_{a_i}, y_{b_i})\omega_i}{\Omega}$$

D stands for single point pair distance,such as Euclidean distance

$\omega_i$ stands for local step weight

and Ω stands for normalization coefficent,usually $\Omega = \sum_{i=1}^{T} \omega_i$

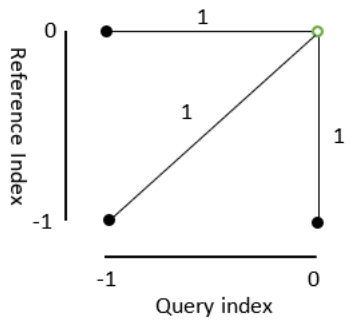
And define optimal curve $\hat{C} = \arg\min_{C} \Phi_C$

Step patterns are a set of rules regarding how one pair of matches can transit to the next. Step patterns can be represented conveniently through a set of consecutive triads, as $\{(\Delta x, \Delta y, \omega), (\Delta x, \Delta y, \omega) \ldots\}, \{(\Delta x, \Delta y, \omega), (\Delta x, \Delta y, \omega) \ldots\}, \ldots$

For example, one widely used step patterns can be written as {(1, 0, 1)}, {(1, 1, 1)}, {(0, 1, 1)}, which means there are three choices while transiting $(a_i, b_i)$ to next match on the curve. They are $(a_i+1, b_i+1)$ and $(a_{i,b_i+1}$. All of them has local step weight 1.

In our DTW algorithm, there are 5 predefined step patterns (see below) and it allows users to define their own step patterns.

1.

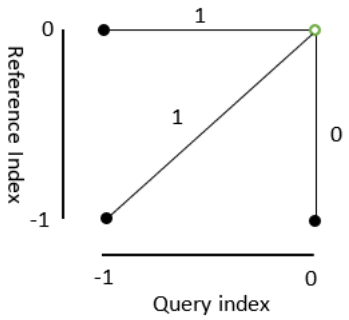



$P_1 \rightarrow (1,0,1)$

$P_2 \rightarrow (1,1,1)$

$P_3 \rightarrow (0,1,1)$

2.

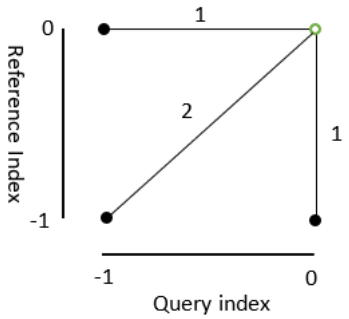



$P_1 \rightarrow (1,0,1)$

$P_2 \rightarrow (1,1,1)$

$P_3 \rightarrow (0,1,0)$

3.

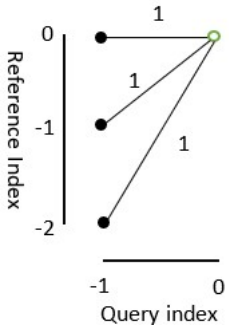



$P_1 \rightarrow (1,0,1)$

$P_2 \rightarrow (1,1,2)$

$P_3 \rightarrow (0,1,1)$

4.

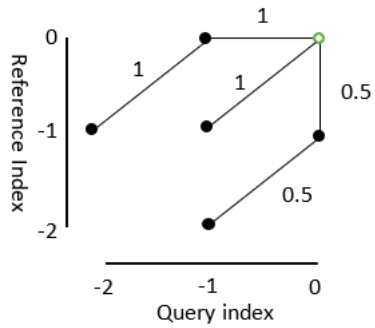



$P_1 \rightarrow (1,0,1)$

$P_2 \rightarrow (1,1,1)$

$P_3 \rightarrow (1,2,1)$

5.




$P_1 \rightarrow (1,1,1),(1,0,1)$

$P_2 \rightarrow (1,1,1)$

$P_3 \rightarrow (1,1,0.5),(0,1,0.5)$

It should be noticed that curve distance Φ_C above is normalized for one usually wishes to compare different curves by average per-step distance due to different curve length that leads to prejudice favour for short one. It follows that it is very important to choose a reasonable Ω that depends only on start and end coordinates of the whole curve. If previous property holds for some step patterns, then such step patterns are called normalizable. Normally there are 3 categories of normalizable step patterns:

1. $\Omega = x_T - x_1$, the length of matched query series X.
2. $\Omega = y_T - y_1$, the length of matched reference series Y.
3. $\Omega = (x_T - x_1) + (y_T - y_1)$, the total length of both.

In predefined step patterns listed above, type 2, 4 and 5 are category 1; type 3 is category 3; and type 1 is not normalizable as it favours diagonal. So be careful comparing DTW distance while applying step pattern type 1 and take cautions while design your own step patterns.

## Prerequisite

- The tables used to store the data is a column table and contains no null data.

## _AFL_SYS.PAL_DTW

This function calculates DTW distances between pairs of query series and reference series.

**Overview of all tables**

| Position | Table name | Parameter type |
|---|---|---|
| 1 | Query Data table | IN |
| 2 | Reference Data table | IN |
| 2 | Parameter table | IN |
| 3 | Result table | OUT |

| Position | Table name | Parameter type |
|---|---|---|
| 4 | Alignment table | OUT |
| 5 | Statistic table | OUT |

### Signature

### Input Table

| Table | Column | Column Data Type | Description | Constraint |
|---|---|---|---|---|
| Query DATA | 1st column | VARCHAR, NVARCHAR or INTEGER | ID of query series | |
| | 2nd column | VARCHAR, NVARCHAR or INTEGER | Order of query series data | When sorting by this column, it should reflect the order of series data |
| | Other columns | INTEGER, DOUBLE, or DECIMAL(p, s) | Series data | |
| Reference DATA | 1st column | VARCHAR, NVARCHAR or INTEGER | ID of reference series | |
| | 2nd column | VARCHAR, NVARCHAR or INTEGER | Order of reference series data | When sorting by this column, it should reflect the order of series data |
| | Other columns | INTEGER, DOUBLE, or DECIMAL(p, s) | Series data | Must have the same number of columns as query data table |

### Parameter Table

Mandatory Parameters

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data type | Default value | Description | Dependency |
|---|---|---|---|---|
| THREAD_RATIO | DOUBLE | -1 | The ratio of available threads.<br>• 0: single thread<br>• 0~1: percentage<br>• Others: heuristically determined | |
| WINDOW | INTEGER | -1 | A constraint to restrict match curve in an area near diagonal. To be specific, it makes sure every step within curve satisfies \|ai – bi\| <= window.<br>-1 means no such constraint, otherwise it must be nonnegative. | By setting this constraint it may get non-optimal result in exchange for runtime reduction. Be careful that inappropriate value may lead to no result at all. |
| DISTANCE_METHOD | INTEGER | 2 | Specifies the method to compute the distance between two points.<br>• 1: Manhattan<br>• 2: Euclidean<br>• 3: Minkowski<br>• 4: Chebyshev<br>• 6: Cosine | |
| MINKOWSKI_POWER | DOUBLE | 3 | Specifies the power of the Minkowski method. | Only valid when DISTANCE_METHOD is 3. |

| Name | Data type | Default value | Description | Dependency |
|---|---|---|---|---|
| BEGIN_END_ALIGNMENT | VARCHAR/NVARCHAR | CLOSED | Specifies if there is any constraint of begin/end point alignment:<br>CLOSED: beginning and end point must be aligned.<br>OPEN_END: Only beginning point needs to be aligned.<br>OPEN_BEGIN: only end point needs to be aligned.<br>OPEN_BEGIN_END: neither beginning nor end point need to be aligned. | These constraints only apply to reference series. |
| STEP_PATTERN_TYPE | INTEGER | 3 | Specifies predefined step patterns described above. The range is [1, 5]. | |
| STEP_PATTERN | VARCHAR/NVARHCAR | | Specifies custom defined step patterns.<br>There can have multiple entities each of which represent one step pattern. For example, predefined step pattern 1 equals 3 strings:<br>"(1,0,1)", "(1,1,1)" and "(0,1,1)", each of them taking one row. | STEP_PATTERN_TYPE takes precedence over STEP_PATTERN. |
| SAVE_ALIGNMENT | INTEGER | 0 | Specifies if output alignment information.<br>• 0: no<br>• 1: yes | |

## Output Tables

| Table | Column | Data type | Column name | Description |
|---|---|---|---|---|
| RESULT | 1st column | ID column type of query data table. | QUERY_<ID column name of query data table> | ID of query series |
| | 2nd column | ID column type of reference data table. | REF_<ID column name of reference data table> | ID of reference series |
| | 3rd column | DOUBLE | DISTANCE | DTW distance of the two series. NULL if there is no valid result. |
| | 4th column | DOUBLE | WEIGHT | Total weight of match |
| | 5th column | DOUBLE | AVG_DISTANCE | Normalized distance of two series. NULL if WEIGHT is near 0 |
| ALIGNMENT | 1st column | ID column type of query data table | QUERY_<ID column name of query data table> | ID of query series |
| | 2nd column | ID column type of reference data table | REF_<ID column name of reference data table> | ID of reference series |
| | 3rd column | INTEGER | QUERY_INDEX | Corresponding to index of query series data. |
| | 4th column | INTEGER | REF_INDEX | Corresponding to index of reference series data. |
| STATISTICS | 1st column | NVARCHAR(1000) | STAT_NAME | Statistics name |
| | 2nd column | NVARCHAR(1000) | STAT_VALUE | Statistics value |

### Examples

Assume that:

- `DM_PAL` is a schema belonging to USER1; and
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
set schema DM_PAL;
drop table PAL_DTW_QUERY_DATA_TAB;
create column table PAL_DTW_QUERY_DATA_TAB (
    "ID" nvarchar(100),
    "TIMESTAMP" integer,
    "ATTR1" integer,    -- only support integer or double
```

```
    "ATTR2" double
);
drop table PAL_DTW_REF_DATA_TAB;
create column table PAL_DTW_REF_DATA_TAB (
    "ID" nvarchar(100),
    "TIMESTAMP" integer,
    "ATTR1" integer,    -- must have same number of data columns as query table
    "ATTR2" double
);
drop table PAL_PARAMETER_TAB;
create column table PAL_PARAMETER_TAB (
    "PARAM_NAME" nvarchar (256),
    "INT_VALUE" integer,
    "DOUBLE_VALUE" double,
    "STRING_VALUE" nvarchar (1000)
);
drop table PAL_DTW_RES_TAB;
create column table PAL_DTW_RES_TAB (
    "LEFT_ID" nvarchar(100),
    "RIGHT_ID" nvarchar(100),
    "DISTANCE" double
);
drop table PAL_DTW_ALIGN_TAB;
create column table PAL_DTW_ALIGN_TAB (
    "LEFT_ID" nvarchar(100),
    "RIGHT_ID" nvarchar(100),
    "LEFT_INDEX" integer,
    "RIGHT_INDEX" integer
);
drop table PAL_DTW_STAT_TAB;
create column table PAL_DTW_STAT_TAB (
    "STAT_NAME" nvarchar(256),
    "STAT_VALUE" nvarchar(1000)
);
truncate table PAL_DTW_QUERY_DATA_TAB;
insert into PAL_DTW_QUERY_DATA_TAB values (1, 1, 1, 5.2);
insert into PAL_DTW_QUERY_DATA_TAB values (1, 2, 2, 5.1);
insert into PAL_DTW_QUERY_DATA_TAB values (1, 3, 3, 2.0);
insert into PAL_DTW_QUERY_DATA_TAB values (1, 4, 4, 0.3);
insert into PAL_DTW_QUERY_DATA_TAB values (1, 5, 5, 1.2);
insert into PAL_DTW_QUERY_DATA_TAB values (1, 6, 6, 7.7);
insert into PAL_DTW_QUERY_DATA_TAB values (1, 7, 7, 0.0);
insert into PAL_DTW_QUERY_DATA_TAB values (1, 8, 8, 1.1);
insert into PAL_DTW_QUERY_DATA_TAB values (1, 9, 9, 3.2);
insert into PAL_DTW_QUERY_DATA_TAB values (1, 10, 10, 2.3);
insert into PAL_DTW_QUERY_DATA_TAB values (2, 1, 7, 2.0);
insert into PAL_DTW_QUERY_DATA_TAB values (2, 2, 6, 1.4);
insert into PAL_DTW_QUERY_DATA_TAB values (2, 3, 1, 0.9);
insert into PAL_DTW_QUERY_DATA_TAB values (2, 4, 3, 1.2);
insert into PAL_DTW_QUERY_DATA_TAB values (2, 5, 2, 10.2);
insert into PAL_DTW_QUERY_DATA_TAB values (2, 6, 5, 2.3);
insert into PAL_DTW_QUERY_DATA_TAB values (2, 7, 4, 4.5);
insert into PAL_DTW_QUERY_DATA_TAB values (2, 8, 3, 4.6);
insert into PAL_DTW_QUERY_DATA_TAB values (2, 9, 3, 3.5);
truncate table PAL_DTW_REF_DATA_TAB;
insert into PAL_DTW_REF_DATA_TAB values (3, 1, 10, 1.0);
insert into PAL_DTW_REF_DATA_TAB values (3, 2, 5, 2.0);
insert into PAL_DTW_REF_DATA_TAB values (3, 3, 2, 3.0);
insert into PAL_DTW_REF_DATA_TAB values (3, 4, 8, 1.4);
insert into PAL_DTW_REF_DATA_TAB values (3, 5, 1, 10.8);
insert into PAL_DTW_REF_DATA_TAB values (3, 6, 5, 7.7);
insert into PAL_DTW_REF_DATA_TAB values (3, 7, 5, 6.3);
insert into PAL_DTW_REF_DATA_TAB values (3, 8, 12, 2.4);
insert into PAL_DTW_REF_DATA_TAB values (3, 9, 20, 9.4);
insert into PAL_DTW_REF_DATA_TAB values (3, 10, 4, 0.5);
insert into PAL_DTW_REF_DATA_TAB values (3, 11, 6, 2.2);
truncate table PAL_PARAMETER_TAB;
insert into PAL_PARAMETER_TAB values ('THREAD_RATIO', null, 1, null);
```

```
insert into PAL_PARAMETER_TAB values ('DISTANCE_METHOD', 2, null, null); -- 1:
MANHATTAN; 2: EUCLIDEAN (default); 3: MINKOWSKI; 4: CHEBYSHEV; 6: COSINE;
--insert into PAL_PARAMETER_TAB values ('MINKOWSKI_POWER', null, 0.7, null); --
only valid when DISTANCE_METHOD = 3, default 3.0
insert into PAL_PARAMETER_TAB values ('WINDOW', -1, null, null);
--insert into PAL_PARAMETER_TAB values ('STEP_PATTERN_TYPE', 5, null, null);
-- equivalent to STEP_PATTERN_TYPE = 5
insert into PAL_PARAMETER_TAB values ('STEP_PATTERN', null, null, '(1,1,1),
(1,0,1)');
insert into PAL_PARAMETER_TAB values ('STEP_PATTERN', null, null, '(1,1,1)');
insert into PAL_PARAMETER_TAB values ('STEP_PATTERN', null, null, '(1,1,0.5),
(0,1,0.5)');
--insert into PAL_PARAMETER_TAB values ('BEGIN_END_ALIGNMENT', null, null,
'CLOSED');
insert into PAL_PARAMETER_TAB values ('SAVE_ALIGNMENT', 1, null, null);
call _SYS_AFL.PAL_DTW(PAL_DTW_QUERY_DATA_TAB, PAL_DTW_REF_DATA_TAB,
PAL_PARAMETER_TAB, ?, ?, ?);
```

### Expected Result

result:

| | QUERY_ID | REF_ID | DISTANCE | WEIGHT | AVG_DISTANCE |
|---|---|---|---|---|---|
| 1 | 1 | 3 | 48.02742676633052 | 10 | 4.802742676633052 |
| 2 | 2 | 3 | 36.93392876496954 | 9 | 4.103769862774393 |

alignment:

| | QUERY_ID | REF_ID | QUERY_INDEX | REF_INDEX |
|---|---|---|---|---|
| 1 | 1 | 3 | 0 | 0 |
| 2 | 1 | 3 | 1 | 1 |
| 3 | 1 | 3 | 2 | 2 |
| 4 | 1 | 3 | 3 | 2 |
| 5 | 1 | 3 | 4 | 3 |
| 6 | 1 | 3 | 5 | 4 |
| 7 | 1 | 3 | 5 | 5 |
| 8 | 1 | 3 | 6 | 6 |
| 9 | 1 | 3 | 6 | 7 |
| 10 | 1 | 3 | 7 | 8 |
| 11 | 1 | 3 | 7 | 9 |
| 12 | 1 | 3 | 8 | 10 |
| 13 | 1 | 3 | 9 | 10 |
| 14 | 2 | 3 | 0 | 0 |
| 15 | 2 | 3 | 1 | 1 |
| 16 | 2 | 3 | 2 | 2 |
| 17 | 2 | 3 | 3 | 3 |
| 18 | 2 | 3 | 4 | 4 |
| 19 | 2 | 3 | 4 | 5 |
| 20 | 2 | 3 | 5 | 6 |
| 21 | 2 | 3 | 6 | 6 |
| 22 | 2 | 3 | 7 | 7 |
| 23 | 2 | 3 | 7 | 8 |
| 24 | 2 | 3 | 8 | 9 |
| 25 | 2 | 3 | 8 | 10 |

## 5.5.14 Fast Dynamic Time Warping

Dynamic time warping (DTW) calculates the distance or similarity between two time series. DTW stretches or compresses one or both of the two time series to make one match the other as much as possible. It also provides the optimal match between two given sequences with certain constraints and rules.

Fast DTW is a derivative version of DTW to accelerate computation when the size of time series is huge. It recursively reduces the size of time series and calculates the DTW path on reduced versions, then refines them. In exchange of computing acceleration, fast DTW may lose some accuracy of actual DTW distance.

## Prerequisites

The table used to store the data is a column partition table, and contains no null data.

## _AFL_SYS.PAL_FAST_DTW

This function calculates DTW distances between pairs of time series.

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<RESULT table>` | OUT |
| 4 | `<Alignment table>` | OUT |
| 5 | `<STATISTIC table>` | OUT |

### Signature

### Input Table

| Table | Column | Column Data Type | Description | Constraint |
|---|---|---|---|---|
| DATA | 1st column | VARCHAR, NVARCHAR or INTEGER | ID of time series | – |
| | 2nd column | VARCHAR, NVARCHAR or INTEGER | Timestamps of time series | When sorting by this column, the result reflects the order of data for time series. |
| | Other columns | INTEGER, DOUBLE, or DECIMAL(p, s) | Time series data | – |

### Parameter Table

Mandatory Parameters

| Name | Data Type | Description |
|---|---|---|
| RADIUS | INTEGER | Used for balancing the accuracy and run time of DTW. Bigger value with more accuracy and slower run time.<br>The value must be positive numbers. |

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| THREAD_RATIO | DOUBLE | -1 | Indicates the ratio of available threads:<br>• 0: single thread<br>• 0–1: percentage<br>• Others: heuristically determined | _ |
| DISTANCE_METHOD | INTEGER | 2 | Specifies the method to compute the distance between two points:<br>• 1: Manhattan<br>• 2: Euclidean<br>• 3: Minkowski<br>• 4: Chebyshev<br>• 6: Cosine | _ |
| MINKOWSKI_POWER | DOUBLE | 3 | Specifies the power of the Minkowski method. | Only valid when DISTANCE_METHOD is 3. |
| SAVE_ALIGNMENT | INTEGER | 0 | Specifies if output alignment information.<br>• 0: no<br>• 1: yes | |

## Output Tables

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| RESULT | 1st column | ID (in input table) column type | LEFT_`<ID column name of input table>` | ID of one time series |
| | 2nd column | ID (in input table) column type | RIGHT_`<ID column name of input table>` | ID of the other time series |

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| | 3rd column | DOUBLE | DISTANCE | DTW distance of the two time series |
| ALIGNMENT | 1st column | ID (in input table) column type | LEFT_<ID column name of input table> | ID of one time series |
| | 2nd column | ID (in input table) column type | RIGHT_<ID column name of input table> | ID of the other time series |
| | 3rd column | INTEGER | LEFT_INDEX | Corresponding to the timestamps index of time series with the 1st column's ID |
| | 4th column | INTEGER | RIGHT_INDEX | Corresponding to the timestamps index of time series with the 2nd column's ID |
| STATISTICS | 1st column | NVARCHAR(1000) | STAT_NAME | Statistics name |
| | 2nd column | NVARCHAR(1000) | STAT_VALUE | Statistics value |

### Examples

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
set schema DM_PAL;
drop table PAL_FAST_DTW_DATA_TAB;
create column table PAL_FAST_DTW_DATA_TAB (
    "ID" nvarchar(100),
    "TIMESTAMP" integer,
    "ATTR1" integer,
    "ATTR2" double
);
drop table PAL_PARAMETER_TAB;
create column table PAL_PARAMETER_TAB (
    "PARAM_NAME" nvarchar (256),
    "INT_VALUE" integer,
    "DOUBLE_VALUE" double,
    "STRING_VALUE" nvarchar (1000)
);
drop table PAL_FAST_DTW_RES_TAB;
create column table PAL_FAST_DTW_RES_TAB (
    "LEFT_ID" nvarchar(100),
    "RIGHT_ID" nvarchar(100),
    "DISTANCE" double
);
drop table PAL_FAST_DTW_ALIGN_TAB;
```

```
create column table PAL_FAST_DTW_ALIGN_TAB (
    "LEFT_ID" nvarchar(100),
    "RIGHT_ID" nvarchar(100),
    "LEFT_INDEX" integer,
    "RIGHT_INDEX" integer
);
drop table PAL_FAST_DTW_STAT_TAB;
create column table PAL_FAST_DTW_STAT_TAB (
    "STAT_NAME" nvarchar(256),
    "STAT_VALUE" nvarchar(1000)
);
truncate table PAL_FAST_DTW_DATA_TAB;
insert into PAL_FAST_DTW_DATA_TAB values (1, 1, 1, 5.2);
insert into PAL_FAST_DTW_DATA_TAB values (1, 2, 2, 5.1);
insert into PAL_FAST_DTW_DATA_TAB values (1, 3, 3, 2.0);
insert into PAL_FAST_DTW_DATA_TAB values (1, 4, 4, 0.3);
insert into PAL_FAST_DTW_DATA_TAB values (1, 5, 5, 1.2);
insert into PAL_FAST_DTW_DATA_TAB values (1, 6, 6, 7.7);
insert into PAL_FAST_DTW_DATA_TAB values (1, 7, 7, 0.0);
insert into PAL_FAST_DTW_DATA_TAB values (1, 8, 8, 1.1);
insert into PAL_FAST_DTW_DATA_TAB values (1, 9, 9, 3.2);
insert into PAL_FAST_DTW_DATA_TAB values (1, 10, 10, 2.3);
insert into PAL_FAST_DTW_DATA_TAB values (2, 1, 7, 2.0);
insert into PAL_FAST_DTW_DATA_TAB values (2, 2, 6, 1.4);
insert into PAL_FAST_DTW_DATA_TAB values (2, 3, 1, 0.9);
insert into PAL_FAST_DTW_DATA_TAB values (2, 4, 3, 1.2);
insert into PAL_FAST_DTW_DATA_TAB values (2, 5, 2, 10.2);
insert into PAL_FAST_DTW_DATA_TAB values (2, 6, 5, 2.3);
insert into PAL_FAST_DTW_DATA_TAB values (2, 7, 4, 4.5);
insert into PAL_FAST_DTW_DATA_TAB values (2, 8, 3, 4.6);
insert into PAL_FAST_DTW_DATA_TAB values (2, 9, 3, 3.5);
insert into PAL_FAST_DTW_DATA_TAB values (3, 1, 10, 1.0);
insert into PAL_FAST_DTW_DATA_TAB values (3, 2, 5, 2.0);
insert into PAL_FAST_DTW_DATA_TAB values (3, 3, 2, 3.0);
insert into PAL_FAST_DTW_DATA_TAB values (3, 4, 8, 1.4);
insert into PAL_FAST_DTW_DATA_TAB values (3, 5, 1, 10.8);
insert into PAL_FAST_DTW_DATA_TAB values (3, 6, 5, 7.7);
insert into PAL_FAST_DTW_DATA_TAB values (3, 7, 5, 6.3);
insert into PAL_FAST_DTW_DATA_TAB values (3, 8, 12, 2.4);
insert into PAL_FAST_DTW_DATA_TAB values (3, 9, 20, 9.4);
insert into PAL_FAST_DTW_DATA_TAB values (3, 10, 4, 0.5);
insert into PAL_FAST_DTW_DATA_TAB values (3, 11, 6, 2.2);
truncate table PAL_PARAMETER_TAB;
insert into PAL_PARAMETER_TAB values ('RADIUS', 5, null, null);
insert into PAL_PARAMETER_TAB values ('THREAD_RATIO', null, 1, null);
insert into PAL_PARAMETER_TAB values ('DISTANCE_METHOD', 2, null, null);
insert into PAL_PARAMETER_TAB values ('SAVE_ALIGNMENT', 1, null, null);
call _SYS_AFL.PAL_FAST_DTW(PAL_FAST_DTW_DATA_TAB, PAL_PARAMETER_TAB, ?, ?, ?);
```

### Expected Result

Result:

| LEFT_ID | RIGHT_ID | DISTANCE |
|---|---|---|
| 1 | 2 | 44.1547416154823866 |
| 2 | 3 | 60.4507211902561 5 |

| LEFT_ID | RIGHT_ID | DISTANCE |
|---|---|---|
| 2 | 3 | 51.9536820257581 8 |

Alignment:

| LEFT_ID | RIGHT_ID | LEFT_INDEX | RIGHT_INDEX |
|---|---|---|---|
| 1 | 2 | 0 | 0 |
| 2 | 2 | 1 | 1 |
| 3 | 2 | 2 | 2 |
| 4 | 2 | 3 | 3 |
| 5 | 2 | 4 | 3 |
| 6 | 2 | 5 | 4 |
| 7 | 2 | 6 | 5 |
| 8 | 2 | 7 | 6 |
| 9 | 2 | 8 | 7 |

| LEFT_ID | RIGHT_ID | LEFT_INDEX | RIGHT_INDEX |
|---|---|---|---|
| 10 | 2 | 9 | 8 |
| 11 | 3 | 0 | 0 |
| 12 | 3 | 1 | 1 |
| 13 | 3 | 2 | 2 |
| 14 | 3 | 3 | 2 |
| 15 | 3 | 4 | 3 |
| 16 | 3 | 5 | 4 |
| 17 | 3 | 5 | 5 |
| 18 | 3 | 5 | 6 |
| 19 | 3 | 6 | 7 |
| 20 | 3 | 7 | 8 |
| 21 | 3 | 8 | 9 |
| 22 | 3 | 9 | 10 |
| 23 | 3 | 0 | 0 |
| 24 | 3 | 1 | 1 |
| 25 | 3 | 2 | 2 |
| 26 | 3 | 3 | 3 |

| LEFT_ID | RIGHT_ID | LEFT_INDEX | RIGHT_INDEX |
|---|---|---|---|
| 27 | 3 | 4 | 4 |
| 28 | 3 | 4 | 5 |
| 29 | 3 | 5 | 6 |
| 30 | 3 | 5 | 7 |
| 31 | 3 | 6 | 8 |
| 32 | 3 | 7 | 9 |
| 33 | 3 | 8 | 10 |

## 5.5.15 Fast Fourier Transform

Fourier transform, first discussed by Joseph Fourier in 1822, decomposes a function of time (a signal) into the frequencies that make it up.

In engineering, discrete Fourier transform is applied more frequently, which is realized in PAL.

Consider that a sequence of `N` complex elements $x_0, x_1, \ldots, x_{N-1}$, can be transformed into an N-periodic sequence of complex numbers,

$$X_k = F(x)[k] = \sum_{n=0}^{N-1} x_n \cdot \exp(-i2\pi kn/N), \quad k \in \mathbb{Z}$$

which is the so-called discrete Fourier transform (DFT). For simplicity, as it is N-periodic, `k=0,1,...,N-1` is often adopted.

Likewise, $x_n$ can be transformed back from $X_k$ via inverse discrete Fourier transform (IDFT),

$$x_n = F^{-1}(X)[n] = \frac{1}{N} \sum_{k=0}^{N-1} X_k \cdot \exp(i2\pi nk/N), \; n \in \mathbb{Z}$$

Also, the inverse transform is N-periodic, and generally `n=0,1,...,N-1` is used.

Executing DFT straightforwardly will take a time complexity of $O(N^2)$. Danielson-Lanczos formula shows that the discrete Fourier transform can be computed in $O(N\log_2 N)$, which is the so-called fast Fourier transform (FFT).

However, this formula requires that the length of sequence is of order of 2, which is not satisfied generally. In PAL, chirp z-transform algorithm is employed to deal with the situation that length of sequence is not exactly power of 2, taking advantage of convolution, which assures $O(N\log_2 N)$ time complexity.

For the case of pure real or pure imaginary series, real/imaginary FFT is applied, which reduces the computing time by half, in principle.

## Prerequisites

- No null data in the inputs.
- An ID column must be provided.

## _SYS_AFL.PAL_FFT

This function carries out fast Fourier transform on a complex series of any length.

### Overview of all Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<RESULT table>` | OUT |

### Input Table

| Table | Column | Reference for Output Table | Data Type | Description |
|---|---|---|---|---|
| DATA | 1st column | ID | INTEGER | The ID column that indicates the order of sequence.<br>This must be the first column. |
| | 2nd column | | INTEGER, DECIMAL(p, s), or DOUBLE | Real part of the sequence |

| Table | Column | Reference for Output Table | Data Type | Description |
|---|---|---|---|---|
| | 3rd column (optional) | | INTEGER, DECIMAL(p, s), or DOUBLE | Imaginary part of the sequence |

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| INVERSE | INTEGER | 0 | Indicates whether to use forward FFT or inverse FFT.<br>• 0: Forward FFT<br>• 1: Inverse FFT | |
| NUMBER_TYPE | INTEGER | 1 | Indicates the number type of the data.<br>• 1: Real<br>• 2: Imaginary | Valid only when the Data table consists of 2 columns. |

### Output Tables

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| RESULT | 1st column | ID (in input table) data type | ID (in input table) column name | ID column |
| | 2nd column | DOUBLE | REAL | Real part of the transformed sequence |
| | 3rd column | DOUBLE | IMAG | Imaginary part of the transformed sequence |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;

- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_FFT_DATA_TBL;
CREATE COLUMN TABLE PAL_FFT_DATA_TBL (
    "ID" INTEGER,
    "RE" DOUBLE,
    "IM" DOUBLE
);
INSERT INTO PAL_FFT_DATA_TBL VALUES (1, 2, 9);
INSERT INTO PAL_FFT_DATA_TBL VALUES (2, 3, -3);
INSERT INTO PAL_FFT_DATA_TBL VALUES (3, 5, 0);
INSERT INTO PAL_FFT_DATA_TBL VALUES (4, 0, 0);
INSERT INTO PAL_FFT_DATA_TBL VALUES (5, -2, -2);
INSERT INTO PAL_FFT_DATA_TBL VALUES (6, -9, -7);
INSERT INTO PAL_FFT_DATA_TBL VALUES (7, 7, 0);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR(100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR(100)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('INVERSE', 0, NULL, NULL);
CALL _SYS_AFL.PAL_FFT(PAL_FFT_DATA_TBL, "#PAL_PARAMETER_TBL", ?);
```

**Expected Result**

RESULT:

| | ID | REAL | IMAG |
|---|---|---|---|
| 1 | 1 | 5.999999999999995 | -2.999999999999998 |
| 2 | 2 | 16.273687906321225 | -0.9003174509856899 |
| 3 | 3 | -5.39394634697538 | 26.265112234777575 |
| 4 | 4 | -13.883221788811... | 18.514839745267736 |
| 5 | 5 | -4.233990248281248 | -2.947800028050258 |
| 6 | 6 | 9.65731940363351 | 3.1896183121592436 |
| 7 | 7 | 5.5801510741136315 | 21.87854718683137 |

> **Note**
>
> The wrapper procedure generation approach to call PAL functions is still supported. For more information, refer to the older versions (prior to SAP HANA Platform 2.0 SPS 02) of this document.

## 5.5.16 Forecast Accuracy Measures

Measures are used to check the accuracy of the forecast made by PAL algorithms. They are calculated based on the difference between the historical values and the forecasted values of the fitted model. The measures supported in PAL are MPE, MSE, RMSE, ET, MAD, MASE, WMAPE, SMAPE, MAPE, and SPEC.

For stock-keeping-oriented prediction error costs (SPEC), it consists of stock-keeping costs and opportunity costs. Stock-keeping costs are incurred when forecasted values are larger than the customer needs, while opportunity costs arise when customer demands are not satisfied. The formula of SPEC is as follows:

$$\mathrm{SPEC}_{\alpha_1,\alpha_2} = \frac{1}{n}\sum_{t=1}^{n}\sum_{i=1}^{t}\left(\max\left[0;\min\left[y_i;\sum_{k=1}^{i}y_k-\sum_{j=1}^{t}f_j\right]\cdot\alpha_1;\min\left[f_i;\sum_{k=1}^{i}f_k-\sum_{j=1}^{t}y_j\right]\cdot\alpha_2\right]\cdot(t-i+1)\right)$$

Here $n$ is the length of the time series; $y_t$ is the actual value at time $t$ and $f_t$ is the forecasted value at time $t$. $\alpha_1$ is the opportunity cost parameter and $\alpha_2$ is the stock-keeping cost parameter. Unlike other measures, the result of SPEC is related to the order of the input data, so the ID of the input data is required.

### Prerequisite

The input data is numeric, not categorical.

### _SYS_AFL.PAL_ACCURACY_MEASURES

This is a forecast accuracy measures function.

#### Overview of all Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<RESULT table>` | OUT |

#### Input Table

The input table can be two or three columns. The difference is whether the table has an ID column. For SPEC, the ID column is mandatory. For others, the ID column is optional.

#### Two-column Input Table

| Table | Column | Data Type | Description |
|---|---|---|---|
| DATA | 1st column | DOUBLE | Actual data |
| | 2nd column | DOUBLE or DECIMAL(p, s) | Forecasted data |

### Three-column Input Table

| Table | Column | Data Type | Description |
|---|---|---|---|
| DATA | 1st column | INTEGER | ID of data |
| | 2nd column | DOUBLE | Actual data |
| | 3rd column | DOUBLE or DECIMAL(p, s) | Forecasted data |

### Parameter Table

Mandatory Parameter

The following parameter is mandatory and must be given a value.

| Name | Data Type | Description |
|---|---|---|
| MEASURE_NAME | VARCHAR | Specifies measure name:<br>• MPE: mean percentage error<br>• MSE: mean squared error<br>• RMSE: root mean squared error<br>• ET: error total<br>• MAD: mean absolute deviation<br>• MASE: out-of-sample mean absolute scaled error<br>• WMAPE: weighted mean absolute percentage error<br>• SMAPE: symmetric mean absolute percentage error<br>• MAPE: mean absolute percentage error<br>• SPEC: stock-keeping-oriented prediction error costs |

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| IGNORE_ZERO | INTEGER | 0 | Valid values are:<br>• 0: Uses zero values in the actual dataset when calculating MPE or MAPE;<br>• 1: Ignores zero values in the actual dataset when calculating MPE or MAPE. | Only valid when MEASURE_NAME is MPE or MAPE. |
| ALPHA1 | DOUBLE | 0.5 | Opportunity cost parameter, not less than 0. | Only valid when MEASURE_NAME is SPEC. |
| ALPHA2 | DOUBLE | 0.5 | Stock-keeping cost parameter, not less than 0. | Only valid when MEASURE_NAME is SPEC. |

### Output Table

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| RESULT | 1st column | NVARCHAR(100) | STAT_NAME | Name of accuracy measures |
| | 2nd column | DOUBLE | STAT_VALUE | Value of accuracy measures |

### Examples

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

#### Example 1: The input table without the ID column

```
SET SCHEMA DM_PAL;
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME " VARCHAR(100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (100)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MEASURE_NAME', NULL, NULL, 'MSE');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MEASURE_NAME', NULL, NULL, 'RMSE');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MEASURE_NAME', NULL, NULL, 'MPE');
```

```
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MEASURE_NAME', NULL, NULL, 'ET');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MEASURE_NAME', NULL, NULL, 'MAD');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MEASURE_NAME', NULL, NULL, 'MASE');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MEASURE_NAME', NULL, NULL, 'WMAPE');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MEASURE_NAME', NULL, NULL, 'SMAPE');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MEASURE_NAME', NULL, NULL, 'MAPE');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('IGNORE_ZERO', 1, NULL, NULL);
DROP TABLE PAL_FORECASTACCURACYMEASURES_DATA_TBL;
CREATE COLUMN TABLE PAL_FORECASTACCURACYMEASURES_DATA_TBL (
    "ACTUALCOL" DOUBLE,
    "FORECASTCOL" DOUBLE
);
INSERT INTO PAL_FORECASTACCURACYMEASURES_DATA_TBL VALUES (1130, 1270);
INSERT INTO PAL_FORECASTACCURACYMEASURES_DATA_TBL VALUES (2410, 2340);
INSERT INTO PAL_FORECASTACCURACYMEASURES_DATA_TBL VALUES (2210, 2310);
INSERT INTO PAL_FORECASTACCURACYMEASURES_DATA_TBL VALUES (2500, 2340);
INSERT INTO PAL_FORECASTACCURACYMEASURES_DATA_TBL VALUES (2432, 2348);
INSERT INTO PAL_FORECASTACCURACYMEASURES_DATA_TBL VALUES (1980, 1890);
INSERT INTO PAL_FORECASTACCURACYMEASURES_DATA_TBL VALUES (2045, 2100);
INSERT INTO PAL_FORECASTACCURACYMEASURES_DATA_TBL VALUES (2340, 2231);
INSERT INTO PAL_FORECASTACCURACYMEASURES_DATA_TBL VALUES (2460, 2401);
INSERT INTO PAL_FORECASTACCURACYMEASURES_DATA_TBL VALUES (2350, 2310);
INSERT INTO PAL_FORECASTACCURACYMEASURES_DATA_TBL VALUES (2345, 2340);
INSERT INTO PAL_FORECASTACCURACYMEASURES_DATA_TBL VALUES (2650, 2560);
CALL _SYS_AFL.PAL_ACCURACY_MEASURES(PAL_FORECASTACCURACYMEASURES_DATA_TBL,
#PAL_PARAMETER_TBL, ?);
```

### Expected Result

PAL_FORECASTACCURACYMEASURES_RESULT_TBL:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | ET | 412 |
| 2 | MAD | 83.5 |
| 3 | MAPE | 0.04106315... |
| 4 | MASE | 0.28793103... |
| 5 | MPE | 0.00839023... |
| 6 | MSE | 8,614 |
| 7 | RMSE | 92.8116372... |
| 8 | SMAPE | 0.04087645... |
| 9 | WMAPE | 0.03731565... |

### Example 2: The input table with the ID column

```
SET SCHEMA DM_PAL;
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME " VARCHAR(100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (100)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MEASURE_NAME', NULL, NULL, 'MSE');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MEASURE_NAME', NULL, NULL, 'RMSE');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MEASURE_NAME', NULL, NULL, 'MPE');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MEASURE_NAME', NULL, NULL, 'ET');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MEASURE_NAME', NULL, NULL, 'MAD');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MEASURE_NAME', NULL, NULL, 'MASE');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MEASURE_NAME', NULL, NULL, 'WMAPE');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MEASURE_NAME', NULL, NULL, 'SMAPE');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MEASURE_NAME', NULL, NULL, 'MAPE');
```

```
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MEASURE_NAME', NULL, NULL, 'SPEC');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('ALPHA1', NULL, 0.25, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('ALPHA2', NULL, 0.75, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('IGNORE_ZERO', 1, NULL, NULL);
DROP TABLE PAL_FORECASTACCURACYMEASURES_DATA_TBL;
CREATE COLUMN TABLE PAL_FORECASTACCURACYMEASURES_DATA_TBL (
    "ID" INTEGER,
    "ACTUALCOL" DOUBLE,
    "FORECASTCOL" DOUBLE
);
INSERT INTO PAL_FORECASTACCURACYMEASURES_DATA_TBL VALUES (1, 1130, 1270);
INSERT INTO PAL_FORECASTACCURACYMEASURES_DATA_TBL VALUES (2, 2410, 2340);
INSERT INTO PAL_FORECASTACCURACYMEASURES_DATA_TBL VALUES (3, 2210, 2310);
INSERT INTO PAL_FORECASTACCURACYMEASURES_DATA_TBL VALUES (4, 2500, 2340);
INSERT INTO PAL_FORECASTACCURACYMEASURES_DATA_TBL VALUES (5, 2432, 2348);
INSERT INTO PAL_FORECASTACCURACYMEASURES_DATA_TBL VALUES (6, 1980, 1890);
INSERT INTO PAL_FORECASTACCURACYMEASURES_DATA_TBL VALUES (7, 2045, 2100);
INSERT INTO PAL_FORECASTACCURACYMEASURES_DATA_TBL VALUES (8, 2340, 2231);
INSERT INTO PAL_FORECASTACCURACYMEASURES_DATA_TBL VALUES (9, 2460, 2401);
INSERT INTO PAL_FORECASTACCURACYMEASURES_DATA_TBL VALUES (10, 2350, 2310);
INSERT INTO PAL_FORECASTACCURACYMEASURES_DATA_TBL VALUES (11, 2345, 2340);
INSERT INTO PAL_FORECASTACCURACYMEASURES_DATA_TBL VALUES (12, 2650, 2560);
CALL _SYS_AFL.PAL_ACCURACY_MEASURES(PAL_FORECASTACCURACYMEASURES_DATA_TBL,
#PAL_PARAMETER_TBL, ?);
```

**Expected Result**

PAL_FORECASTACCURACYMEASURES_RESULT_TBL:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | ET | 412 |
| 2 | MAD | 83.5 |
| 3 | MAPE | 0.04106315... |
| 4 | MASE | 0.28793103... |
| 5 | MPE | 0.00839023... |
| 6 | MSE | 8,614 |
| 7 | RMSE | 92.8116372... |
| 8 | SMAPE | 0.04087645... |
| 9 | SPEC | 63.8125 |
| 10 | WMAPE | 0.03731565... |

## 5.5.17 GARCH

Generalized AutoRegressive Conditional Heteroskedasticity (GARCH) is a statistic model used to analysis variance of error (innovation or residual) term in time series. It is typically used in the analyzing of financial data, such as estimation of volatility of returns for stocks and bonds.

GARCH assumes variance of error term is heteroskedastic which means it is not a constant value. In appearance, it tends to cluster. GARCH assumes variance of error term subjects to an autoregressive moving average pattern, in other words it is an average of past values.

Assuming a time series model:

$y_t = \mu_t + \varepsilon_t$

In above equation, $\mu_t$ is called mean model, it can be an ARMA model or just a constant value.

It is $\sigma_t^2 = var(\varepsilon_t|F_{t-1})$, the conditional variance of $\varepsilon_t$ that causes interest, where $F_{t-1}$ stands for information set known at time t – 1.

Define GARCH (p, q) model as:

$$\sigma_t^2=\alpha_0+\sum_{i=1}^{p}\alpha_i\varepsilon_{t-i}^2+\sum_{j=1}^{q}\beta_j\sigma_{t-j}^2$$

where $\alpha_0 > 0, \alpha_i, \beta_j \geq 0$, $i \in [1,p]$, $j \in [1,q]$

In our procedure, it is assumed $\mu_t$ already be deducted from $y_t$. So, input time series is $\varepsilon_t$ only.

Another assumption is $P(\varepsilon_t|F_{t-1}) \sim N(0,\sigma_t^2)$, to estimate model factors with MLE.

## Prerequisites

- The tables used to store the data is a column table and contains no null data.

## _AFL_SYS.PAL_GARCH

This function calculates GARCH model out of time series.

**Overview of all tables**

| Position | Table name | Table type |
|---|---|---|
| 1 | Training Data table | IN |
| 2 | Parameter table | IN |
| 3 | Model table | OUT |
| 4 | Variance table | OUT |
| 5 | Statistic table | OUT |

**Signature**

**Input Table**

| Table | Column | Column Data Type | Description | Constraint |
|---|---|---|---|---|
| Training Data | 1st column | INTEGER | Timestamp | |

| Table | Column | Column Data Type | Description | Constraint |
|---|---|---|---|---|
| | Other columns | INTEGER, DOUBLE, DECIMAL (p, s),VARCHAR or NVARCHAR | One of those columns contains time series data and should be numerical one | |

### Parameter Table

Mandatory Parameters

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data type | Default value | Description | Dependency |
|---|---|---|---|---|
| THREAD_RATIO | DOUBLE | -1 | The ratio of available threads.<br>• 0: single thread<br>• 0~1: percentage<br>• Others: heuristically determined | |
| MODEL_TYPE | INTEGER | 0 | Specifies variants of GARCH model:<br>• 0: GARCH<br>• 1: IGARCH<br>• 2: TGARCH<br>• 3: EGARCH | |
| P | INTEGER | 1 | Number of delayed error terms. Must not be negative. | Ignored if MODEL_TYPE = 1. And P is constant 1. |
| Q | INTEGER | 1 | Number of delayed variance term. Must not be negative. | Ignored if MODEL_TYPE = 1 |
| DEPENDENT_VARIABLE | VARCHAR or NVARCHAR | The last column name of DATA table | Specifies the variable used as time series data. | Cannot be the first column name and should be a numerical one. |

### Output Tables

| Table | Column | Data type | Column name | Description |
|---|---|---|---|---|
| MODEL | 1st column | INTEGER | ROW_INDEX | Indicates the row |
| | 2nd column | NCLOB | MODEL_CONTENT | Model value |
| VARIANCE | 1st column | INTEGER | First column name of DATA table | Timestamp of DATA table |

| Table | Column | Data type | Column name | Description |
|---|---|---|---|---|
| | 2nd column | DOUBLE | VARIANCE | Variance of residual |
| | 3rd column | DOUBLE | RESIDUAL | Residual value |
| STATISTICS | 1st column | NVARCHAR (1000) | STAT_NAME | Statistics name |
| | 2nd column | NVARCHAR (1000) | STAT_VALUE | Statistics value |

### Examples

Assume that:

- `DM_PAL` is a schema belonging to USER1; and
- • USER1 has been assigned the`AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
set schema DM_PAL;
drop table PAL_GARCH_DATA_TAB;
create column table PAL_GARCH_DATA_TAB (
    "TIME" integer,
    "VAR1" integer,
    "VAR2" double,
    "VAR3" nvarchar(100)
);
insert into PAL_GARCH_DATA_TAB values (1, 2, 0.17, 'A');
insert into PAL_GARCH_DATA_TAB values (2, 2, 0.19, 'A');
insert into PAL_GARCH_DATA_TAB values (3, 2, 0.28, 'A');
insert into PAL_GARCH_DATA_TAB values (4, 2, 0.35, 'A');
insert into PAL_GARCH_DATA_TAB values (5, 2, 1.04, 'A');
insert into PAL_GARCH_DATA_TAB values (6, 2, 1.12, 'A');
insert into PAL_GARCH_DATA_TAB values (7, 2, 1.99, 'A');
insert into PAL_GARCH_DATA_TAB values (8, 2, 0.73, 'A');
insert into PAL_GARCH_DATA_TAB values (9, 2, 0.50, 'A');
insert into PAL_GARCH_DATA_TAB values (10, 2, 0.32, 'A');
insert into PAL_GARCH_DATA_TAB values (11, 2, 0.40, 'A');
insert into PAL_GARCH_DATA_TAB values (12, 2, 0.38, 'A');
insert into PAL_GARCH_DATA_TAB values (13, 2, 0.33, 'A');
insert into PAL_GARCH_DATA_TAB values (14, 2, 0.39, 'A');
insert into PAL_GARCH_DATA_TAB values (15, 2, 0.98, 'A');
insert into PAL_GARCH_DATA_TAB values (16, 2, 0.70, 'A');
insert into PAL_GARCH_DATA_TAB values (17, 2, 0.89, 'A');
insert into PAL_GARCH_DATA_TAB values (18, 2, 1.21, 'A');
insert into PAL_GARCH_DATA_TAB values (19, 2, 1.32, 'A');
insert into PAL_GARCH_DATA_TAB values (20, 2, 1.10, 'A');
drop table PAL_GARCH_MODEL_TAB;
create column table PAL_GARCH_MODEL_TAB (
    "ROW_INDEX" integer,
    "MODEL_CONTENT" nclob
);

drop table PAL_PARAMETER_TAB;
create column table PAL_PARAMETER_TAB (
    "PARAM_NAME" nvarchar(100),
    "INT_VALUE" integer,
    "DOUBLE_VALUE" double,
    "STRING_VALUE" nvarchar(100)
);
insert into PAL_PARAMETER_TAB values ('P', 1, null, null);
insert into PAL_PARAMETER_TAB values ('Q', 1, null, null);
insert into PAL_PARAMETER_TAB values ('DEPENDENT_VARIABLE', null, null, 'VAR2');
do begin
    LT_DATA_TAB = select * from PAL_GARCH_DATA_TAB;
    LT_PARAM_TAB = select * from PAL_PARAMETER_TAB;
```

```
    call _SYS_AFL.PAL_GARCH(:LT_DATA_TAB, :LT_PARAM_TAB, LT_MODEL_TAB,
LT_VAR_TAB, LT_STAT_TAB);
    select * from :LT_MODEL_TAB;
    select * from :LT_VAR_TAB;
    select * from :LT_STAT_TAB;
    insert into PAL_GARCH_MODEL_TAB select * from :LT_MODEL_TAB;
end;
```

**Expected Result**

model:

| | ROW_INDEX | MODEL_CONTENT |
|---|---|---|
| 1 | 0 | {"garch":{"factors":[0.13309693320030375,1.060072669991767... |

variance:

| | TIME | VARIANCE | RESIDUAL |
|---|---|---|---|
| 1 | 1 | 0.8958192192593806 | 0.17 |
| 2 | 2 | 0.16373303336306583 | 0.19 |
| 3 | 3 | 0.17136555658700656 | 0.28 |
| 4 | 4 | 0.21620663052765832 | 0.35 |
| 5 | 5 | 0.2629558352742952 | 1.04 |
| 6 | 6 | 1.2796715330633996 | 1.12 |
| 7 | 7 | 1.4628520904379771 | 1.99 |
| 8 | 8 | 4.331090713634703 | 0.73 |
| 9 | 9 | 0.6980096590389167 | 0.5 |
| 10 | 10 | 0.3981151006982456 | 0.32 |
| 11 | 11 | 0.24164837460746075 | 0.4 |
| 12 | 12 | 0.3027085603989866 | 0.38 |
| 13 | 13 | 0.286171426747115 | 0.33 |
| 14 | 14 | 0.24853884696240724 | 0.39 |
| 15 | 15 | 0.29433398630605156 | 0.98 |
| 16 | 16 | 1.1511907254603972 | 0.7 |
| 17 | 17 | 0.6525325414962697 | 0.89 |
| 18 | 18 | 0.9727804951007828 | 1.21 |
| 19 | 19 | 1.6851493293352506 | 1.32 |
| 20 | 20 | 1.9801675533939598 | 1.1 |

## _AFL_SYS.PAL_GARCH_FORECAST

This function forecasts variance of error terms in time series, based on trained GARCH model.

**Overview of all tables**

| Position | Table Name | Table Type |
|---|---|---|
| 1 | Model table | IN |

| Position | Table Name | Table Type |
|---|---|---|
| 2 | Parameter table | IN |
| 3 | Variance table | OUT |
| 4 | Statistic table | OUT |

### Signature

### Input Table

| Table | Column | Column Data Type | Description | Constraint |
|---|---|---|---|---|
| Model | 1st column | INTEGER | Row index | |
| | 2nd column | NCLOB | Model content | |

### Parameter Table

Mandatory Parameters

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Names | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| FORECAST_STEPS | INTEGER | 1 | Number of steps of forecasting | |

### Output Tables

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| VARIANCE | 1st column | INTEGER | First column name of DATA table | Timestamp of DATA table |
| | 2nd column | DOUBLE | VARIANCE | Variance of residual |
| | 3rd column | DOUBLE | RESIDUAL | Residual value |
| STATISTICS | 1st column | NVARCHAR (1000) | STAT_NAME | Statistics name |
| | 2nd column | NVARCHAR (1000) | STAT_VALUE | Statistics value |

### Examples

Assume that:

- DM_PAL is a schema belonging to USER1; and
- USER1 has been assigned the AFL__SYS_AFL_AFLPAL_EXECUTE or AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION role.

```
set schema DM_PAL;
drop table PAL_PARAMETER_TAB;
create column table PAL_PARAMETER_TAB (
    "PARAM_NAME" nvarchar(100),
    "INT_VALUE" integer,
    "DOUBLE_VALUE" double,
    "STRING_VALUE" nvarchar(100)
```

```
);
insert into PAL_PARAMETER_TAB values ('FORECAST_STEPS', 10, null, null);
call _SYS_AFL.PAL_GARCH_FORECAST(PAL_GARCH_MODEL_TAB, PAL_PARAMETER_TAB, ?, ?);
```

**Expected Result**

Variance:

| | STEP | VARIANCE | RESIDUAL |
|---|---|---|---|
| 1 | 1 | 1.415805745585892 | ? |
| 2 | 2 | 1.6339788504841866 | ? |
| 3 | 3 | 1.8652624988850373 | ? |
| 4 | 4 | 2.110444534693459 | ? |
| 5 | 5 | 2.370360145238373 | ? |
| 6 | 6 | 2.6458947062521405 | ? |
| 7 | 7 | 2.9379867978117225 | ? |
| 8 | 8 | 3.247631401514951 | ? |
| 9 | 9 | 3.575883289782771 | ? |
| 10 | 10 | 3.9238606188327507 | ? |

## _AFL_SYS.PAL_GARCH_PREDICT

This function predicts variance of error terms in time series, based on trained GARCH model.

**Overview of all tables**

Position Table name Table type

| Position | Table name | Table type |
|---|---|---|
| 1 | Training Data table | IN |
| 2 | Model table | IN |
| 3 | Parameter table | IN |
| 4 | Variance table | OUT |
| 5 | Statistic table | OUT |

**Signature**

**Input Table**

| Table | Column | Column Data Type | Description | Constraint |
|---|---|---|---|---|
| Predicting Data | 1st column | INTEGER | Timestamp | |
| | Other columns | INTEGER, DOUBLE, DECIMAL (p, s), VARCHAR or NVARCHAR | One of those columns contains time series data and should be numerical one | |
| Model | 1st column | INTEGER | Row index | |
| | 2nd column | NCLOB | Model content | |

### Parameter Table

Mandatory Parameters

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data type | Default value | Description | Dependency |
|---|---|---|---|---|
| DEPENDENT_VARIABLE | VARCHAR or NVARCHAR | The last column name of DATA table | Specifies the variable used as time series data | Cannot be the first column name and should be a numerical one. |

### Output Tables

| Table | Column | Data type | Column name | Description |
|---|---|---|---|---|
| VARIANCE | 1st column | INTEGER | First column name of DATA table | Timestamp of DATA table |
| | 2nd column | DOUBLE | VARIANCE | Variance of residual |
| | 3rd column | DOUBLE | RESIDUAL | Residual value |
| STATISTICS | 1st column | NVARCHAR (1000) | STAT_NAME | Statistics name |
| | 2nd column | NVARCHAR (1000) | STAT_VALUE | Statistics value |

### Examples

Assume that:

- DM_PAL is a schema belonging to USER1; and
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
set schema DM_PAL;
drop table PAL_GARCH_PREDICT_TAB;
create column table PAL_GARCH_PREDICT_TAB (
    "TIME" integer,
    "VAR1" double,
    "VAR2" nvarchar(100)
);
insert into PAL_GARCH_PREDICT_TAB values (1, 0.1, 'A');
insert into PAL_GARCH_PREDICT_TAB values (2, 0.2, 'A');
insert into PAL_GARCH_PREDICT_TAB values (3, 0.3, 'A');
insert into PAL_GARCH_PREDICT_TAB values (4, 0.4, 'A');
insert into PAL_GARCH_PREDICT_TAB values (5, 0.5, 'A');
```

```
insert into PAL_GARCH_PREDICT_TAB values (6, 0.6, 'A');
insert into PAL_GARCH_PREDICT_TAB values (7, 0.7, 'A');
insert into PAL_GARCH_PREDICT_TAB values (8, 0.8, 'A');
insert into PAL_GARCH_PREDICT_TAB values (9, 0.9, 'A');
insert into PAL_GARCH_PREDICT_TAB values (10, 1.0, 'A');
drop table PAL_PARAMETER_TAB;
create column table PAL_PARAMETER_TAB (
    "PARAM_NAME" nvarchar(100),
    "INT_VALUE" integer,
    "DOUBLE_VALUE" double,
    "STRING_VALUE" nvarchar(100)
);
insert into PAL_PARAMETER_TAB values ('DEPENDENT_VARIABLE', null, null, 'VAR1');
call _SYS_AFL.PAL_GARCH_PREDICT(PAL_GARCH_PREDICT_TAB, PAL_GARCH_MODEL_TAB,
PAL_PARAMETER_TAB, ?, ?);
```

**Expected Result**

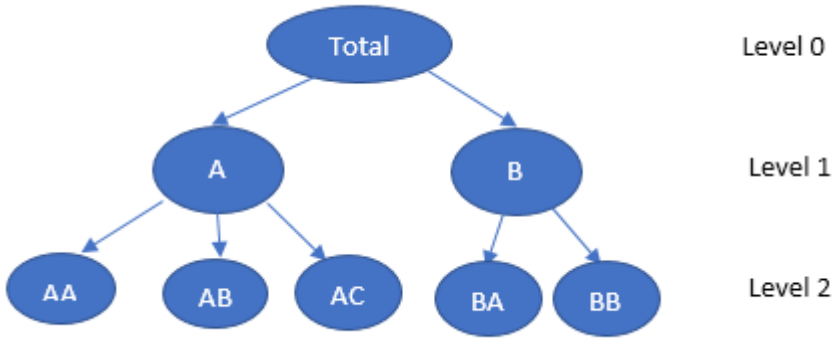
variance:

| | TIME | VARIANCE | RESIDUAL |
|---|---|---|---|
| 1 | 1 | 1.4157807000000004 | 0.1 |
| 2 | 2 | 0.14369669999999998 | 0.2 |
| 3 | 3 | 0.1754988 | 0.3 |
| 4 | 4 | 0.2285023 | 0.4 |
| 5 | 5 | 0.30270720000000007 | 0.5 |
| 6 | 6 | 0.3981135 | 0.6 |
| 7 | 7 | 0.5147212 | 0.7 |
| 8 | 8 | 0.6525303 | 0.8 |
| 9 | 9 | 0.8115408000000002 | 0.9 |
| 10 | 10 | 0.9917527000000002 | 1 |

## 5.5.18 Hierarchical Forecast

In many applications, there are multiple time series that are hierarchically organized and can be aggregated at several different levels in groups based on products, geography, or some other features. We call these hierarchical time series.

The structure is:




We consider hierarchical models by matrix and vector notation. $Y_{i,t}$ denotes the vector of all observations at level `i` and time `t` then, $Y_t=[Y_t, Y_{1,t}, \ldots, Y_{K,t}]'$. Using the 'summing' matrix, `S`, which stores the structure of the hierarchy, $Y_t$ can be found from the bottom level series, $Y_t=SY_{K,t}$.

The summing matrix `S` is a matrix of order $m*m_K$. Where `m` is the total number of series in the hierarchy, and the number of series in each level is $m_i$ $(i=0,1,\ldots,K)$. For the above hierarchy model, the summing matrix is:

$$S=\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

This has led to the need for reconciling forecasts across the hierarchy (that is, ensuring the forecasts sum appropriately across the levels).

There are commonly forecast methods to handle it. In PAL, we provide three forecast methods:

- **Bottom-up**
  A commonly applied method for hierarchical forecasting is bottom-up approach. This approach involves generating base independent forecasts for each series at the bottom level of the hierarchy, and then aggregating these upwards to produce revised forecasts for the whole hierarchy.
- **Top-down**
  Top-down approaches involves generating base forecasts for the "Total" series $y_t$ on the top of the hierarchy and then disaggregating these downwards. We let $p_1, p_2, \ldots, p_{m_k}$ be a set of proportions, which is calculated based on average historical proportion and dictate how the base forecasts of the "Total" series are to be distributed to revised forecasts for each series at the bottom level of the hierarchy. Once the bottom level forecasts have been generated, we can use the summing matrix to generate forecasts for the rest of the series in the hierarchy.
- **Optimal combination**

This approach incorporates the use of a regression model to optimally combine and reconcile forecasts, involving generating independent base forecast for each series in the hierarchy. As these base forecasts are independently generated, they will not add up according to the hierarchical structure. The optimal combination approach, optimally combines the independent base forecasts and generates a set of revised forecasts that are as close as possible to the univariate forecast but also aggregates consistently with the hierarchical structure. Optimal combination approach uses all available information within hierarchy, provided the base forecasts are unbiased and revised forecasts are unbiased.

## Prerequisites

- No missing or null data in predictive data table.

## _SYS_AFL.PAL_HIERARCHICAL_FORECAST

This function reconciles forecasts across the hierarchy.

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<ORIGINAL DATA table>` | IN |
| 2 | `<PREDICTIVE DATA table>` | IN |
| 3 | `<STRUCTURE table>` | IN |
| 4 | `<PARAMETERS table>` | IN |
| 5 | `<RESULT table>` | OUT |
| 6 | `<STATISTICS table>` | OUT |

### Input Table

| Table | Column | Reference for Output Table | Data Type | Description |
|---|---|---|---|---|
| ORIGINAL DATA | 1st column | | NVARCHAR | Time series name. |
| | 2nd column | | INTEGER | Time stamp. |
| | 3rd column | | INTEGER, DECIMAL(p, s), or DOUBLE | Raw data. |

| Table | Column | Reference for Output Table | Data Type | Description |
|---|---|---|---|---|
| | 4th column | | INTEGER, DECIMAL(p, s), or DOUBLE | Residuals values of base forecasts. |
| PREDICTIVE DATA | 1st column | NAME | NVARCHAR | Time series name. |
| | 2nd column | TS | INTEGER | Time stamp. |
| | 3rd column | VALUE | INTEGER, DECIMAL(p, s), or DOUBLE | Predictive raw data. |
| STRUCTURE DATA | 1st column | | INTEGER | ID. |
| | 2nd column | | NVARCHAR | Time series name. |
| | 3rd column | | INTEGER | Number of child nodes at each level. |

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified. PAL will use its default value.

| Name | Data type | Default Value | Description |
|---|---|---|---|
| METHOD | INTEGER | 0 | Method for reconciling forecasts across hierarchy:<br>• 0: optimal combination.<br>• 1: bottom-up.<br>• 2: top-down. |
| WEIGHTS | INTEGER | 0 | Only valid when parameter METHOD is 0:<br>• 0: Ordinary least squares.<br>• 1: Minimum trace.<br>• 2: Weighted least squares. |

### Output Tables

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| RESULT DATA | 1st column | NVARCHAR | NAME (in input table) column | Time series name. |
| | 2nd column | INTEGER | TS (in input table) column | Time stamp. |
| | 3rd column | INTEGER or DOUBLE | VALUE (in input) column | Raw data. |
| STATISTICS | 1st column | NVARCHAR (256) | STAT_NAME | Statistic names. |
| | 2nd column | NVARCHAR (1000) | STAT_VALUE | Statistic values. |

### Examples

Assume that:

- DM_PAL is a schema belonging to USER1;
- USER1 has been assigned the AFL__SYS_AFL_AFLPAL_EXECUTE or AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION role.

```
SET SCHEMA DM_PAL;
-- prepare original data table
DROP TABLE ORIGINAL_TAB;
CREATE COLUMN TABLE ORIGINAL_TAB("Series" VARCHAR(100), "TimeStamp" INTEGER,
"Original" DOUBLE, "Residual" double);
INSERT INTO ORIGINAL_TAB VALUES('Total',1992,48.74808009,0.058251736);
INSERT INTO ORIGINAL_TAB VALUES('Total',1993,49.48046916,0.236069215);
INSERT INTO ORIGINAL_TAB VALUES('Total',1994,49.93238408,-0.044404927);
INSERT INTO ORIGINAL_TAB VALUES('Total',1995,50.24070241,-0.188001523);
INSERT INTO ORIGINAL_TAB VALUES('Total',1996,50.60846449,-0.128557779);
INSERT INTO ORIGINAL_TAB VALUES('Total',1997,50.84850648,-0.256277857);
INSERT INTO ORIGINAL_TAB VALUES('Total',1998,51.70922002,0.364393685);
INSERT INTO ORIGINAL_TAB VALUES('Total',1999,51.9432984,-0.262241472);
INSERT INTO ORIGINAL_TAB VALUES('Total',2000,52.57795621,0.138337958);
INSERT INTO ORIGINAL_TAB VALUES('Total',2001,53.21495876,0.1406827);
INSERT INTO ORIGINAL_TAB VALUES('A',1992,27.21133833,0.026941198);
INSERT INTO ORIGINAL_TAB VALUES('A',1993,27.83882668,0.35736118);
INSERT INTO ORIGINAL_TAB VALUES('A',1994,28.14534842,0.036394576);
INSERT INTO ORIGINAL_TAB VALUES('A',1995,28.27712455,-0.138351039);
INSERT INTO ORIGINAL_TAB VALUES('A',1996,28.47800096,-0.069250754);
INSERT INTO ORIGINAL_TAB VALUES('A',1997,28.56446636,-0.183661771);
INSERT INTO ORIGINAL_TAB VALUES('A',1998,28.90753332,0.072939797);
INSERT INTO ORIGINAL_TAB VALUES('A',1999,29.02154763,-0.156112852);
INSERT INTO ORIGINAL_TAB VALUES('A',2000,29.40308006,0.111405265);
INSERT INTO ORIGINAL_TAB VALUES('A',2001,29.64248283,-0.030724402);
INSERT INTO ORIGINAL_TAB VALUES('B',1992,21.53674176,0.021310538);
INSERT INTO ORIGINAL_TAB VALUES('B',1993,21.64164248,-0.121291965);
INSERT INTO ORIGINAL_TAB VALUES('B',1994,21.78703566,-0.080799503);
INSERT INTO ORIGINAL_TAB VALUES('B',1995,21.96357786,-0.049650484);
INSERT INTO ORIGINAL_TAB VALUES('B',1996,22.13046352,-0.059307026);
INSERT INTO ORIGINAL_TAB VALUES('B',1997,22.28404012,-0.072616086);
INSERT INTO ORIGINAL_TAB VALUES('B',1998,22.8016867,0.291453889);
INSERT INTO ORIGINAL_TAB VALUES('B',1999,22.92175077,-0.10612862);
INSERT INTO ORIGINAL_TAB VALUES('B',2000,23.17487615,0.026932693);
INSERT INTO ORIGINAL_TAB VALUES('B',2001,23.57247593,0.171407102);
INSERT INTO ORIGINAL_TAB VALUES('AA',1992,7.862669196,0.007719336);
INSERT INTO ORIGINAL_TAB VALUES('AA',1993,8.34035474,0.334355942);
INSERT INTO ORIGINAL_TAB VALUES('AA',1994,8.549132204,0.065447862);
```

```
INSERT INTO ORIGINAL_TAB VALUES('AA',1995,8.597698777,-0.094763029);
INSERT INTO ORIGINAL_TAB VALUES('AA',1996,8.770196121,0.029167742);
INSERT INTO ORIGINAL_TAB VALUES('AA',1997,8.837792506,-0.075733217);
INSERT INTO ORIGINAL_TAB VALUES('AA',1998,8.893266655,-0.087855453);
INSERT INTO ORIGINAL_TAB VALUES('AA',1999,8.933993433,-0.102602824);
INSERT INTO ORIGINAL_TAB VALUES('AA',2000,9.133652687,0.056329652);
INSERT INTO ORIGINAL_TAB VALUES('AA',2001,9.152635614,-0.124346675);
INSERT INTO ORIGINAL_TAB VALUES('AB',1992,9.267647091,0.009183739);
INSERT INTO ORIGINAL_TAB VALUES('AB',1993,9.416543734,0.064992652);
INSERT INTO ORIGINAL_TAB VALUES('AB',1994,9.479022166,-0.021425559);
INSERT INTO ORIGINAL_TAB VALUES('AB',1995,9.532376112,-0.030550045);
INSERT INTO ORIGINAL_TAB VALUES('AB',1996,9.54759077,-0.068689333);
INSERT INTO ORIGINAL_TAB VALUES('AB',1997,9.555663663,-0.075831097);
INSERT INTO ORIGINAL_TAB VALUES('AB',1998,9.730346869,0.090779215);
INSERT INTO ORIGINAL_TAB VALUES('AB',1999,9.797293978,-0.016956882);
INSERT INTO ORIGINAL_TAB VALUES('AB',2000,9.921261705,0.040063736);
INSERT INTO ORIGINAL_TAB VALUES('AB',2001,10.02278301,0.017617313);
INSERT INTO ORIGINAL_TAB VALUES('AC',1992,10.08102205,0.010038123);
INSERT INTO ORIGINAL_TAB VALUES('AC',1993,10.08192821,-0.041987414);
INSERT INTO ORIGINAL_TAB VALUES('AC',1994,10.11719405,-0.007627727);
INSERT INTO ORIGINAL_TAB VALUES('AC',1995,10.14704966,-0.013037965);
INSERT INTO ORIGINAL_TAB VALUES('AC',1996,10.16021407,-0.029729163);
INSERT INTO ORIGINAL_TAB VALUES('AC',1997,10.17101019,-0.032097457);
INSERT INTO ORIGINAL_TAB VALUES('AC',1998,10.2839198,0.070016035);
INSERT INTO ORIGINAL_TAB VALUES('AC',1999,10.29026022,-0.036553147);
INSERT INTO ORIGINAL_TAB VALUES('AC',2000,10.34816567,0.015011877);
INSERT INTO ORIGINAL_TAB VALUES('AC',2001,10.46706421,0.07600496);
INSERT INTO ORIGINAL_TAB VALUES('BA',1992,10.49276117,0.010432473);
INSERT INTO ORIGINAL_TAB VALUES('BA',1993,10.51711828,-0.035926057);
INSERT INTO ORIGINAL_TAB VALUES('BA',1994,10.54012728,-0.037274163);
INSERT INTO ORIGINAL_TAB VALUES('BA',1995,10.61219997,0.011789519);
INSERT INTO ORIGINAL_TAB VALUES('BA',1996,10.64931887,-0.023164273);
INSERT INTO ORIGINAL_TAB VALUES('BA',1997,10.7073609,-0.002241141);
INSERT INTO ORIGINAL_TAB VALUES('BA',1998,10.86322633,0.095582262);
INSERT INTO ORIGINAL_TAB VALUES('BA',1999,10.92462763,0.001118138);
INSERT INTO ORIGINAL_TAB VALUES('BA',2000,11.00622388,0.021313074);
INSERT INTO ORIGINAL_TAB VALUES('BA',2001,11.03530969,-0.031197358);
INSERT INTO ORIGINAL_TAB VALUES('BB',1992,11.04398059,0.010878066);
INSERT INTO ORIGINAL_TAB VALUES('BB',1993,11.1245242,-0.085365908);
INSERT INTO ORIGINAL_TAB VALUES('BB',1994,11.24690838,-0.04352534);
INSERT INTO ORIGINAL_TAB VALUES('BB',1995,11.35137789,0.061440003);
INSERT INTO ORIGINAL_TAB VALUES('BB',1996,11.48114466,-0.036142753);
INSERT INTO ORIGINAL_TAB VALUES('BB',1997,11.57667923,-0.10374945);
INSERT INTO ORIGINAL_TAB VALUES('BB',1998,11.93846037,0.195871626);
INSERT INTO ORIGINAL_TAB VALUES('BB',1999,11.99712313,-0.107246758);
INSERT INTO ORIGINAL_TAB VALUES('BB',2000,12.16865227,0.05619619);
INSERT INTO ORIGINAL_TAB VALUES('BB',2001,12.53716625,0.20260446);
-- prepare predict table
DROP TABLE PREDICT_TAB;
CREATE COLUMN TABLE PREDICT_TAB("Series" VARCHAR(100), "TimeStamp" INTEGER,
"VALUE" DOUBLE);
INSERT INTO PREDICT_TAB VALUES('Total',2002,54.71127862);
INSERT INTO PREDICT_TAB VALUES('Total',2003,54.20759847);
INSERT INTO PREDICT_TAB VALUES('Total',2004,54.70391832);
INSERT INTO PREDICT_TAB VALUES('Total',2005,55.20023817);
INSERT INTO PREDICT_TAB VALUES('Total',2006,55.69655803);
INSERT INTO PREDICT_TAB VALUES('Total',2007,56.19287788);
INSERT INTO PREDICT_TAB VALUES('A',2002,29.91260999);
INSERT INTO PREDICT_TAB VALUES('A',2003,30.18273716);
INSERT INTO PREDICT_TAB VALUES('A',2004,30.45286433);
INSERT INTO PREDICT_TAB VALUES('A',2005,30.72299149);
INSERT INTO PREDICT_TAB VALUES('A',2006,30.99311866);
INSERT INTO PREDICT_TAB VALUES('A',2007,31.26324583);
INSERT INTO PREDICT_TAB VALUES('B',2002,23.79866862);
INSERT INTO PREDICT_TAB VALUES('B',2003,24.02486131);
INSERT INTO PREDICT_TAB VALUES('B',2004,24.25105399);
INSERT INTO PREDICT_TAB VALUES('B',2005,24.47724668);
INSERT INTO PREDICT_TAB VALUES('B',2006,24.70343937);
```

```
INSERT INTO PREDICT_TAB VALUES('B',2007,24.92963205);
INSERT INTO PREDICT_TAB VALUES('AA',2002,9.295965216);
INSERT INTO PREDICT_TAB VALUES('AA',2003,9.439294818);
INSERT INTO PREDICT_TAB VALUES('AA',2004,9.58262442);
INSERT INTO PREDICT_TAB VALUES('AA',2005,9.725954022);
INSERT INTO PREDICT_TAB VALUES('AA',2006,9.869283624);
INSERT INTO PREDICT_TAB VALUES('AA',2007,10.01261323);
INSERT INTO PREDICT_TAB VALUES('AB',2002,10.106687);
INSERT INTO PREDICT_TAB VALUES('AB',2003,10.19059099);
INSERT INTO PREDICT_TAB VALUES('AB',2004,10.27449498);
INSERT INTO PREDICT_TAB VALUES('AB',2005,10.35839897);
INSERT INTO PREDICT_TAB VALUES('AB',2006,10.44230296);
INSERT INTO PREDICT_TAB VALUES('AB',2007,10.52620695);
INSERT INTO PREDICT_TAB VALUES('AC',2002,10.50995778);
INSERT INTO PREDICT_TAB VALUES('AC',2003,10.55285135);
INSERT INTO PREDICT_TAB VALUES('AC',2004,10.59574493);
INSERT INTO PREDICT_TAB VALUES('AC',2005,10.6386385);
INSERT INTO PREDICT_TAB VALUES('AC',2006,10.68153207);
INSERT INTO PREDICT_TAB VALUES('AC',2007,10.72442565);
INSERT INTO PREDICT_TAB VALUES('BA',2002,11.09559286);
INSERT INTO PREDICT_TAB VALUES('BA',2003,11.15587602);
INSERT INTO PREDICT_TAB VALUES('BA',2004,11.21615919);
INSERT INTO PREDICT_TAB VALUES('BA',2005,11.27644236);
INSERT INTO PREDICT_TAB VALUES('BA',2006,11.33672553);
INSERT INTO PREDICT_TAB VALUES('BA',2007,11.3970087);
INSERT INTO PREDICT_TAB VALUES('BB',2002,12.70307577);
INSERT INTO PREDICT_TAB VALUES('BB',2003,12.96898528);
INSERT INTO PREDICT_TAB VALUES('BB',2004,12.9348948);
INSERT INTO PREDICT_TAB VALUES('BB',2005,13.20080432);
INSERT INTO PREDICT_TAB VALUES('BB',2006,13.36671384);
INSERT INTO PREDICT_TAB VALUES('BB',2007,14.13262335);
-- prepare structure table
DROP TABLE STRUCTURE_TAB;
CREATE COLUMN TABLE STRUCTURE_TAB("Index" INTEGER, "Series" VARCHAR(100), "NUM"
INTEGER);
INSERT INTO STRUCTURE_TAB VALUES(1,'Total',2);
INSERT INTO STRUCTURE_TAB VALUES(2,'A',3);
INSERT INTO STRUCTURE_TAB VALUES(3,'B',2);
INSERT INTO STRUCTURE_TAB VALUES(4,'AA',0);
INSERT INTO STRUCTURE_TAB VALUES(5,'AB',0);
INSERT INTO STRUCTURE_TAB VALUES(6,'AC',0);
INSERT INTO STRUCTURE_TAB VALUES(7,'BA',0);
INSERT INTO STRUCTURE_TAB VALUES(8,'BB',0);
-- prepare parameter
DROP TABLE PAL_PARAMETER_HTS_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_HTS_TBL(
       "PARAM_NAME" NVARCHAR(256),
       "INT_VALUE" INTEGER,
       "DOUBLE_VALUE" DOUBLE,
       "STRING_VALUE" NVARCHAR(1000)
);
-- 0: optimal combination; 1: bottom-up; 2: top-down
INSERT INTO PAL_PARAMETER_HTS_TBL VALUES ('METHOD', 0, null,null);
-- 0: ols; 1: MinT; 2: wls. this parameter only valid when METHOD is 0.
INSERT INTO PAL_PARAMETER_HTS_TBL VALUES ('WEIGHTS', 1, null,null);
CALL _SYS_AFL.PAL_HIERARCHICAL_FORECAST(DM_PAL.ORIGINAL_TAB, DM_PAL.PREDICT_TAB,
DM_PAL.STRUCTURE_TAB, PAL_PARAMETER_HTS_TBL, ?, ?);
```

### Expected Results

RESULT DATA:

| | Series | TimeStamp | VALUE |
|---|---|---|---|
| 1 | Total | 2,002 | 48.862704524819264 |
| 2 | Total | 2,003 | 54.2556314478982 |
| 3 | Total | 2,004 | 54.66368772094472 |
| 4 | Total | 2,005 | 55.192435733767724 |
| 5 | Total | 2,006 | 55.7199652734216 |
| 6 | Total | 2,007 | 56.43426142010945 |
| 7 | A | 2,002 | 27.204558325255768 |
| 8 | A | 2,003 | 30.215277973104776 |
| 9 | A | 2,004 | 30.424968305284633 |
| 10 | A | 2,005 | 30.71834668841508 |
| 11 | A | 2,006 | 31.00705304079031 |
| 12 | A | 2,007 | 31.43062200419304 |
| 13 | B | 2,002 | 21.658146218189952 |
| 14 | B | 2,003 | 24.040353500870463 |
| 15 | B | 2,004 | 24.238719428698605 |
| 16 | B | 2,005 | 24.47408906397909 |
| 17 | B | 2,006 | 24.71291224380716 |
| 18 | B | 2,007 | 25.003639415916407 |
| 19 | AA | 2,002 | 8.51487299937618 |
| 20 | AA | 2,003 | 9.475738664927327 |
| 21 | AA | 2,004 | 9.549233296508934 |
| 22 | AA | 2,005 | 9.72290129318236 |
| 23 | AA | 2,006 | 9.878441796993993 |
| 24 | AA | 2,007 | 10.21295995969811 |
| 25 | AB | 2,002 | 9.186801271243143 |
| 26 | AB | 2,003 | 10.186817058088039 |
| 27 | AB | 2,004 | 10.278773426412576 |
| 28 | AB | 2,005 | 10.357894454557274 |
| 29 | AB | 2,006 | 10.44381650276927 |
| 30 | AB | 2,007 | 10.500536269769459 |

STATISTICS:

empty

## 5.5.19 Intermittent Time Series Forecast (ITSF)

Intermittent Time Series Forecast (ITSF) is a forecast strategy for products with intermittent demand.

The ITSF model can be expressed as the below formula showed:

If $x_t = 0$:

$x_t = \hat{x}_{t-1}$,

$\hat{n}_t = \hat{n}_{t-1}$,

else:

$\hat{x}_t = \left(\frac{p}{t}\right) x_t + \left(\frac{t-p}{t}\right) \hat{x}_{t-1}$,

$\hat{n}_t = (\frac{q}{t}) n_t + \left(\frac{t-q}{t}\right) \hat{n}_{t-1}$.

Forecast equation is as follows:

$\hat{y}_{t+m} = \frac{\hat{x}_t}{\hat{n}_t}$.

Where `m = 1,2,3,...`, `1<=p<=n`, and `1<=q<=p`, and letting $\hat{x}_t = x_t$ for `t<=p` and letting $\hat{n}_t = n_t$ for `t<=q`,

$x_t$ is the original demand of the series,

$\hat{x}_t$ is the smoothed value of demand,

$\hat{n}_t$ is the estimate for the time interval between non-zero demands,

$n_t$ is the time interval since the last non-zero demand,

$\hat{y}_t$ is the estimate for the average size of non-zero demands,

and `P` and `Q` are parameters.

The same as the croston method, ITSF supports two different algorithm types to output: constant mode and sporadic mode.

- Constant mode: average demand between two non-zero demands, which means $x_t = \hat{x}_{t-1}$ if $x_t = 0$, then $\hat{y}_t = \frac{\hat{x}_t}{\hat{n}_t}$, as the formula is shown.

- Sporadic mode: predicts the demand at the non-zero demand time, which means $\hat{x}_t = 0$ if $x_t = 0$, then $\hat{y}_t = \hat{x}_t$ with $\hat{n}_t$ interval time.

Difference to constant weight of the croston method,

1. ITSF provides a exponential weight to estimate, which means the closer the data, the greater the weight;
2. ITSF does not need the initial value of non-zero demands & time interval between non-zero demands.

## Prerequisites

1. No missing or null data in the inputs.
2. The data is numeric, not categorical.

## _SYS_AFL.PAL_ITSF

This is a ITSF function.

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA Table>` | IN |
| 2 | `<PARAMETER Table>` | IN |
| 3 | `<RESULT Table>` | OUT |
| 4 | `<STATISTICS Table>` | OUT |

### Input Table

| Table | Column | Reference for Output Table | Data Type | Description |
|---|---|---|---|---|
| DATA | 1st column | TS_ID | INTEGER | Record id |
| | 2nd column | TS_DATA | INTEGER, DOUBLE, and DECIMAL (p,s) | Raw data |

### Parameter Table

Mandatory Parameters

None

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| P | INTEGER | -1 | Smoothing parameter.<br>• -1: Optimize the parameter automatically;<br>• Positive values ([1,n]): forecast by manually specifying value. |
| Q | INTEGER | -1 | Smoothing parameter.<br>• -1: Optimize the parameter automatically;<br>• Non-negative values ([1,P]): forecast by manually specifying value. |
| FORECAST_NUM | INTEGER | 1 | Forecast length. When it is set to 1, the algorithm only forecasts one value. |
| METHOD | INTEGER | 0 | Specifying optimization function for P & Q:<br>• 0: Simulated annealing method. A probabilistic technique for approximating the global optimum of a given function.<br>• 1: Brute method. Grid search followed by LBFGSB optimization.<br>• 2: Bounded Limited-memory Broyden-Fletcher-Goldfarb-Shanno (LBFGSB) method. |
| ALGORITHM_TYPE | INTEGER | 0 | Specifying the algorithm type:<br>• Constant mode<br>• Sporadic mode |
| BRUTE_STEP | INTEGER | 20 | Specifying steps ranged from the start point to the length of data. Only valid for brute method.<br>* The default value is the data size if the data size < 20.<br>* The range is [1, n]. |
| OPTIMIZE_STEP | DOUBLE | 0.001 | Specifying minimum step for each iteration of LBFGSB method. Only valid for LBFGSB method. |

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| MEASURE_NAME | VARCHAR | MSE | Measure name. Supports MSE, RMSE, MAE, MAPE, sMAPE, and MASE. For detailed description, see Forecast Accuracy Measures [page 762]. |
| EXPOST_FLAG | INTEGER | 1 | • 0: Does not output the expost forecast, and just outputs the forecast values;<br>• 1: Outputs the expost forecast and the forecast values. |
| IGNORE_ZERO | INTEGER | 0 | • 0: Uses zero values in the input dataset when calculating MAPE;<br>• 1: Ignores zero values in the input dataset when calculating MAPE. |
| THREAD_RATIO | DOUBLE | 0 | The ratio of available threads for multi-thread task:<br>• 0: single thread;<br>• 0~1: percentage. |
| SEED | INTEGER | 1 | Specifies the seed for random number generator. Valid for Simulated annealing method. |
| ITER_COUNT | INTEGER | 1000 | Iteration times for the simulated annealing method. |
| PENALTY | DOUBLE | 1.0 | A penalty is applied to the cost function to avoid overfitting. |

## Output Tables

| Table | Column | Column Data Type | Column Name | Description |
|---|---|---|---|---|
| RESULT | 1st column | TS_ID (in input table) data type | TS_ID (in input table) column name | Data id |
| | 2nd column | TS_DATA (in input table) data type | TS_DATA (in input table) column name | Output result |
| STATISTICS | 1st column | NVARCHAR | STAT_NAME | Name of statistics |
| | 2nd column | DOUBLE | STAT_VALUE | Value of statistics |

## Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

Example:

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_ITSF_DATA_TBL;
CREATE COLUMN TABLE PAL_ITSF_DATA_TBL (
"ID" INT,
"RAWDATA" DOUBLE
);
INSERT INTO PAL_ITSF_DATA_TBL VALUES (0, 0.0);
INSERT INTO PAL_ITSF_DATA_TBL VALUES (1, 1.0);
INSERT INTO PAL_ITSF_DATA_TBL VALUES (2, 4.0);
INSERT INTO PAL_ITSF_DATA_TBL VALUES (3, 0.0);
INSERT INTO PAL_ITSF_DATA_TBL VALUES (4, 0.0);
INSERT INTO PAL_ITSF_DATA_TBL VALUES (5, 0.0);
INSERT INTO PAL_ITSF_DATA_TBL VALUES (6, 5.0);
INSERT INTO PAL_ITSF_DATA_TBL VALUES (7, 3.0);
INSERT INTO PAL_ITSF_DATA_TBL VALUES (8, 0.0);
INSERT INTO PAL_ITSF_DATA_TBL VALUES (9, 0.0);
INSERT INTO PAL_ITSF_DATA_TBL VALUES (10, 0.0);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
"PARAM_NAME" VARCHAR (256),
"INT_VALUE" INTEGER,
"DOUBLE_VALUE" DOUBLE,
"STRING_VALUE" VARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('P', -1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('Q', -1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FORECAST_NUM',3, NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES('MEASURE_NAME',NULL,NULL,'MSE');
INSERT INTO #PAL_PARAMETER_TBL VALUES('THREAD_RATIO', NULL, 0.5, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES('EXPOST_FLAG', 0, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES('IGNORE_ZERO', 0, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES('METHOD', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES('BRUTE_STEP', 11, NULL, NULL);
CALL "_SYS_AFL"."PAL_ITSF"(PAL_ITSF_DATA_TBL, #PAL_PARAMETER_TBL, ?, ?);
```

**Expected Result:**

RESULT:

Rows (3)

| | ID | RAWDATA |
|---|---|---|
| 1 | 11 | 2.004830917874396 |
| 2 | 12 | 2.004830917874396 |
| 3 | 13 | 2.004830917874396 |

STATISTIC:

Rows (5)

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | MSE | 8.293467078189302 |
| 2 | LAST_DEMAND | 3.952380952380953 |
| 3 | LAST_INTERVAL | 1.7142857142857144 |
| 4 | OPT_P | 2 |
| 5 | OPT_Q | 2 |

## 5.5.20 Linear Regression with Damped Trend and Seasonal Adjust

Linear regression with damped trend and seasonal adjust is an approach for forecasting when a time series presents a trend.

In PAL, it provides a damped smoothing parameter for smoothing the forecasted values. This dampening parameter avoids the over-casting due to the "indefinitely" increasing or decreasing trend. In addition, if the time series presents seasonality, you can deal with it by providing the length of the periods in order to adjust the forecasting results. On the other hand, it also helps you to detect the seasonality and to determine the periods.

This algorithm decomposes the time series with multiplicative model. Therefore, it works for time series subject to a multiplicative model. Suppose we have the time series $x_t$, $t = 0,1,...,N-1$, with period length $T$. In PAL, the algorithm works as follows.

1. Obtain the length of periods. It can be provided by you or detected by our algorithm.
2. Obtain the seasonal indexes. There are two methods as follows:
   - Method 1: Average method. In this method, we get the mean of the time series as $\bar{x} = \frac{1}{N}\sum_{t=0}^{N-1} x_t$. The seasonal index is $Index_i = \frac{1}{n\bar{x}} \sum_{j \geq 0, i+jT<N} x_{i+jT}$, $i = 0,1,...,T-1$, here $n$ is the number of the elements in the sum.
   - Method 2: Fitting linear regression. In this method, firstly we get the linear fitting of the raw time series as $\hat{x}_t$, $t = 0,1,...,N-1$. Secondly, we de-trend the time series and get $y_t = x_t/\hat{x}_t$, $t = 0,1,...,N-1$. Finally, we get the seasonal index $Index_i = \frac{1}{n} \sum_{j \geq 0, i+jT<N} y_{i+jT}$, $i = 0,1,...,T-1$, here $n$ is the number of the elements in the sum.
3. Linear fitting of de-season data. The de-season data is $z_t = x_t/Index_{t\%T}$. In this step, we get the intercept $b$ and slope $k$.
4. Forecast. The forecast part is divided into expost forecast and future forecast. In the expost forecast, the expost forecast at $t$ is $f_t = h_t * Index_{t\%T}$, where $h_t = \Phi kt + b$ is the expost linear forecast. $\Phi$ is the dampening trend factor. If AFFECT_FUTURE_ONLY = 1, $\Phi$ is 1 in the expost forecast. The furture forecast is based on the last value of expost linear forecast $h_{N-1}$. In the future forecast with forecast length $M$, the future forecast at $N-1+m$ is $f_{N-1+m} = h_{N-1+m} * Index_{(N-1+m)\%T}$, where $h_{N-1+m} = \left(\sum_{p=1}^{m} \phi^p\right) k + h_{N-1}$, $m = 1,2,...,M$.

> **Note**
>
> Occasionally, there is probability that the average, the linear forecast, or the seasonal index is calculated to be 0, which gives rise to the issue of division by zero in the subsequent calculation. To address this, therefore, a tiny value, 1.0e-6 for example, is adopted as the divisor instead of 0. In the Result output table, an indicator named "HandleZero" is given, which represents if the substitution takes place (1) or not (0).

## Prerequisites

- No missing or null data in the inputs. The algorithm will issue errors when encountering null values.
- The data is numeric, not categorical.
- The number of valid historical data should be at least equal to the value of the PERIODS parameter.

## _SYS_AFL.PAL_LR_SEASONAL_ADJUST

This is the function for linear regression with damped trend and seasonal adjust.

### Overview of all Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<FORECAST table>` | OUT |
| 4 | `<STATISTICS table>` | OUT |

### Input Table

| Table | Column | Data Type | Description |
|---|---|---|---|
| DATA | 1st column | INTEGER | Time stamp |
| | 2nd column | INTEGER, DECIMAL(p, s), or DOUBLE | Raw data |

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| FORECAST_LENGTH | INTEGER | 1 | Length of the final forecast results. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| TREND | DOUBLE | 1 | Damped trend factor. Value range is (0,1]. | |
| AFFECT_FUTURE_ONLY | INTEGER | 1 | Specifies whether the damped trend affects the history.<br>• 0: Affects all.<br>• 1: Affects the future only. | Only valid when TREND is not 1. |
| SEASONALITY | INTEGER | 0 | Specifies whether the data represents seasonality.<br>• 0: Non-seasonality.<br>• 1: Seasonality exists and user inputs the value of periods.<br>• 2: Automatically detects seasonality. | |
| PERIODS | INTEGER | No default value | Length of the periods. | Only valid when SEASONALITY is 1.<br>If this parameter is not specified, the SEASONALITY value will be changed from 1 to 2, that is, from user-defined to automatically-detected. |
| MEASURE_NAME | VARCHAR | No default value | Measure name. Supported measures are MPE, MSE, RMSE, ET, MAD, MASE, WMAPE, SMAPE, and MAPE.<br>For detailed information on measures, see *Forecast Accuracy Measures*. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| SEASONAL_HANDLE_METHOD | INTEGER | 0 | Method used for calculating the index value in the periods.<br>• 0: Average method.<br>• 1: Fitting linear regression. | Only valid when SEASONALITY is 2. |
| EXPOST_FLAG | INTEGER | 1 | • 0: Does not output the expost forecast, and just outputs the forecast values.<br>• 1: Outputs the expost forecast and the forecast values. | |
| IGNORE_ZERO | INTEGER | 0 | • 0: Uses zero values in the input dataset when calculating MPE or MAPE.<br>• 1: Ignores zero values in the input dataset when calculating MPE or MAPE. | Only valid when MEASURE_NAME is MPE or MAPE. |

### Output Tables

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| FORECAST | 1st column | INTEGER | TIMESTAMP | Time stamp |
| | 2nd column | DOUBLE | VALUE | Forecasted value |
| STATISTICS | 1st column | NVARCHAR(100) | STAT_NAME | Statistics name |
| | 2nd column | DOUBLE | STAT_VALUE | Statistics value |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;

- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA DM_PAL;
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME " VARCHAR(100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (100)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FORECAST_LENGTH', 10,null,null);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('TREND', null,0.9,null);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('AFFECT_FUTURE_ONLY', 1,null,null);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SEASONALITY', 1,null,null);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PERIODS', 4,null,null);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MEASURE_NAME', null, null, 'MSE');
DROP TABLE PAL_FORECASTSLR_DATA_TBL;
CREATE COLUMN TABLE PAL_FORECASTSLR_DATA_TBL
(
    "TIMESTAMP" INTEGER,
    "Y" DOUBLE
);
INSERT INTO PAL_FORECASTSLR_DATA_TBL VALUES(1, 5384);
INSERT INTO PAL_FORECASTSLR_DATA_TBL VALUES(2, 8081);
INSERT INTO PAL_FORECASTSLR_DATA_TBL VALUES(3, 10282);
INSERT INTO PAL_FORECASTSLR_DATA_TBL VALUES(4, 9156);
INSERT INTO PAL_FORECASTSLR_DATA_TBL VALUES(5, 6118);
INSERT INTO PAL_FORECASTSLR_DATA_TBL VALUES(6, 9139);
INSERT INTO PAL_FORECASTSLR_DATA_TBL VALUES(7, 12460);
INSERT INTO PAL_FORECASTSLR_DATA_TBL VALUES(8, 10717);
INSERT INTO PAL_FORECASTSLR_DATA_TBL VALUES(9, 7825);
INSERT INTO PAL_FORECASTSLR_DATA_TBL VALUES(10, 9693);
INSERT INTO PAL_FORECASTSLR_DATA_TBL VALUES(11, 15177);
INSERT INTO PAL_FORECASTSLR_DATA_TBL VALUES(12, 10990);
CALL _SYS_AFL.PAL_LR_SEASONAL_ADJUST(PAL_FORECASTSLR_DATA_TBL,
#PAL_PARAMETER_TBL, ?,?);
```

**Expected Result**

FORECAST:

| | TIMESTAMP | VALUE |
|---|---|---|
| 1 | 1 | 5,328.171740542723 |
| 2 | 2 | 7,701.608246960384 |
| 3 | 3 | 11,248.606331576377 |
| 4 | 4 | 9,478.94478351929 |
| 5 | 5 | 6,138.471080754076 |
| 6 | 6 | 8,829.956471884489 |
| 7 | 7 | 12,838.389571679569 |
| 8 | 8 | 10,772.899796555095 |
| 9 | 9 | 6,948.77042096543 |
| 10 | 10 | 9,958.304696808595 |
| 11 | 11 | 14,428.172811782759 |
| 12 | 12 | 12,066.854809590903 |
| 13 | 13 | 7,738.812277671501 |
| 14 | 14 | 11,004.8476754257 |
| 15 | 15 | 15,794.988952461474 |
| 16 | 16 | 13,068.084849802683 |
| 17 | 17 | 8,303.102712446336 |
| 18 | 18 | 11,712.048670095626 |
| 19 | 19 | 16,691.75702236078 |
| 20 | 20 | 13,724.991879185629 |
| 21 | 21 | 8,673.333666702107 |
| 22 | 22 | 12,176.043242698564 |

STATISTICS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | Intercept | 7,626.072427992673 |
| 2 | Slope | 301.39911364215277 |
| 3 | Periods | 4 |
| 4 | Index0 | 0.6721148997583071 |
| 5 | Index1 | 0.9359253012467181 |
| 6 | Index2 | 1.3186694719271097 |
| 7 | Index3 | 1.0732903270678653 |
| 8 | MSE | 332,202.479081552... |
| 9 | HandleZero | 0 |

> **Note**
>
> The wrapper procedure generation approach to call PAL functions is still supported. For more information, refer to the older versions (prior to SAP HANA Platform 2.0 SPS 02) of this document.

## Related Information

Forecast Accuracy Measures [page 762]

# 5.5.21 Long-Term Series Forecasting

As is revealed from its name, this function is designed to solve a critical business demand that predicts values in the long future.

Although traditional algorithms are capable of predicting values in the near future, their performance will deteriorate greatly when it comes to long-term series forecasting. With the help of deep learning, this function implements a novel neural network architecture to achieve the state-of-the-art performance among the PAL family.

Particularly, this neural network employs the 1-dimensional convolution layer (Conv1d) and arranges them in a perfect binary tree architecture. Every tree node in the perfect binary tree accepts a time series as input, downsamples it by half in every even and every odd time period respectively, then processes it by convolution layer and other non-linear operations and finally output two time series.

Therefore, regarding the parameter selection, we must make sure that TRAIN_LENGTH is at least 2^{NUM_LEVELS}.

## Prerequisites

- No null data in the inputs. The time periods should be unique (no duplication) and equal sampling.

## _SYS_AFL.PAL_LTSF_TRAIN

This function trains the input time series and renders LOSS table that captures the training loss and MODEL table for prediction.

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<LOSS table>` | OUT |
| 4 | `<MODEL table>` | OUT |

### Input Table

| Table | Column | Data Type | Description |
|---|---|---|---|
| DATA | 1st column | INTEGER | Time stamp. Time stamp does not need to be in order, but must be unique and equal sampling. |
| | 2nd column | INTEGER, DOUBLE, or DECIMAL (p,s) | Values of target time series. |
| | Other columns | INTEGER, DOUBLE, or DECIMAL (p,s) | Values of exogenous features. |

### Parameter Table

During training, the network will first be given a time series of a user-defined length and then predict the following time series with another length.

Mandatory Parameters

| Name | Data Type | Description | Dependency |
|---|---|---|---|
| TRAIN_LENGTH | INTEGER | Length of training series inputted to the network. | |
| FORECAST_LENGTH | INTEGER | Length of predictions. | TRAIN_LENGTH + FORECAST_LENGTH <= DATA_LENGTH |

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| NUM_LEVELS | INTEGER | 2 | Number of levels within the tree network architecture. | |
| KERNEL_SIZE | INTEGER | 3 | Kernel size of Conv1d layer. | |
| HIDDEN_EXPANSION | INTEGER | 3 | Expands the input channel size of Conv1d layer: where DATA_DIM is the number of external features plus 1. | |
| BATCH_SIZE | INTEGER | 16 | Number of training series in a batch. | |
| NUM_EPOCHS | INTEGER | 1 | Number of training epochs. | |
| RANDOM_SEED | INTEGER | 0 | 0 indicates using machine time as seed. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| POSITION_ENCODING | INTEGER | 1 | Position encoding adds extra positional embeddings to the training series:<br>• 0: Do not use<br>• 1: Use | |
| ADJUST_LEARNING_RATE | INTEGER | 0 | Decays the learning rate to its half after every epoch:<br>• 0: Do not use<br>• 1: Use | |
| LEARNING_RATE | DOUBLE | 0.005 | Learning rate for Adam optimizer. | |
| DROPOUT_PROB | DOUBLE | 0.2 | Dropout probability of Dropout layer. | |

### Output Table

| Table | Column | Data Type | Column Name | Description | Constraint |
|---|---|---|---|---|---|
| LOSS | 1st column | INTEGER | EPOCH | Indicates the current epoch. | |
| | 2nd column | NVARCHAR | BATCH | Indicates the information of training loss either batch ID or average batch loss indicator. | |
| | 3rd column | DOUBLE | LOSS | Indicates the training loss. | |
| MODEL | 1st column | INTEGER | ROW_INDEX | Indicates the id of current row. | |
| | 2nd column | NVARCHAR | MODEL_CONTENT | JSON string. | |

### Example: LTSF train

Assume that:

- DM_PAL is a schema belonging to USER1;
- USER1 has been assigned the AFL__SYS_AFL_AFLPAL_EXECUTE or AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION role.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_LTSF_DATA_TBL;
CREATE COLUMN TABLE PAL_LTSF_DATA_TBL (
    TIME_STAMP INTEGER,
    TARGET DOUBLE,
    FEAT1 INTEGER,
    FEAT2 DOUBLE,
```

```
    FEAT3 VARCHAR(2),
    FEAT4 DOUBLE
);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (1,  129.0, -16,  -4.0, 'SE', 1.79);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (2,  148.0, -15,  -4.0, 'SE', 2.68);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (3,  159.0, -11,  -5.0, 'SE', 3.57);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (4,  181.0,  -7,  -5.0, 'SE', 5.36);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (5,  138.0,  -7,  -5.0, 'SE', 6.25);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (6,  109.0,  -7,  -6.0, 'SE', 7.14);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (7,  105.0,  -7,  -6.0, 'SE', 8.93);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (8,  124.0,  -7,  -5.0, 'SE', 10.72);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (9,  120.0,  -8,  -6.0, 'SE', 12.51);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (10, 132.0,  -7,  -5.0, 'SE', 14.3);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (11, 140.0,  -7,  -5.0, 'SE', 17.43);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (12, 152.0,  -8,  -5.0, 'SE', 20.56);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (13, 148.0,  -8,  -5.0, 'SE', 23.69);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (14, 164.0,  -8,  -5.0, 'SE', 27.71);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (15, 158.0,  -9,  -5.0, 'SE', 31.73);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (16, 154.0,  -9,  -5.0, 'SE', 35.75);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (17, 159.0,  -9,  -5.0, 'SE', 37.54);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (18, 164.0,  -8,  -5.0, 'SE', 39.33);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (19, 170.0,  -8,  -5.0, 'SE', 42.46);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (20, 149.0,  -8,  -5.0, 'SE', 44.25);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (21, 154.0,  -7,  -5.0, 'SE', 46.04);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (22, 164.0,  -7,  -5.0, 'SE', 49.17);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (23, 156.0,  -8,  -6.0, 'SE', 52.3);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (24, 126.0,  -8,  -6.0, 'SE', 55.43);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (25,  90.0,  -7,  -6.0, 'SE', 58.56);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (26,  63.0,  -8,  -6.0, 'SE', 61.69);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (27,  65.0,  -8,  -7.0, 'SE', 65.71);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (28,  55.0,  -8,  -7.0, 'SE', 68.84);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (29,  65.0,  -8,  -7.0, 'SE', 72.86);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (30,  83.0,  -9,  -8.0, 'SE', 76.88);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (31,  91.0, -10,  -8.0, 'SE', 80.9);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (32,  86.0, -10,  -9.0, 'SE', 84.92);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (33,  82.0, -10,  -9.0, 'SE', 89.84);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (34,  86.0, -11,  -9.0, 'SE', 93.86);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (35,  78.0, -11,  -9.0, 'SE', 97.88);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (36,  98.0, -11,  -9.0, 'SE', 102.8);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (37, 107.0, -11,  -9.0, 'SE', 105.93);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (38,  90.0, -11,  -9.0, 'SE', 111.74);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (39,  96.0, -11,  -9.0, 'SE', 116.66);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (40,  95.0, -11,  -9.0, 'SE', 121.58);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (41,  86.0, -11,  -9.0, 'SE', 124.71);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (42,  70.0, -11,  -9.0, 'SE', 127.84);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (43,  61.0, -11,  -9.0, 'cv', 0.89);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (44,  53.0, -11,  -9.0, 'cv', 1.78);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (45,  71.0, -10,  -9.0, 'NW', 4.02);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (46,  72.0, -11, -10.0, 'NW', 7.15);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (47,  76.0, -11,  -9.0, 'NW', 11.17);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (48,  73.0, -12, -11.0, 'NW', 14.3);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (49,  79.0, -14, -12.0, 'NW', 16.09);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (50,  58.0, -16,  -9.0, 'NW', 21.9);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (51,  25.0, -17, -10.0, 'NW', 29.95);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (52,  26.0, -18, -11.0, 'NW', 39.78);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (53,  28.0, -19, -11.0, 'NW', 48.72);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (54,  26.0, -20, -12.0, 'NW', 55.87);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (55,  20.0, -21, -12.0, 'NW', 64.81);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (56,  29.0, -21, -13.0, 'NW', 73.75);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (57,  26.0, -22, -13.0, 'NW', 80.9);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (58,  27.0, -22, -13.0, 'NW', 90.73);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (59,  27.0, -22, -12.0, 'NW', 100.56);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (60,  25.0, -23, -12.0, 'NW', 108.61);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (61,  29.0, -21, -11.0, 'NW', 117.55);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (62,  32.0, -20, -10.0, 'NW', 127.38);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (63,  28.0, -21, -10.0, 'NW', 136.32);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (64,  29.0, -21,  -9.0, 'NW', 145.26);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (65,  30.0, -21,  -9.0, 'NW', 152.41);
```

```
INSERT INTO PAL_LTSF_DATA_TBL VALUES (66,  30.0, -20, -11.0, 'NW', 159.56);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (67,  28.0, -23, -11.0, 'NW', 165.37);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (68,  26.0, -21, -12.0, 'NW', 171.18);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (69,  31.0, -24, -12.0, 'NW', 180.12);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (70,  33.0, -24, -13.0, 'NW', 187.27);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (71,  29.0, -24, -13.0, 'NW', 195.32);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (72,  31.0, -26, -15.0, 'NW', 198.45);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (73,  30.0, -26, -17.0, 'NW', 201.58);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (74,  34.0, -26, -18.0, 'NW', 205.6);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (75,  27.0, -26, -19.0, 'NW', 208.73);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (76,  25.0, -27, -18.0, 'NW', 213.65);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (77,  28.0, -27, -19.0, 'NW', 218.57);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (78,  28.0, -27, -16.0, 'NE', 4.92);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (79,  27.0, -26, -16.0, 'NE', 8.05);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (80,  27.0, -27, -16.0, 'NE', 13.86);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (81,  27.0, -26, -16.0, 'NE', 18.78);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (82,  29.0, -26, -15.0, 'NE', 24.59);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (83,  36.0, -25, -14.0, 'NE', 29.51);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (84,  30.0, -25, -13.0, 'NE', 34.43);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (85,  27.0, -25, -12.0, 'NE', 39.35);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (86,  39.0, -24, -11.0, 'NE', 41.14);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (87,  41.0, -22, -11.0, 'cv', 0.89);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (88,  33.0, -23, -11.0, 'NW', 1.79);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (89,  50.0, -24, -11.0, 'NW', 3.58);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (90,  56.0, -23, -11.0, 'NW', 5.37);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (91,  59.0, -23, -11.0, 'NW', 7.16);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (92,  60.0, -22, -13.0, 'NW', 10.29);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (93,  84.0, -22, -12.0, 'NW', 13.42);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (94, 106.0, -24, -18.0, 'NW', 16.55);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (95,  66.0, -22, -13.0, 'NW', 20.57);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (96,  50.0, -22, -16.0, 'NW', 23.7);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (97,  56.0, -25, -17.0, 'NW', 26.83);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (98,  77.0, -25, -14.0, 'NE', 4.02);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (99,  50.0, -26, -14.0, 'NE', 8.04);
INSERT INTO PAL_LTSF_DATA_TBL VALUES (100, 44.0, -26, -14.0, 'NE', 13.85);
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL (PARAM_NAME VARCHAR(256), INT_VALUE
INTEGER, DOUBLE_VALUE DOUBLE, STRING_VALUE VARCHAR(1000));
INSERT INTO PAL_PARAMETER_TBL VALUES ('TRAIN_LENGTH',        32,    Null,
Null);
INSERT INTO PAL_PARAMETER_TBL VALUES ('FORECAST_LENGTH',     16,    Null,
Null);
INSERT INTO PAL_PARAMETER_TBL VALUES ('NUM_LEVELS',           2,    Null,
Null);
INSERT INTO PAL_PARAMETER_TBL VALUES ('KERNEL_SIZE',          3,    Null,
Null);
INSERT INTO PAL_PARAMETER_TBL VALUES ('HIDDEN_EXPANSION',     2,    Null,
Null);
INSERT INTO PAL_PARAMETER_TBL VALUES ('BATCH_SIZE',           8,    Null,
Null);
INSERT INTO PAL_PARAMETER_TBL VALUES ('NUM_EPOCHS',           2,    Null,
Null);
INSERT INTO PAL_PARAMETER_TBL VALUES ('POSITION_ENCODING',    1,    Null,
Null);
INSERT INTO PAL_PARAMETER_TBL VALUES ('ADJUST_LEARNING_RATE', 1,    Null,
Null);
INSERT INTO PAL_PARAMETER_TBL VALUES ('RANDOM_SEED',          1,    Null,
Null);
INSERT INTO PAL_PARAMETER_TBL VALUES ('DROPOUT_PROB',      Null,     0.2,
Null);
INSERT INTO PAL_PARAMETER_TBL VALUES ('LEARNING_RATE',     Null,   0.005,
Null);
DROP TABLE PAL_LTSF_MODEL_TBL;
CREATE COLUMN TABLE PAL_LTSF_MODEL_TBL (
    ROW_INDEX INTEGER,
    MODEL_CONTENT NVARCHAR(5000)
);
DROP TABLE PAL_LTSF_LOSS_TBL;
```

```
CREATE COLUMN TABLE PAL_LTSF_LOSS_TBL (
    EPOCH INTEGER,
    BATCH NVARCHAR(5000),
    LOSS DOUBLE
);
DO BEGIN
    lt_data = SELECT * FROM PAL_LTSF_DATA_TBL;
    lt_param = SELECT * FROM PAL_PARAMETER_TBL;
    CALL _SYS_AFL.PAL_LTSF_TRAIN (:lt_data, :lt_param, lt_loss, lt_model);
    INSERT INTO PAL_LTSF_LOSS_TBL SELECT * FROM :lt_loss;
    INSERT INTO PAL_LTSF_MODEL_TBL SELECT * FROM :lt_model;
END;
select * from PAL_LTSF_LOSS_TBL;
select * from PAL_LTSF_MODEL_TBL;
```

### Expected Result

Loss table

| | EPOCH | BATCH | LOSS |
|---|---|---|---|
| 1 | 1 | 0 | 1.1631603647059074 |
| 2 | 1 | 1 | 0.9376457470358657 |
| 3 | 1 | 2 | 0.8689782025981028 |
| 4 | 1 | 3 | 0.7431656046633617 |
| 5 | 1 | 4 | 0.7415275558287107 |
| 6 | 1 | 5 | 0.7238949275668671 |
| 7 | 1 | epoch average | 0.8630620670664692 |
| 8 | 2 | 0 | 0.7055796168272005 |
| 9 | 2 | 1 | 0.6629633901304192 |
| 10 | 2 | 2 | 0.6494527196792508 |
| 11 | 2 | 3 | 0.6556326078813421 |
| 12 | 2 | 4 | 0.662661142774968 |
| 13 | 2 | 5 | 0.595311241802508 |
| 14 | 2 | epoch average | 0.6552667865159482 |

Model table

| | ROW_INDEX | |
|---|---|---|
| 1 | 0 | {"len_train":32,"len_pred":16,"pos_enc":1,"PreprocessArgs1":[73.93,-16.4,-10.1,63.8583],"PreprocessArgs2":[46.99771378418268,7.266 |
| 2 | 1 | 8626,0.08171352846088237,0.17396134441096348,0.24080647829960787,-0.14446713460616915,0.07269783617601952,-0.09911 |
| 3 | 2 | 74264809353,-0.05710789790111052,0.03233696508363507,0.006406575176847177,-0.15593301723371856,0.0122127288518512 |
| 4 | 3 | 55,0.12368384437944057,0.03851080517586198,0.022604843171718775,0.1962622664829714,-0.1496420894063394,-0.23284280 |
| 5 | 4 | 1155087,0.09222282556125916,-0.10084095910186589,-0.033248422827601157,-0.1521724105100547,0.022280060756972497,0.0 |
| 6 | 5 | 10367587,0.0878305671075838,0.12251355597254909,-0.0869224207884401,0.10815999574522585,-0.06364455295868231,-0.173 |
| 7 | 6 | 37475,-0.14416977822564018,-0.13758026550107836,0.09833027469237003,-0.0946126627574058,-0.06628482840183446,-0.137 |
| 8 | 7 | 39954016,0.03902895218083014,0.0988449587948664,0.13468123588193466,-0.07605268325177718,-0.1492123850527865,-0.112 |
| 9 | 8 | 0837934116935,0.05223243234934812,0.13138498962482235,-0.1440158045441907,0.05419541260627 4608,0.08557097511493,0. |
| 10 | 9 | 22018502,0.033158655536090337,0.14742316208425294,-0.11456029016396534,0.14293917273324559,0.14295798983447914,0.1 |
| 11 | 10 | 083592261409617,-0.16453758973334726,-0.039963051348484397,-0.16025532046775843,0.08392061802978777,0.07719388638 |
| 12 | 11 | 6493,0.11010028805331165,0.06775038997129934,0.07916674054394829,-0.1004373379912471,-0.16123091342023727,-0.160612 |
| 13 | 12 | 0.06125476430652217,-0.009799836135102099,-0.12021399231928132,0.05873093989551508,0.03957126596115618,-0.15807945 |
| 14 | 13 | 01,0.05836047072963304,-0.1493705113332568,0.2002792721385911,-0.069689560080542,0.07218242499735694,0.05885318519 |
| 15 | 14 | 5,0.12408190945715038,-0.1514412469554849,-0.11218324084896757,0.19244555347274479,0.12272866126870963,0.209845658 |
| 16 | 15 | 071,-0.04877962967475525,-0.008185552989658103,-0.08422696108983291,-0.1294410340995935,0.05300935463058244,-0.0544 |
| 17 | 16 | 004,-0.1318736707965833,0.022866901409838017,-0.09784033122818332,-0.1310729717170202,0.12431983019317969,0.0389000 |
| 18 | 17 | 267,-0.0653659020006048,0.14021229267824626,-0.06496105906759524,-0.04550805556906706,-0.0000945118182493006,0.0909 |
| 19 | 18 | 77428055,-0.21919827670996049,-0.0030302808632884216,0.20895780343108517,-0.07747610568991569,-0.20242153843045586 |
| 20 | 19 | 40170254,-0.024458673826165568,-0.13243060008814099,-0.004257167656276 6949,0.14825262536428555,0.10255817747033626 |
| 21 | 20 | 79748443,-0.07993691243024877,0.12463706373242913,0.06288578413028429,-0.029947441862203825,0.01794929809101 1705, |

# _SYS_AFL.PAL_LTSF_TRAIN_CONTINUE

This function is used to train the input time series continuously given a pre-trained model table, which is meant to help user monitor the training process.

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<IN_MODEL table>` | IN |
| 3 | `<PARAMETER table>` | IN |
| 4 | `<LOSS table>` | OUT |
| 5 | `<OUT_MODEL table>` | OUT |

### Input Table

Features in DATA table must be exactly the same as those from _SYS_AFL.PAL_LTSF_TRAIN.

| Table | Column | Data Type | Description |
|---|---|---|---|
| DATA | 1st column | INTEGER | Time stamp. Time stamp does not need to be in order, but must be unique and equal sampling. |
| | 2nd column | INTEGER, DOUBLE, or DECIMAL (p,s) | Values of target time series. |
| | Other columns | INTEGER, DOUBLE, or DECIMAL (p,s) | Values of exogenous features. |
| IN_MODEL | 1st column | INTEGER | ROW_INDEX |
| | 2nd column | NVARCHAR | MODEL_CONTENT |

### Parameter Table

Network structure cannot be changed in continuous training and its information is included in IN_MODEL table, hence several parameters regarding the structure from _SYS_AFL.PAL_LTSF_TRAIN are disabled, while others are all the same.

Mandatory Parameters

None

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| BATCH_SIZE | INTEGER | 16 | Number of training series in a batch. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| NUM_EPOCHS | INTEGER | 1 | Number of training epochs. | |
| RANDOM_SEED | INTEGER | 0 | 0 indicates using machine time as seed. | |
| POSITION_ENCODING | INTEGER | 1 | Position encoding adds extra positional embeddings to the training series:<br>• 0: Do not use<br>• 1: Use | |
| ADJUST_LEARNING_RATE | INTEGER | 0 | Decays the learning rate after every epoch:<br>• 0: Do not use<br>• 1: Use | |
| LEARNING_RATE | DOUBLE | 0.005 | Learning rate for Adam optimizer. | |
| DROPOUT_PROB | DOUBLE | 0.2 | Dropout probability of Dropout layer. | |

### Output Table

| Table | Column | Data Type | Column Name | Description | Constraint |
|---|---|---|---|---|---|
| LOSS | 1st column | INTEGER | EPOCH | Indicates the current epoch. | |
| | 2nd column | NVARCHAR | BATCH | Indicates the information of training loss either batch ID or average batch loss indicator | |
| | 3rd column | DOUBLE | LOSS | Indicates the training loss. | |
| OUT_MODEL | 1st column | INTEGER | ROW_INDEX | Indicates the id of current row. | |
| | 2nd column | NVARCHAR | MODEL_CONTENT | JSON string. | |

### Example: LTSF continuous train

Assume that:

- DM_PAL is a schema belonging to USER1;
- USER1 has been assigned the AFL__SYS_AFL_AFLPAL_EXECUTE or AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION role.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_PARAMETER_TBL;
```

```
CREATE COLUMN TABLE PAL_PARAMETER_TBL (PARAM_NAME VARCHAR(256), INT_VALUE
INTEGER, DOUBLE_VALUE DOUBLE, STRING_VALUE VARCHAR(1000));
INSERT INTO PAL_PARAMETER_TBL VALUES ('BATCH_SIZE',           4,    Null,
Null);
INSERT INTO PAL_PARAMETER_TBL VALUES ('NUM_EPOCHS',           2,    Null,
Null);
INSERT INTO PAL_PARAMETER_TBL VALUES ('POSITION_ENCODING',    1,    Null,
Null);
INSERT INTO PAL_PARAMETER_TBL VALUES ('ADJUST_LEARNING_RATE', 1,    Null,
Null);
INSERT INTO PAL_PARAMETER_TBL VALUES ('RANDOM_SEED',          1,    Null,
Null);
INSERT INTO PAL_PARAMETER_TBL VALUES ('DROPOUT_PROB',      Null,    0.15,
Null);
INSERT INTO PAL_PARAMETER_TBL VALUES ('LEARNING_RATE',     Null,   0.002,
Null);
DROP TABLE PAL_LTSF_NEW_MODEL_TBL;
CREATE COLUMN TABLE PAL_LTSF_NEW_MODEL_TBL (
    ROW_INDEX INTEGER,
    MODEL_CONTENT NVARCHAR(5000)
);
DROP TABLE PAL_LTSF_LOSS_TBL;
CREATE COLUMN TABLE PAL_LTSF_LOSS_TBL (
    EPOCH INTEGER,
    BATCH NVARCHAR(5000),
    LOSS DOUBLE
);
DO BEGIN
    lt_data     = SELECT * FROM PAL_LTSF_DATA_TBL;
    lt_in_model = SELECT * FROM PAL_LTSF_MODEL_TBL;
    lt_param    = SELECT * FROM PAL_PARAMETER_TBL;
    CALL _SYS_AFL.PAL_LTSF_TRAIN_CONTINUE (:lt_data, :lt_in_model, :lt_param,
lt_loss, lt_out_model);
    INSERT INTO PAL_LTSF_LOSS_TBL SELECT * FROM :lt_loss;
    INSERT INTO PAL_LTSF_NEW_MODEL_TBL SELECT * FROM :lt_out_model;
END;
select * from PAL_LTSF_LOSS_TBL;
select * from PAL_LTSF_NEW_MODEL_TBL;
```

**Expected Result**

New loss table:

| | EPOCH | BATCH | LOSS |
|---|---|---|---|
| 1 | 1 | 0 | 0.562973068808559 |
| 2 | 1 | 1 | 0.6950602551781587 |
| 3 | 1 | 2 | 0.5473561849012835 |
| 4 | 1 | 3 | 0.6201819306787328 |
| 5 | 1 | 4 | 0.5544251088450898 |
| 6 | 1 | 5 | 0.6420558493660996 |
| 7 | 1 | 6 | 0.5315466524770641 |
| 8 | 1 | 7 | 0.5371194515038508 |
| 9 | 1 | 8 | 0.5592406390060203 |
| 10 | 1 | 9 | 0.482765733559522 |
| 11 | 1 | 10 | 0.5212030046816695 |
| 12 | 1 | 11 | 0.6051549236764637 |
| 13 | 1 | 12 | 0.5692462569242404 |
| 14 | 1 | epoch average | 0.5714099276620581 |
| 15 | 2 | 0 | 0.5515185777530889 |
| 16 | 2 | 1 | 0.47890513341249796 |
| 17 | 2 | 2 | 0.5133451129045044 |
| 18 | 2 | 3 | 0.47068671887058383 |
| 19 | 2 | 4 | 0.4583889474094458 |
| 20 | 2 | 5 | 0.5136831369299472 |
| 21 | 2 | 6 | 0.5064748325427416 |
| 22 | 2 | 7 | 0.45242198102818254 |
| 23 | 2 | 8 | 0.5611249363677503 |
| 24 | 2 | 9 | 0.4191249967073958 |
| 25 | 2 | 10 | 0.4964501040397966 |
| 26 | 2 | 11 | 0.48515391556080056 |
| 27 | 2 | 12 | 0.5302392745888665 |
| 28 | 2 | epoch average | 0.4951936667781232 |

New model table:

| | ROW_INDEX | |
|---|---|---|
| 1 | 0 | {"len_train":32,"len_pred":16,"pos_enc":1,"PreprocessArgs1":[73.93,-16.4,-10.1,63.8583],"PreprocessArgs2":[46.99771378418268,7.266 |
| 2 | 1 | 037211663125206,0.04724112632845442,0.17446821096252847,0.25134928092899819,-0.1314571148557803,0.067214691673732 |
| 3 | 2 | 732475,0.01378858221397539,-0.05742626784728651,0.03616571390000677,-0.021079335733485875,-0.17818813567262746,0.0( |
| 4 | 3 | 93647,0.014239199041806166,0.13059372457090108,0.04118086869979658,0.017340062882020666,0.2041736274330916,-0.1519 |
| 5 | 4 | 2243038,0.07923702273536831,-0.07396548425284085,-0.024262803807942516,-0.14055918851673616,-0.0008522841698432348 |
| 6 | 5 | 6,0.07648497360885198,0.12604496081240158,-0.08890726323145656,0.097851583234684,-0.06810179256544175,-0.164273113( |
| 7 | 6 | 6986639902241544,-0.13858123445331434,0.09598076001890746,-0.12882250376896024,-0.04347791018223878,-0.14174333080 |
| 8 | 7 | 65305435310526,0.11253917362145134,0.15829722673062147,-0.08186808762016899,-0.11921255569671581,-0.09712892982155 |
| 9 | 8 | -0.032713801480713417,0.04759519583629039,0.13595391185204634,-0.14716536809240075,0.04368469501409266,0.097336910 |
| 10 | 9 | 3890795256812,0.11005819309797281,0.03355354451832425,0.1664296432013677,-0.08622996840110063,0.1420495957409465, |
| 11 | 10 | 514240387469823,-0.022734034358571438,-0.1789694293659162,-0.04891472375126582,-0.19454523598590998,0.087711022307 |
| 12 | 11 | 327327,0.0710281321067872,0.12444597569211087,0.09000675093292468,0.06746606165820737,-0.10481387199916444,-0.1331 |
| 13 | 12 | 7009175,-0.07721818410665075,-0.037999887019195859,-0.126421579839849,0.02556190946312814,0.05095546308632861,-0.17 |
| 14 | 13 | 29,0.03896454061320169,-0.1688250223337272,0.23072891208424049,-0.06844921164720369,0.07100828722475795,0.04236857 |
| 15 | 14 | 030683785835613,-0.14227900335480082,-0.11484598255542009,0.169588325389507,0.12222919936266037,0.236330724561779 |
| 16 | 15 | 3870647749,-0.08674480713641296,-0.1200885444291275,0.0349699002242796,-0.04453865747401785,0.19111738187337774,-0. |
| 17 | 16 | 0.08821574935303767,-0.15241830260128268,0.11024951710112174,0.05086966338018771,0.05254522952504415,-0.077635969 |
| 18 | 17 | 2902948,-0.07556042359129381,-0.03985341147256471 8,0.014918155275451296,0.0887379349886616,0.09758190723504995,0.0 |
| 19 | 18 | 46284038964,-0.07467111310991609,-0.23634384089303668,-0.010856820279152585,-0.1113077186118028,0.1649060824567315 |
| 20 | 19 | 084019247,-0.024847843400097035,0.1393346072111151,0.09076249628381172,-0.04330945577347291,-0.0104958488377898,-0. |
| 21 | 20 | 8,-0.06411382469987134,0.04435276088828448 9,-0.08478443342749 81,0.00174094994510719 56,0.15077273553841 48,-0.0524956 |

# _SYS_AFL.PAL_LTSF_PREDICT

This function generates predictions given the input time series.

Particularly, the format of the input data should be the same as every series inputted into the network in the training period, which means the number of rows of DATA table must be equal to the value of TRAIN_LENGTH during training and the length of predicitons will be equal to the value of FORECAST_LENGTH afterwards.

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<MODEL table>` | IN |
| 3 | `<PARAMETER table>` | IN |
| 4 | `<RESULT table>` | OUT |

### Input Table

Features in DATA table must be exactly the same as those from _SYS_AFL.PAL_LTSF_TRAIN.

| Table | Column | Data Type | Description |
|---|---|---|---|
| DATA | 1st column | INTEGER | Time stamp. Time stamp does not need to be in order, but must be unique and equal sampling. |
| | 2nd column | INTEGER, DOUBLE, or DECIMAL(p,s) | Values of target time series. |
| | Other columns | INTEGER, DOUBLE, or DECIMAL(p,s) | Values of exogenous features. |
| MODEL | 1st column | INTEGER | ROW_INDEX |
| | 2nd column | NVARCHAR | MODEL_CONTENT |

### Parameter Table

Mandatory Parameters

None

Optional Parameters

None

### Output Table

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| FORECAST | 1st column | INTEGER | ID | Time stamp that is monotonically increasing sorted. |
| | 2nd column | DOUBLE | FORECAST | Forecast value. |

### Example: LTSF predict

Assume that:

- DM_PAL is a schema belonging to USER1;
- USER1 has been assigned the AFL__SYS_AFL_AFLPAL_EXECUTE or AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION role.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_LTSF_PRED_DATA_TBL;
CREATE COLUMN TABLE PAL_LTSF_PRED_DATA_TBL (
    TIME_STAMP INTEGER,
    TARGET DOUBLE,
    FEAT1 INTEGER,
    FEAT2 DOUBLE,
    FEAT3 VARCHAR(2),
    FEAT4 DOUBLE
);
INSERT INTO PAL_LTSF_PRED_DATA_TBL VALUES (69,  31.0, -24, -12.0, 'NW', 180.12);
INSERT INTO PAL_LTSF_PRED_DATA_TBL VALUES (70,  33.0, -24, -13.0, 'NW', 187.27);
INSERT INTO PAL_LTSF_PRED_DATA_TBL VALUES (71,  29.0, -24, -13.0, 'NW', 195.32);
INSERT INTO PAL_LTSF_PRED_DATA_TBL VALUES (72,  31.0, -26, -15.0, 'NW', 198.45);
INSERT INTO PAL_LTSF_PRED_DATA_TBL VALUES (73,  30.0, -26, -17.0, 'NW', 201.58);
INSERT INTO PAL_LTSF_PRED_DATA_TBL VALUES (74,  34.0, -26, -18.0, 'NW', 205.6);
INSERT INTO PAL_LTSF_PRED_DATA_TBL VALUES (75,  27.0, -26, -19.0, 'NW', 208.73);
INSERT INTO PAL_LTSF_PRED_DATA_TBL VALUES (76,  25.0, -27, -18.0, 'NW', 213.65);
INSERT INTO PAL_LTSF_PRED_DATA_TBL VALUES (77,  28.0, -27, -19.0, 'NW', 218.57);
INSERT INTO PAL_LTSF_PRED_DATA_TBL VALUES (78,  28.0, -27, -16.0, 'NE', 4.92);
INSERT INTO PAL_LTSF_PRED_DATA_TBL VALUES (79,  27.0, -26, -16.0, 'NE', 8.05);
INSERT INTO PAL_LTSF_PRED_DATA_TBL VALUES (80,  27.0, -27, -16.0, 'NE', 13.86);
INSERT INTO PAL_LTSF_PRED_DATA_TBL VALUES (81,  27.0, -26, -16.0, 'NE', 18.78);
INSERT INTO PAL_LTSF_PRED_DATA_TBL VALUES (82,  29.0, -26, -15.0, 'NE', 24.59);
INSERT INTO PAL_LTSF_PRED_DATA_TBL VALUES (83,  36.0, -25, -14.0, 'NE', 29.51);
INSERT INTO PAL_LTSF_PRED_DATA_TBL VALUES (84,  30.0, -25, -13.0, 'NE', 34.43);
INSERT INTO PAL_LTSF_PRED_DATA_TBL VALUES (85,  27.0, -25, -12.0, 'NE', 39.35);
INSERT INTO PAL_LTSF_PRED_DATA_TBL VALUES (86,  39.0, -24, -11.0, 'NE', 41.14);
INSERT INTO PAL_LTSF_PRED_DATA_TBL VALUES (87,  41.0, -22, -11.0, 'cv', 0.89);
INSERT INTO PAL_LTSF_PRED_DATA_TBL VALUES (88,  33.0, -23, -11.0, 'NW', 1.79);
INSERT INTO PAL_LTSF_PRED_DATA_TBL VALUES (89,  50.0, -24, -11.0, 'NW', 3.58);
INSERT INTO PAL_LTSF_PRED_DATA_TBL VALUES (90,  56.0, -23, -11.0, 'NW', 5.37);
INSERT INTO PAL_LTSF_PRED_DATA_TBL VALUES (91,  59.0, -23, -11.0, 'NW', 7.16);
INSERT INTO PAL_LTSF_PRED_DATA_TBL VALUES (92,  60.0, -22, -13.0, 'NW', 10.29);
INSERT INTO PAL_LTSF_PRED_DATA_TBL VALUES (93,  84.0, -22, -12.0, 'NW', 13.42);
INSERT INTO PAL_LTSF_PRED_DATA_TBL VALUES (94, 106.0, -24, -18.0, 'NW', 16.55);
INSERT INTO PAL_LTSF_PRED_DATA_TBL VALUES (95,  66.0, -22, -13.0, 'NW', 20.57);
INSERT INTO PAL_LTSF_PRED_DATA_TBL VALUES (96,  50.0, -22, -16.0, 'NW', 23.7);
INSERT INTO PAL_LTSF_PRED_DATA_TBL VALUES (97,  56.0, -25, -17.0, 'NW', 26.83);
INSERT INTO PAL_LTSF_PRED_DATA_TBL VALUES (98,  77.0, -25, -14.0, 'NE', 4.02);
INSERT INTO PAL_LTSF_PRED_DATA_TBL VALUES (99,  50.0, -26, -14.0, 'NE', 8.04);
INSERT INTO PAL_LTSF_PRED_DATA_TBL VALUES (100, 44.0, -26, -14.0, 'NE', 13.85);
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL ("PARAM_NAME" VARCHAR(256), "INT_VALUE"
INTEGER, "DOUBLE_VALUE" DOUBLE, "STRING_VALUE" VARCHAR(1000));
CALL _SYS_AFL.PAL_LTSF_PREDICT(PAL_LTSF_PRED_DATA_TBL, PAL_LTSF_MODEL_TBL,
PAL_PARAMETER_TBL, ?);
```

### Expected Result

Result table

| | ID | FORECAST |
|---|---|---|
| 1 | 0 | 51.9069685921464 |
| 2 | 1 | 58.554727230420134 |
| 3 | 2 | 70.31251265850798 |
| 4 | 3 | 69.40523095532441 |
| 5 | 4 | 41.95996570250268 |
| 6 | 5 | 57.52291471856539 |
| 7 | 6 | 53.436433886263295 |
| 8 | 7 | 40.23025691599181 |
| 9 | 8 | 28.19935809787582 |
| 10 | 9 | 101.12128907985966 |
| 11 | 10 | 133.77559104170402 |
| 12 | 11 | 47.21086099800379 |
| 13 | 12 | 46.16222998205511 |
| 14 | 13 | 70.75296524457373 |
| 15 | 14 | 66.04472332430893 |
| 16 | 15 | 71.15564651337542 |

## 5.5.22 Online ARIMA

Online ARIMA implements an online learning method to estimate the parameters of ARIMA models by reformulating it into a full information online optimization task (without random noise terms), which has no limitations of depending on noise terms and accessing the entire large-scale dataset in advance.

A time series $y_t$ described by ARIMA (k, d, q) model could be approximated as

$$y_t = \sum_{i=1}^{p+m} \phi_i \nabla^d y_{t-i} + \sum_{i=0}^{d-1} \nabla^i y_{t-1},$$

in online ARIMA (p+m, d, 0) model.

This implementation is based on Online Newton Step (ONS) optimization.

Model arguments are listed as follows:

| Key Name | Description |
|---|---|
| mp | Number of extended AR terms |
| ar | Coefficients of extended AR terms |
| d | Degree of first differencing |
| dy | Tailing values of zero and high order differencing values in last call. |
| epsilon | Convergence criteria |
| lrate | Learning rate |

| Key Name | Description |
| --- | --- |
| fitted | Should fittings be included in output |

## Prerequisite

- The input data must not have missing value.
- The ID column must not have any identical value.

## _SYS_AFL.PAL_ONLINE_ARIMA

This function generates Online ARIMA models with given orders.

### Overview of All Tables

| Position | Table Name | Parameter Type |
| --- | --- | --- |
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<INPUT-MODEL table>` | IN |
| 4 | `<OUTPUT-MODEL table>` | OUT |
| 5 | `<FIT table>` | OUT |

### Input Table

| Table | Column | Reference for Output Table | Data Type | Description |
| --- | --- | --- | --- | --- |
| DATA | 1st column | ID | INTEGER | Time stamp.<br>No identical value is allowed. |
| | 2nd column | | INTEGER, DOUBLE, or DECIMAL(p, s) | Raw data |

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value. The parameter table takes effect only when the input-model table is empty.

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| P | INTEGER | 1 | Value of auto regression (AR) order |
| M | INTEGER | 0 | Extended order needed to transform all moving-averaging term to AR term. |
| D | INTEGER | 0 | Value of the differentiation order |
| OUTPUT_FITTED | INTEGER | 0 | If output fitted:<br>• 0: not output<br>• 1: output |
| LEARNING_RATE | DOUBLE | Calculated according to P, M and D. | Learning rate. Must be greater than zero. |
| EPSILON | DOUBLE | Calculated according to LEARNING_RATE, P and M. | Convergence criterion. |

### Input-model Table

The model outputted in last running. If it is the first call, just keep this model empty.

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| INPUT-MODEL | 1st column | NVARCHAR(100) | KEY | Name |
| | 2nd column | NVARCHAR(5000) | VALUE | Value |

### Output Tables

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| OUPUT-MODEL | 1st column | NVARCHAR(100) | KEY | Name |
| | 2nd column | NVARCHAR(5000) | VALUE | Value |
| FIT | 1st column | ID (in input table) column name | ID (in input table) column name | Time stamp |
| | 2nd column | DOUBLE | FITTED | Fitted values |
| | 3rd column | DOUBLE | RESIDUALS | Residual values |

### Examples

Assume that:

- `DM_PAL` is a schema belonging to USER1;

- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

**Example: Online ARIMA training**

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_ARIMA_DATA_TBL;
CREATE COLUMN TABLE PAL_ARIMA_DATA_TBL ("TIMESTAMP" INTEGER, "Y" DOUBLE);
----------------------- first round -----------------------------------
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (1, 450);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (2, 806);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (3, 647);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (4, -3300);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (5, 5308);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (6, -5201);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (7, 539);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (8, 10515);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (9, -3244);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (10, -4780);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (11, 2997);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (12, -6572);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (13, 4535);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (14, 5307);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (15, -612);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (16, -1399);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (17, 1385);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (18, -8420);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (19, 287);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (20, 9437);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (21, -584);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (22, -274);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (23, -2824);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (24, -5981);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (25, 2446);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (26, 63);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (27, 4233);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (28, 1999);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (29, -8218);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (30, -9725);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (31, 2443);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (32, 8493);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (33, -3494);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (34, -1763);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (35, -2697);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (36, -1798);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (37, 874);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (38, 4768);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (39, 1259);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (40, -3342);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (41, 2012);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (42, -5225);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (43, 1506);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (44, 4843);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (45, 314);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (46, -822);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (47, -2776);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (48, -1093);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (49, 1132);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (50, 4228);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (51, -1507);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (52, 3567);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (53, -2692);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (54, -3972);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (55, -1461);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (56, 3579);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (57, 2397);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (58, -1007);
```

```
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (59, -4925);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (60, 1886);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (61, -643);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (62, 3683);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (63, 387);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (64, -8401);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (65, 4998);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (66, -267);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (67, -2714);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (68, 3988);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (69, -599);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (70, -2339);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (71, 1107);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (72, -3051);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (73, 4560);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (74, 259);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (75, 677);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (76, -2779);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (77, 765);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (78, -4065);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (79, 250);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (80, 4092);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (81, 2048);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (82, -496);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (83, -1038);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (84, -3193);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (85, 3348);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (86, 4077);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (87, -684);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (88, -2980);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (89, 2712);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (90, -7051);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (91, -233);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (92, 4778);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (93, 6285);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (94, -2549);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (95, -106);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (96, -5638);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (97, -1150);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (98, 6995);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (99, -2835);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (100, -225);
DROP TABLE PAL_ARIMA_MODEL_TBL_IN;  -- model table - input
CREATE COLUMN TABLE PAL_ARIMA_MODEL_TBL_IN ("KEY" NVARCHAR(100), "VALUE"
NVARCHAR(5000));
DROP TABLE PAL_ARIMA_MODEL_TBL_OUT;  -- model table - output
CREATE COLUMN TABLE PAL_ARIMA_MODEL_TBL_OUT ("KEY" NVARCHAR(100), "VALUE"
NVARCHAR(5000));
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL ( "NAME" VARCHAR
(50),"INT_VALUE" INTEGER,"DOUBLE_VALUE" DOUBLE,"STRING_VALUE" VARCHAR (100));
INSERT INTO #PAL_PARAMETER_TBL VALUES ('P', 4, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('M', 8, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('LEARNING_RATE', NULL, 0.00001, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('OUTPUT_FITTED', 1, NULL, NULL);
CALL _SYS_AFL.PAL_ONLINE_ARIMA (PAL_ARIMA_DATA_TBL, "#PAL_PARAMETER_TBL",
PAL_ARIMA_MODEL_TBL_IN, PAL_ARIMA_MODEL_TBL_OUT, ?) WITH OVERVIEW;
TRUNCATE TABLE PAL_ARIMA_MODEL_TBL_IN;
INSERT INTO PAL_ARIMA_MODEL_TBL_IN (SELECT * FROM PAL_ARIMA_MODEL_TBL_OUT);
------------------------ second round -----------------------------------
TRUNCATE TABLE PAL_ARIMA_DATA_TBL;
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (101, 386);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (102, -7807);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (103, 3374);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (104, 6074);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (105, 241);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (106, 2137);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (107, -111);
```

```
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (108, -3984);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (109, -751);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (110, 4307);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (111, -2218);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (112, 2908);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (113, -3830);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (114, -7253);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (115, 1678);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (116, 7304);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (117, 3580);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (118, -879);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (119, -5217);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (120, 2246);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (121, 902);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (122, 3327);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (123, 4760);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (124, -8948);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (125, 11347);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (126, -14919);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (127, -3903);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (128, 6915);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (129, 5823);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (130, -3662);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (131, -4320);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (132, 1355);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (133, 2199);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (134, -5);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (135, 4977);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (136, -3395);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (137, 2206);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (138, -6743);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (139, -4286);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (140, 10419);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (141, 4858);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (142, -4507);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (143, -561);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (144, -5693);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (145, 4240);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (146, 4536);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (147, 4519);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (148, -2665);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (149, 6963);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (150, -14162);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (151, -7328);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (152, 9783);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (153, 6730);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (154, -1137);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (155, 588);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (156, -8073);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (157, 4205);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (158, 8168);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (159, -2627);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (160, 250);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (161, -397);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (162, -10121);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (163, 1063);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (164, 5150);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (165, 1638);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (166, 1728);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (167, -1385);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (168, -6889);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (169, 334);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (170, 6279);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (171, 1535);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (172, -2877);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (173, 2291);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (174, -12725);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (175, 1368);
```

```
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (176, 3667);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (177, 6124);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (178, -1152);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (179, -3744);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (180, -1978);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (181, -1548);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (182, 4627);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (183, -3400);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (184, 1845);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (185, -246);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (186, -10621);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (187, 1162);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (188, 1714);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (189, 4010);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (190, -3570);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (191, -3126);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (192, -508);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (193, 436);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (194, 4198);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (195, 3197);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (196, -1359);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (197, -1743);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (198, -6029);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (199, -138);
INSERT INTO PAL_ARIMA_DATA_TBL VALUES (200, 3065);
TRUNCATE TABLE #PAL_PARAMETER_TBL;
TRUNCATE TABLE PAL_ARIMA_MODEL_TBL_OUT;
CALL _SYS_AFL.PAL_ONLINE_ARIMA (PAL_ARIMA_DATA_TBL, "#PAL_PARAMETER_TBL",
PAL_ARIMA_MODEL_TBL_IN, PAL_ARIMA_MODEL_TBL_OUT, ?) WITH OVERVIEW;
TRUNCATE TABLE PAL_ARIMA_MODEL_TBL_IN;
INSERT INTO PAL_ARIMA_MODEL_TBL_IN (SELECT * FROM PAL_ARIMA_MODEL_TBL_OUT);
----------------------- more rounds executed until all data being loaded
----------------
```

### Expected Result

MODEL:

| | KEY | VALUE |
|---|---|---|
| 1 | lrate | 1e-05 |
| 2 | fitted | 1 |
| 3 | d | 0 |
| 4 | mp | 12 |
| 5 | epsilon | 2.4e-09 |
| 6 | ar | 0.108672;0.274177;-0.07472... |
| 7 | dy | 4010;-3570;-3126;-508;436;4... |

FIT:

| | TIMESTAMP | FITTED | RESIDUALS |
|---|---|---|---|
| 101 | 101 | 772 | -386 |
| 102 | 102 | -15,662.66302 | 7,855.66302 |
| 103 | 103 | 7,509.2050603779... | -4,135.205060377904 |
| 104 | 104 | 13,960.873344704... | -7,886.873344704485 |
| 105 | 105 | -1,702.234044818... | 1,943.2340448187415 |
| 106 | 106 | 2,383.0778864318... | -246.0778864318404 |
| 107 | 107 | 4,284.7821163235... | -4,395.782116323562 |
| 108 | 108 | -7,366.329370191... | 3,382.329370191595 |
| 109 | 109 | -5,367.447849609... | 4,616.447849609294 |
| 110 | 110 | 7,704.4742297071... | -3,397.4742297071607 |
| 111 | 111 | -6,243.636635807... | 4,025.6366358078926 |
| 112 | 112 | 3,860.9141423574... | -952.9141423574997 |
| 113 | 113 | -4,910.790809659... | 1,080.7908096599663 |
| 114 | 114 | -12,717.07317728... | 5,464.073177283965 |
| 115 | 115 | 3,094.3304683307... | -1,416.3304683307329 |
| 116 | 116 | 15,841.915331021... | -8,537.915331021008 |
| 117 | 117 | 4,321.4439643889... | -741.4439643889718 |
| 118 | 118 | -3,452.121114640... | 2,573.1211146407777 |
| 119 | 119 | -5,393.172161995... | 176.17216199572704 |
| 120 | 120 | 7,742.9842917944... | -5,496.984291794464 |
| 121 | 121 | -1,465.805346908... | 2,367.805346908042 |
| 122 | 122 | 914.86939525310... | 2,412.1306047468906 |
| 123 | 123 | 8,113.7025956192... | -3,353.702595619282 |
| 124 | 124 | -16,161.66136552... | 7,213.661365521438 |
| 125 | 125 | 23,346.723497763... | -11,999.72349776354 |
| 126 | 126 | -29,033.45202431... | 14,114.452024316526 |

## _SYS_AFL.PAL_ONLINE_ARIMA_FORECAST

This function makes time series forecast based on the estimated ARIMA model.

**Overview of All Tables**

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<MODEL table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<FORECAST table>` | OUT |

### Input Tables

| Table | Column | Data Type | Description |
|---|---|---|---|
| MODEL | 1st column | VARCHAR or NVARCHAR | Model key |
| | 2nd column | VARCHAR or NVARCHAR | Model value |

### Parameter Table

Mandatory Parameters

None.

Optional Parameter

The following parameter is optional. If it is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| FORECAST_LENGTH | INTEGER | 1 | Number of points to forecast. |

### Output Table

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| RESULT | 1st column | INTEGER | ID (in input table) column name | Time stamp |
| | 2nd column | DOUBLE | FORECAST | Forecast value |

### Examples

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

### Example: Online ARIMA forecast

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_ARIMA_PREDICT_TBL;
CREATE COLUMN TABLE PAL_ARIMA_PREDICT_TBL ("TIMESTAMP" INTEGER, "Y" DOUBLE);
TRUNCATE TABLE #PAL_PARAMETER_TBL;
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FORECAST_LENGTH', 10, NULL, NULL);
CALL _SYS_AFL.PAL_ONLINE_ARIMA_FORECAST(PAL_ARIMA_MODEL_TBL_OUT,
"#PAL_PARAMETER_TBL", ?) WITH OVERVIEW;
```

**Expected Result**

RESULT:

| | ID | FORECAST |
|---|---|---|
| 1 | 0 | 1,370.6701168459995 |
| 2 | 1 | -207.0186525531115 |
| 3 | 2 | -3,869.3434467267657 |
| 4 | 3 | -2,940.2434005781324 |
| 5 | 4 | 172.54342619472195 |
| 6 | 5 | 1,466.5020334022795 |
| 7 | 6 | 1,107.330539635385 |
| 8 | 7 | -1,517.4327466565458 |
| 9 | 8 | -2,972.0456763079133 |
| 10 | 9 | -2,064.764149239913 |

## 5.5.23 Online Bayesian Change Point Detection

Online Bayesian Change Point Detection (OnlineBCPD) is a method of online learning that continuously detects change points within the incoming data.

Unlike the offline version, OnlineBCPD is not decomposition-based. Instead, it focuses on causal predictive filtering, which involves generating an accurate distribution of the next unseen datum in the sequence based solely on the data that has already been observed. Thus, OnlineBCPD can provide either the exact position of the change point or the probability of each time step being a change point.

### Prerequisites

- No null data in the inputs. The timestamp should be unique (no duplication) and the incremental value of each timestamp must be 1.
- The length of time series must be at least 1.
- To start a new OnlineBCPD process, clear the contents in the IN_MODEL table.

### _SYS_AFL.PAL_ONLINE_BCPD

This function continuously detects change points within the time series.

**Overview of All Tables**

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |

| Position | Table Name | Parameter Type |
|---|---|---|
| 2 | `<IN_MODEL table>` | IN |
| 3 | `<PARAMETER table>` | IN |
| 4 | `<OUT_MODEL table>` | OUT |
| 5 | `<CHANGE_POINT table>` | OUT |

### Input Table

| Table | Column | Data Type | Description |
|---|---|---|---|
| DATA | 1st Column | INTEGER | Timestamp. Timestamp does not need to be in order, but must be unique and the incremental value of each timestamp should be 1. |
| | 2nd Column | INTEGER, DOUBLE, or DECIMAL(p,s) | Time series values |
| IN_MODEL | 1st Column | INTEGER | ID column |
| | 2nd Column | INTEGER, DOUBLE, or DECIMAL(p,s) | Alpha value of the distribution in the Bayesian process |
| | 3rd Column | INTEGER, DOUBLE, or DECIMAL(p,s) | Beta value of the distribution in the Bayesian process |
| | 4th Column | INTEGER, DOUBLE, or DECIMAL(p,s) | Kappa value of the distribution in Bayesian process |
| | 5th Column | INTEGER, DOUBLE, or DECIMAL(p,s) | Mu value of the distribution in the Bayesian process |
| | 6th Column | INTEGER, DOUBLE, or DECIMAL(p,s) | Growth probabilities in the Bayesian process |

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION |
|---|---|---|---|
| DELAY | INTEGER | 3 | Number of future time steps to determine whether the current time step is a change point. The greater the value, the more robust the results will be. But this comes at the cost of increased latency. |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION |
|---|---|---|---|
| PRUNE | INTEGER | 0 | Specifies whether to reduce the size of the model table after each run:<br>• 0: Do not prune<br>• 1: Prune |
| THRESHOLD | DOUBLE | 0.0 | Threshold to determine a change point:<br>• 0: Returns the probability of a change point for each time step<br>• 0~1: Only returns the time step of which the probability is above the threshold. |
| LAMBDA | DOUBLE | 250.0 | Parameter of the constant hazard function. |
| ALPHA | DOUBLE | 0.1 | Parameter of t-distribution. |
| BETA | DOUBLE | 0.01 | Parameter of t-distribution. |
| KAPPA | DOUBLE | 1.0 | Parameter of t-distribution. |
| MU | DOUBLE | 0.0 | Parameter of t-distribution. |

### Output Table

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| OUT_MODEL | 1st Column | INTEGER | ID | ID column |
| | 2nd Column | INTEGER, DOUBLE, or DECIMAL(p,s) | ALPHA | Alpha value of the distribution in the Bayesian process |
| | 3rd Column | INTEGER, DOUBLE, or DECIMAL(p,s) | BETA | Beta value of the distribution in the Bayesian process |
| | 4th Column | INTEGER, DOUBLE, or DECIMAL(p,s) | KAPPA | Kappa value of the distribution in the Bayesian process |
| | 5th Column | INTEGER, DOUBLE, or DECIMAL(p,s) | MU | Mu value of the distribution in the Bayesian process |
| | 6th Column | INTEGER, DOUBLE, or DECIMAL(p,s) | PROB | Growth probabilities in the Bayesian process |
| CHANGE_POINT | 1st Column | INTEGER | ID | Indicates the ID of the current row. |

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| | 2nd Column | INTEGER | POSITION | Position of change points in the season part |
| | 3rd Column | DOUBLE | PROBABILITY | Probability of being a change point at the corresponding time step |

### Examples

Assume that:

- `DM_PAL` is a schema belonging to `USER1`;
- `USER1` has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

### Example

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_ONLINE_BCPD_DATA_TBL;
CREATE COLUMN TABLE PAL_ONLINE_BCPD_DATA_TBL ("TIMESTAMP" INTEGER, "Y" DOUBLE);
----------------------- first run -----------------------------------
INSERT INTO PAL_ONLINE_BCPD_DATA_TBL VALUES (0, 10.42493405);
INSERT INTO PAL_ONLINE_BCPD_DATA_TBL VALUES (1, 8.94386084);
INSERT INTO PAL_ONLINE_BCPD_DATA_TBL VALUES (2, 9.11572192);
INSERT INTO PAL_ONLINE_BCPD_DATA_TBL VALUES (3, 9.17946682);
INSERT INTO PAL_ONLINE_BCPD_DATA_TBL VALUES (4, 9.53324699);
INSERT INTO PAL_ONLINE_BCPD_DATA_TBL VALUES (5, 9.46962137);
INSERT INTO PAL_ONLINE_BCPD_DATA_TBL VALUES (6, 9.81696916);
INSERT INTO PAL_ONLINE_BCPD_DATA_TBL VALUES (7, 9.49802322);
INSERT INTO PAL_ONLINE_BCPD_DATA_TBL VALUES (8, 10.48156282);
INSERT INTO PAL_ONLINE_BCPD_DATA_TBL VALUES (9, 8.92201745);
INSERT INTO PAL_ONLINE_BCPD_DATA_TBL VALUES (10, 9.30757129);
INSERT INTO PAL_ONLINE_BCPD_DATA_TBL VALUES (11, 9.99704509);
INSERT INTO PAL_ONLINE_BCPD_DATA_TBL VALUES (12, 8.9834754);
INSERT INTO PAL_ONLINE_BCPD_DATA_TBL VALUES (13, 10.56947665);
INSERT INTO PAL_ONLINE_BCPD_DATA_TBL VALUES (14, 9.20230684);
INSERT INTO PAL_ONLINE_BCPD_DATA_TBL VALUES (15, 9.53051802);
INSERT INTO PAL_ONLINE_BCPD_DATA_TBL VALUES (16, 9.32981842);
INSERT INTO PAL_ONLINE_BCPD_DATA_TBL VALUES (17, 9.38746945);
INSERT INTO PAL_ONLINE_BCPD_DATA_TBL VALUES (18, 9.86918752);
INSERT INTO PAL_ONLINE_BCPD_DATA_TBL VALUES (19, 10.07433313);
INSERT INTO PAL_ONLINE_BCPD_DATA_TBL VALUES (20, 9.36983432);
INSERT INTO PAL_ONLINE_BCPD_DATA_TBL VALUES (21, 10.45570875);
INSERT INTO PAL_ONLINE_BCPD_DATA_TBL VALUES (22, 9.77227027);
INSERT INTO PAL_ONLINE_BCPD_DATA_TBL VALUES (23, 9.35569118);
INSERT INTO PAL_ONLINE_BCPD_DATA_TBL VALUES (24, 9.35977052);
INSERT INTO PAL_ONLINE_BCPD_DATA_TBL VALUES (25, 9.82279237);
INSERT INTO PAL_ONLINE_BCPD_DATA_TBL VALUES (26, 9.93551206);
INSERT INTO PAL_ONLINE_BCPD_DATA_TBL VALUES (27, 9.82017479);
INSERT INTO PAL_ONLINE_BCPD_DATA_TBL VALUES (28, -15.71515274);
INSERT INTO PAL_ONLINE_BCPD_DATA_TBL VALUES (29, -15.58654124);
INSERT INTO PAL_ONLINE_BCPD_DATA_TBL VALUES (30, -15.28297248);
INSERT INTO PAL_ONLINE_BCPD_DATA_TBL VALUES (31, -17.85080158);
INSERT INTO PAL_ONLINE_BCPD_DATA_TBL VALUES (32, -18.277881);
INSERT INTO PAL_ONLINE_BCPD_DATA_TBL VALUES (33, -15.97301583);
INSERT INTO PAL_ONLINE_BCPD_DATA_TBL VALUES (34, -16.70173896);
INSERT INTO PAL_ONLINE_BCPD_DATA_TBL VALUES (35, -17.07498221);
INSERT INTO PAL_ONLINE_BCPD_DATA_TBL VALUES (36, -14.23073668);
INSERT INTO PAL_ONLINE_BCPD_DATA_TBL VALUES (37, -16.93035542);
INSERT INTO PAL_ONLINE_BCPD_DATA_TBL VALUES (38, -15.57166321);
```

```
INSERT INTO PAL_ONLINE_BCPD_DATA_TBL VALUES (39, -17.7200683);
INSERT INTO PAL_ONLINE_BCPD_DATA_TBL VALUES (40, -15.74905358);
INSERT INTO PAL_ONLINE_BCPD_DATA_TBL VALUES (41, -17.50864792);
INSERT INTO PAL_ONLINE_BCPD_DATA_TBL VALUES (42, -15.63466234);
INSERT INTO PAL_ONLINE_BCPD_DATA_TBL VALUES (43, -15.6380366);
INSERT INTO PAL_ONLINE_BCPD_DATA_TBL VALUES (44, -17.99124459);
INSERT INTO PAL_ONLINE_BCPD_DATA_TBL VALUES (45, -15.41031108);
INSERT INTO PAL_ONLINE_BCPD_DATA_TBL VALUES (46, -15.54203995);
INSERT INTO PAL_ONLINE_BCPD_DATA_TBL VALUES (47, -15.908129);
INSERT INTO PAL_ONLINE_BCPD_DATA_TBL VALUES (48, -16.24160879);
INSERT INTO PAL_ONLINE_BCPD_DATA_TBL VALUES (49, -14.25553011);
INSERT INTO PAL_ONLINE_BCPD_DATA_TBL VALUES (50, -13.37089312);
INSERT INTO PAL_ONLINE_BCPD_DATA_TBL VALUES (51, -14.55209207);
INSERT INTO PAL_ONLINE_BCPD_DATA_TBL VALUES (52, -14.32659962);
INSERT INTO PAL_ONLINE_BCPD_DATA_TBL VALUES (53, -13.41518447);
INSERT INTO PAL_ONLINE_BCPD_DATA_TBL VALUES (54, -13.40716758);
INSERT INTO PAL_ONLINE_BCPD_DATA_TBL VALUES (55, -13.70049752);
INSERT INTO PAL_ONLINE_BCPD_DATA_TBL VALUES (56, -14.97310396);
INSERT INTO PAL_ONLINE_BCPD_DATA_TBL VALUES (57, -13.1749954);
INSERT INTO PAL_ONLINE_BCPD_DATA_TBL VALUES (58, -14.26574249);
INSERT INTO PAL_ONLINE_BCPD_DATA_TBL VALUES (59, -14.39144937);
DROP TABLE PAL_ONLINE_BCPD_MODEL_TBL;
CREATE COLUMN TABLE PAL_ONLINE_BCPD_MODEL_TBL (
    "ID"   INTEGER,
    "ALPHA" DOUBLE,
    "BETA"  DOUBLE,
    "KAPPA" DOUBLE,
    "MU"    DOUBLE,
    "PROB"  DOUBLE
);
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR(100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR(100)
);
INSERT INTO PAL_PARAMETER_TBL VALUES ('THRESHOLD', NULL,        0.5,   NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('DELAY',        5,       NULL,   NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('PRUNE',        1,       NULL,   NULL);
DROP TABLE PAL_ONLINE_BCPD_CP_TBL;
CREATE COLUMN TABLE PAL_ONLINE_BCPD_CP_TBL (ID INTEGER, POSITION INTEGER,
PROBABILITY DOUBLE);
DO BEGIN
    lt_data  = SELECT * FROM PAL_ONLINE_BCPD_DATA_TBL;
    lt_model = SELECT * FROM PAL_ONLINE_BCPD_MODEL_TBL;
        lt_param = SELECT * FROM PAL_PARAMETER_TBL;
       CALL _SYS_AFL.PAL_ONLINE_BCPD (:lt_data, :lt_model, :lt_param,
lt_out_model, lt_cp);
    TRUNCATE TABLE PAL_ONLINE_BCPD_MODEL_TBL;
    TRUNCATE TABLE PAL_ONLINE_BCPD_CP_TBL;
    INSERT INTO PAL_ONLINE_BCPD_MODEL_TBL SELECT * FROM :lt_out_model;
    INSERT INTO PAL_ONLINE_BCPD_CP_TBL    SELECT * FROM :lt_cp;
END;
SELECT * FROM PAL_ONLINE_BCPD_MODEL_TBL;
SELECT * FROM PAL_ONLINE_BCPD_CP_TBL;
----------------------- second run -------------------------------------
TRUNCATE TABLE PAL_ONLINE_BCPD_DATA_TBL;
INSERT INTO PAL_ONLINE_BCPD_DATA_TBL VALUES (60, -13.85545588);
INSERT INTO PAL_ONLINE_BCPD_DATA_TBL VALUES (61, -13.21785347);
INSERT INTO PAL_ONLINE_BCPD_DATA_TBL VALUES (62, -13.99568127);
INSERT INTO PAL_ONLINE_BCPD_DATA_TBL VALUES (63, -14.1745645);
INSERT INTO PAL_ONLINE_BCPD_DATA_TBL VALUES (64, -13.61761617);
INSERT INTO PAL_ONLINE_BCPD_DATA_TBL VALUES (65, -12.6770854);
INSERT INTO PAL_ONLINE_BCPD_DATA_TBL VALUES (66, -14.7524888);
INSERT INTO PAL_ONLINE_BCPD_DATA_TBL VALUES (67, -14.07877778);
INSERT INTO PAL_ONLINE_BCPD_DATA_TBL VALUES (68, 24.63360416);
INSERT INTO PAL_ONLINE_BCPD_DATA_TBL VALUES (69, 25.90989795);
```

```
INSERT INTO PAL_ONLINE_BCPD_DATA_TBL VALUES (70, 25.21448161);
INSERT INTO PAL_ONLINE_BCPD_DATA_TBL VALUES (71, 24.88162198);
INSERT INTO PAL_ONLINE_BCPD_DATA_TBL VALUES (72, 25.6508076);
INSERT INTO PAL_ONLINE_BCPD_DATA_TBL VALUES (73, 25.27387217);
INSERT INTO PAL_ONLINE_BCPD_DATA_TBL VALUES (74, 25.23195676);
INSERT INTO PAL_ONLINE_BCPD_DATA_TBL VALUES (75, 25.17773311);
INSERT INTO PAL_ONLINE_BCPD_DATA_TBL VALUES (76, 25.72653204);
INSERT INTO PAL_ONLINE_BCPD_DATA_TBL VALUES (77, 25.40448774);
INSERT INTO PAL_ONLINE_BCPD_DATA_TBL VALUES (78, 26.00679551);
DO BEGIN
    lt_data  = SELECT * FROM PAL_ONLINE_BCPD_DATA_TBL;
    lt_model = SELECT * FROM PAL_ONLINE_BCPD_MODEL_TBL;
       lt_param = SELECT * FROM PAL_PARAMETER_TBL;
       CALL _SYS_AFL.PAL_ONLINE_BCPD (:lt_data, :lt_model, :lt_param,
lt_out_model, lt_cp);
    TRUNCATE TABLE PAL_ONLINE_BCPD_MODEL_TBL;
    TRUNCATE TABLE PAL_ONLINE_BCPD_CP_TBL;
    INSERT INTO PAL_ONLINE_BCPD_MODEL_TBL SELECT * FROM :lt_out_model;
    INSERT INTO PAL_ONLINE_BCPD_CP_TBL    SELECT * FROM :lt_cp;
END;
SELECT * FROM PAL_ONLINE_BCPD_MODEL_TBL;
SELECT * FROM PAL_ONLINE_BCPD_CP_TBL;
```

### Expected Result

MODEL table after the first run:

| | ID | ALPHA | BETA | KAPP | MU | PROB |
|---|---|---|---|---|---|---|
| 1 | 0 | 0.1 | 0.01 | 1 | 0 | 0.003999999999999999 |
| 2 | 1 | 0.6 | 51.788453742318346 | 2 | -7.195724685 | 0.00008483835575909444 |
| 3 | 2 | 1.1 | 68.45017099665735 | 3 | -9.552397286666666 | 0.00001507072153112133 9 |
| 4 | 3 | 1.6 | 73.37137740567968 | 4 | -10.458046815000001 | 0.000005232972586875656 |
| 5 | 4 | 2.1 | 81.5256738147259 | 5 | -11.361058244 | 0.000001703266552818641 7 |
| 6 | 5 | 2.6 | 83.80608053393065 | 6 | -11.75096479 | 8.192661191631451e-7 |
| 7 | 6 | 3.1 | 84.98165525461799 | 7 | -11.987565188571429 | 4.874873728290579e-7 |
| 8 | 7 | 3.6 | 85.87332261017713 | 8 | -12.16601759875 | 3.2876676042557386e-7 |
| 9 | 8 | 4.1 | 87.94804024153211 | 9 | -12.40608226777778 | 2.059076975027205e-7 |
| 10 | 9 | 4.6 | 90.02045137358739 | 10 | -12.620683247999999 | 1.3671731020838705e-7 |
| 11 | 10 | 5.1 | 90.27627630633569 | 11 | -12.688884145454546 | 1.2409943488705882e-7 |
| 12 | 11 | 5.6 | 91.40120027968952 | 12 | -12.819437975833333 | 1.0276691841795217e-7 |
| 13 | 12 | 6.1 | 96.80639400953606 | 13 | -13.082681884615385 | 6.576122752229197e-8 |
| 14 | 13 | 6.6 | 100.51285716038812 | 14 | -13.284499535714286 | 4.818233543724945e-8 |
| 15 | 14 | 7.1 | 102.89121856405036 | 15 | -13.435002229999998 | 4.02084017934096e-8 |
| 16 | 15 | 7.6 | 104.72020843259176 | 16 | -13.558459033125 | 3.67623999767017e-8 |
| 17 | 16 | 8.1 | 113.96707327646924 | 17 | -13.819211124705882 | 2.489257934758194e-8 |
| 18 | 17 | 8.6 | 115.52924393932587 | 18 | -13.920256984444444 | 2.500901378217034e-8 |
| 19 | 18 | 9.1 | 116.92148980819 | 19 | -14.010488845263156 | 2.696754178636514e-8 |
| 20 | 19 | 9.6 | 122.73412034146783 | 20 | -14.185396799 | 2.451561596610013e-8 |
| 21 | 20 | 10.1 | 123.8984167837379 | 21 | -14.259856645714281 | 2.987392066353072e-8 |
| 22 | 21 | 10.6 | 129.61283402331853 | 22 | -14.417138993636364 | 3.079822497256853e-8 |

CHANGE_POINT table after the first run:

| | ID | POSITION | PROBABILITY |
|---|---|---|---|
| 1 | 0 | 28 | 0.9865808853724001 |

MODEL table after the second run:

| | ID | ALPHA | BETA | KAPP | MU | PROB |
|---|---|---|---|---|---|---|
| 1 | 0 | 0.1 | 0.01 | 1 | 0 | 0.004 |
| 2 | 1 | 0.6 | 169.09835317473903 | 2 | 13.003397755 | 0.00011263784405354498 |
| 3 | 2 | 1.1 | 220.36069744676146 | 3 | 17.137094416666667 | 0.000030470499166924346 |
| 4 | 3 | 1.6 | 248.02761195368677 | 4 | 19.2844538225 | 0.00001350730867148601 |
| 5 | 4 | 2.1 | 261.91990825787735 | 5 | 20.463109680000002 | 0.000008244935132513763 |
| 6 | 5 | 2.6 | 271.39570095472095 | 6 | 21.257917526666663 | 0.000006519556689030609 |
| 7 | 6 | 3.1 | 278.3076545392826 | 7 | 21.83162533285714 | 0.000006582393068706986 |
| 8 | 7 | 3.6 | 284.68909655975824 | 8 | 22.30902311625 | 0.000008393220590251697 |
| 9 | 8 | 4.1 | 287.6305476325438 | 9 | 22.594867434444446 | 0.000013855765215627671 |
| 10 | 9 | 4.6 | 290.71861792549106 | 10 | 22.856828852 | 0.000031712337169541606 |
| 11 | 10 | 5.1 | 294.9555410696559 | 11 | 23.134380588181816 | 0.00011951645870900367 |
| 12 | 11 | 5.6 | 295.98572375720784 | 12 | 23.259315885833335 | 0.9956485596815333 |

CHANGE_POINT table after second run:

| | ID | POSITION | PROBABILITY |
|---|---|---|---|
| 1 | 0 | 68 | 0.986458678316938 |

## 5.5.24 Outlier Detection

In time series, an outlier is a data point that is different from the general behavior of remaining data points.

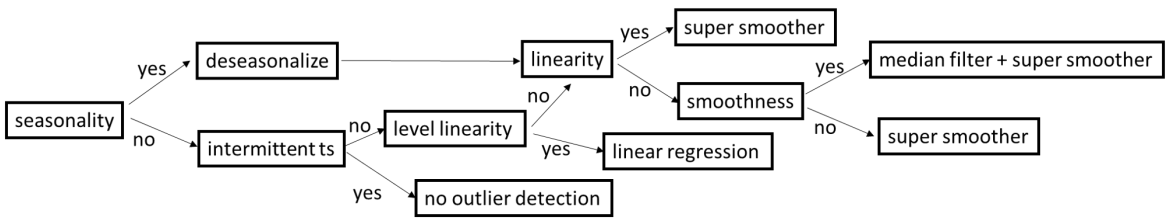
Outliers can have many causes, such as data entry error, experimental error, sampling error and natural outlier. Outliers have a huge impact on the result of data analysis, such as the seasonality test. Outlier detection is an important data preprocessing for time series analysis.

In time series, outliers can be categorized into three scenarios: point outlier, subsequence outlier, and system outlier. In this algorithm, we focus on the point outlier.

In this algorithm, the outlier detection procedure is divided into two steps. In step 1, we get the residual from the original series. In step 2, we detect the outliers from the residual.

In step 1, we have an automatic method and five manual methods to get the residual, depending on the choice of parameters.

- Automatic method:
  Residual from automatic procedure. In this method, we combine seasonal decomposition, linear regression, median filter and super smoother. For the procedure details, see the following picture. Because PAL deseasonalize the time series after seasonal decomposition, PAL takes the logarithm of the time series and do additive decomposition if it is a multiplicative model. If there is no seasonality with raw data, PAL removes the leading and trailing zeros, and does seasonality test again. Multiple seasonal decomposition is also available in this method. You can choose to detect whether the time series is intermittent by setting the parameter `DETECT_INTERMITTENT_TS` to 1. If it is intermittent, the algorithm doesn't do outlier detection. In this method, the window size of median filter is set to 3. If the last step to get the final residual is median filter, super smoother is also applied to detect consecutive outliers and two endpoint outliers.



- Manual method 1:
  Residual from median filter. The median filter is applied to the original series. PAL gets the residual from median filter directly. This method is applicable to the time series which is smooth.
- Manual method 2:
  Seasonal decomposition residual. In this method, median filter is not applied to the original series. PAL takes the original series to do seasonal decomposition and takes the decomposition residual. This method is applicable to the time series which is not smooth, but with strong seasonality.
- Manual method 3:
  Residual from median filter and seasonal decomposition. In this method, PAL applies the median filter to the original series first, and get the smooth series and median residual. Then, PAL applies the seasonal decomposition to the smooth series, and get the seasonal decomposition residual. The final residual is the combination of the median residual and seasonal decomposition residual.
- Manual method 4:
  Residual from LOESS. LOESS (locally estimated scatterplot smoothing) or LOWESS (locally weighted scatterplot smoothing) is a locally weighted linear regression method. This method is applicable to the time series which is non-seasonal. This method is also suitable for non-smooth time series.
- Manual method 5:
  Residual from super smoother. This method combines a set of LOESS methods. Like LOESS, this method is applicable to non-seasonal time series. This method is also suitable for non-smooth time series.

We can also skip step 1 and treat the raw time series as residual. This method is suitable for a time series without seasonality nor trend.

In the above methods, if the seasonal decomposition is not applied, and the residual is additive. If the seasonal decomposition is applied, the type of final residual depends on the seasonal decomposition type. More specifically, if the decomposition type is additive, the final residual is additive, and around 0. If the decomposition type is multiplicative, the final residual is also multiplicative, and around 1. For more information about seasonal decomposition, see Seasonality Test [page 934].

In step 2, we detect outliers from the residual series. We have six methods. They are Z1 method, Z2 method, IQR (Inter-Quartile Range) method, MAD (Median Absolute Deviation) method, isolation forest method, and DBSCAN (Density-Based Spatial Clustering of Applications with Noise) method. For the first five methods, we calculate the outlier score for each data point and compare the absolute value of outlier score with the threshold. If the absolute value of outlier score is beyond the threshold, we treat the data point as an outlier, otherwise, the data point is treated as a normal point. For DBSCAN method, there is no outlier score.

For the scores of the first five methods, we call them `Z1` score $S_{Z1}$, `Z2` score $S_{Z2}$, IQR (Inter-Quartile Range) score $S_{IQR}$, MAD (Median Absolute Deviation) score $S_{MAD}$, and isolation forest score $S_{IF}$.

Assume that the residual series is $\{x_i, i=1,2,...,N\}$.

For `Z1` score, we have

$$S_{Z1}(x_i) = \frac{x_i - \mu}{\sigma},$$

where $\mu = \frac{1}{N}\sum_{i=1}^{N} x_i$ is the mean, $\sigma = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \mu)^2}$ is the standard deviation.

For `Z2` score, because the additive residual is around 0 and the multiplicative residual is around 1, we have

$$S_{Z2}(x_i) = \begin{cases} \frac{x_i - 1}{\sigma}, & \text{multiplicative decomposition} \\ \frac{x_i}{\sigma}, & \text{otherwise} \end{cases},$$

where $\sigma$ is defined the same as in `Z1` score.

For IQR score, we have

$$S_{IQR}(x_i) = \begin{cases} \frac{x_i - Q1}{IQR}, & \text{when } x_i < Q1 \\ \frac{x_i - Q3}{IQR}, & \text{when } x_i > Q3 \\ 0, & \text{Otherwise} \end{cases},$$

where `Q1` is equal to 25th percentile, `Q3` is equal to 75th percentile and `IQR = Q3 - Q1`.

For MAD score, we have

$$S_{MAD}(x_i) = 0.67449 \times \frac{x_i - \tilde{x}}{MAD},$$

where $\tilde{x} = \text{median}(x_i)$ is the median of the residual and $MAD = \text{median}(|x_i - \tilde{x}|)$. The constant 0.67449 is to make the MAD score equivalent to the Z1 score and Z2 score when the data follows a normal distribution.

For isolation forest score, we treat the residual as one dimensional data. We apply isolation forest method to the residual and get $S_{IF}$. $S_{IF}$ is between 0 and 1. For more information about isolation forest, see Isolation Forest [page 1078].

For DBSCAN method, we also treat the residual as one dimensional data. We will do normalization to the residual $x_i$ and get the normalized residual $\hat{x}_i$. The normalization is as $\hat{x}_i = \frac{x_i - \mu}{\hat{\sigma}}$, where $\mu = \frac{1}{N}\sum_{i=1}^{N} x_i$ is the mean, $\hat{\sigma} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}$ is the sample standard deviation. DBSCAN method can divide normalized residual data points into several clusters. The data points with normalized residual points in the largest cluster are considered as normal points. Other points are considered as outliers. For more information about DBSCAN, see DBSCAN [page 111].

From the above formulas, we can find that IQR and MAD may be 0 when the series are very smooth. When `IQR` is 0, it means `Q1 = Q3`. If `IQR = 0` and we choose IQR score to detect outliers, all data points with residual not equal to `Q1` or `Q3` will be considered as outliers. Similarly, for MAD score method, if `MAD = 0`, all data points with residual not equal to the median residual will be considered as outliers.

In step 2, we can also use some of the previous methods and get the outlier result by voting. With the voting logic, there is no score in the output table.

For more details and examples of methods in step 1 and step 2, see Methods to Get Residual in Outlier Detection [page 846] and Detect Outlier from Residual [page 866].

## Prerequisites

- No missing or null data in the inputs.
- The data is numeric, not categorical.

## _SYS_AFL.PAL_OUTLIER_DETECTION_FOR_TIME_SERIES

This is a function of outlier detection for time series.

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA Table>` | IN |
| 2 | `<PARAMETER Table>` | IN |
| 3 | `<RESULT Table>` | OUT |
| 4 | `<STATISTICS Table>` | OUT |
| 5 | `<METRICS Table>` | OUT |

### Input Tables

| Table | Column | Data Type | Description |
|---|---|---|---|
| DATA | 1st column | INTEGER | ID of data |
| | 2nd column | INTEGER, DOUBLE, DECIMAL (p,s) | Raw data |
| PARAMETER | 1st column | VARCHAR or NVARCHAR | Parameter name |
| | 2nd column | INTEGER | Int value |
| | 3rd column | DOUBLE | Double value |
| | 4th column | VARCHAR or NVARCHAR | String value |

### Parameter Table

Mandatory Parameters

None

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| AUTO | INTEGER | 1 | • 0: Manual method to get residual<br>• 1: Automatic method to get residual | |
| DETECT_INTERMITTENT_TS | INTEGER | 0 | • 0: Does not detect whether the time series is intermittent.<br>• 1: Detects whether the time series is intermittent.<br>If DETECT_INTERMITTENT_TS is valid with value 1 and the time series is detected as intermittent, the algorithm doesn't do outlier detection. | Only valid when AUTO is 1. |
| SMOOTH_METHOD | INTEGER | 0 | The method to get the residual.<br>• -1: No smoothing<br>• 0: Median filter<br>• 1: LOESS<br>• 2: Super smoother | Only valid when AUTO is 0. |
| WINDOW_SIZE | INTEGER | 3 | Odd number, the window size for median filter, not less than 1. The value 1 means median filter is not applied. | Only valid when AUTO is 0 and SMOOTH_METHOD = 0. |
| LOESS_LAG | INTEGER | 7 | Odd number, the window size for LOESS, not less than 3. | Only valid when AUTO is 0 and SMOOTH_METHOD = 1. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| CURRENT_VALUE_FLAG | INTEGER | 0 | Whether to take the current data point when using LOESS smoothing method.<br>• 0: Does not take the current data point<br>• 1: Takes the current data point<br>For example, to estimate the value at time t with the window [t-3, t-2, t-1, t, t+1, t+2, t+3], taking the current data point means estimating the value at t with the real data points at [t-3, t-2, t-1, t, t+1, t+2, t+3], while not taking the current data point means estimating the value at t with the real data points at [t-3, t-2, t-1, t+1, t+2, t+3], without the real data point at t. | Only valid when AUTO is 0 and SMOOTH_METHOD = 1. |
| OUTLIER_METHOD | INTEGER | 0 | The method to calculate the outlier score from residual.<br>• 0: Z1 score<br>• 1: Z2 score<br>• 2: IQR score<br>• 3: MAD score<br>• 4: Isolation Forest score<br>• 5: DBSCAN | Only valid when VOTING_OUTLIER_METHOD is not specified. |
| THRESHOLD | DOUBLE | • 3 for OUTLIER_METHOD = 0,1,3<br>• 1.5 for OUTLIER_METHOD = 2<br>• 0.7 for OUTLIER_METHOD = 4 | The threshold for outlier score. If the absolute value of outlier score is beyond the threshold, PAL considers the corresponding data point as an outlier. | Only valid when VOTING_OUTLIER_METHOD is not specified and OUTLIER_METHOD = 0,1,2,3 or 4. For OUTLIER_METHOD = 4 (Isolation Forest), when CONTAMINATION is provided and valid, THRESHOLD is not valid and outliers are decided by CONTAMINATION. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| VOTING_OUTLIER_METHOD | INTEGER | No default value | Specifies the outlier methods for voting. The valid values are 0, 1, 2, 3, 4 and 5. They have the same meaning as OUTLIER_METHOD. This parameter supports multiple input values. | If it is specified, the outlier result is from the voting of selected outlier methods. Otherwise, PAL will use OUTLIER_METHOD to detect outliers. |
| THRESHOLD_OUTLIER_METHOD_{num} | DOUBLE | • 3 for num = 0,1,3<br>• 1.5 for num = 2<br>• 0.7 for num = 4 | The valid values of num are 0, 1, 2, 3, 4. Specifies the threshold for the outlier methods in voting. | Only valid when num is set as a value of VOTING_OUTLIER_METHOD. For THRESHOLD_OUTLIER_METHOD_4, when CONTAMINATION_OUTLIER_METHOD_4 is provided and valid, THRESHOLD_OUTLIER_METHOD_4 is not valid and outliers of Isolation Forest are decided by CONTAMINATION_OUTLIER_METHOD_4. |
| CONTAMINATION_OUTLIER_METHOD_{num} | DOUBLE | No default value | The only valid value of num is 4. This parameter specifies the proportion of outliers in the data set. The value should be in the range (0, 0.5]. | Only valid when 4 is set as a value of VOTING_OUTLIER_METHOD. When 4 is set as the value of VOTING_OUTLIER_METHOD and CONTAMINATION_OUTLIER_METHOD_4 is specified, THRESHOLD_OUTLIER_METHOD_4 is not valid. |
| VOTING_OUTLIER_METHOD_CRITERION | DOUBLE | 0.5 | The criterion for outlier voting. Suppose the number of voters is N, if more than int(criterion * N) voters detect the point as an outlier, the point will be treated as an outlier. | Only valid when VOTING_OUTLIER_METHOD is specified. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| DETECT_SEASONALITY | INTEGER | 0 | When calculating the residual,<br>• 0: Does not consider the seasonal decomposition.<br>• 1: Considers the seasonal decomposition. | Only valid when AUTO is 0 and SMOOTH_METHOD = 0. |
| ALPHA | DOUBLE | • 0.2 for AUTO = 0<br>• 0.4 for AUTO = 1 | The criterion for the autocorrelation coefficient. The value range is (0, 1). A larger value indicates stricter requirement for seasonality. | Only valid when AUTO is 1 or DETECT_SEASONALITY is valid with value 1. |
| EXTRAPOLATION | INTEGER | • 0 for AUTO = 0<br>• 1 for AUTO = 1 | Specifies whether to extrapolate the endpoints.<br>• 0: No<br>• 1: Yes | Set to 1 when there's an end-point issue. Only valid when AUTO is 1 or DETECT_SEASONALITY is valid with value 1. |
| PERIODS | INTEGER | No default value | When this parameter is not specified, the algorithm will search the seasonal period. When this parameter is specified between 2 and half of the series length, autocorrelation value is calculated for this number of periods and the result is compared to ALPHA parameter. If correlation value is equal to or higher than ALPHA, decomposition is executed with the value of PERIODS. Otherwise, the residual is calculated without decomposition. For other value of parameter PERIODS, the residual is also calculated without decomposition. | Only valid when DETECT_SEASONALITY is valid with value 1. If the user knows the seasonal period, specifying PERIODS can speed up the calculation, especially when the time series is long. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| SEED | INTEGER | 0 | Specifies the seed for random number generator.<br>• 0: Uses the current time (in second) as seed.<br>• Others: Uses the specified value as seed. | Only valid when OUTLIER_METHOD is 4 (Isolation Forest) or 4 is specified as the value of VOTING_OUTLIER_METHOD. |
| N_ESTIMATORS | INTEGER | 100 | Specifies the number of trees to grow. | Only valid when OUTLIER_METHOD is 4 (Isolation Forest) or 4 is specified as the value of VOTING_OUTLIER_METHOD. |
| MAX_SAMPLES | INTEGER | 256 | Specifies the number of samples to draw from input to train each tree. If MAX_SAMPLES is larger than the number of samples provided, all samples will be used for all trees. | Only valid when OUTLIER_METHOD is 4 (Isolation Forest) or 4 is specified as the value of VOTING_OUTLIER_METHOD. |
| BOOTSTRAP | INTEGER | 0 | Specifies sampling method.<br>• 0: Sampling without replacement.<br>• 1: Sampling with replacement. | Only valid when OUTLIER_METHOD is 4 (Isolation Forest) or 4 is specified as the value of VOTING_OUTLIER_METHOD. |
| CONTAMINATION | DOUBLE | No default value | The proportion of outliers in the data set. CONTAMINATION should be in the range (0, 0.5]. | Only valid when AUTO is 0 and OUTLIER_METHOD is 4 (Isolation Forest). When AUTO is 0, OUTLIER_METHOD is 4 and CONTAMINATION is specified, THRESHOLD is not valid. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| AUTO_PARAM | VARCHAR or NVARCHAR | false | Specifies whether the MINPTS and RADIUS parameters are determined automatically or by your input. This parameter is case sensitive.<br>• true: automatically determines the parameters.<br>• false: uses parameter values provided by your input. | Only valid when OUTLIER_METHOD is 5 (DBSCAN) or 5 is specified as the value of VOTING_OUTLIER_METHOD. |
| MINPTS | INTEGER | 1 | Specifies the minimum number of points required to form a cluster. The point itself is not included in MINPTS. | Only valid when OUTLIER_METHOD is 5 (DBSCAN) or 5 is specified as the value of VOTING_OUTLIER_METHOD. |
| RADIUS | DOUBLE | 0.5 | Specifies the scan radius (eps). | Only valid when OUTLIER_METHOD is 5 (DBSCAN) or 5 is specified as the value of VOTING_OUTLIER_METHOD. |
| DISTANCE_METHOD | DISTANCE_METHOD | 2 | Specifies the method to compute the distance between two points.<br>• 1: Manhattan<br>• 2: Euclidean<br>• 3: Minkowski<br>• 4: Chebyshev<br>• 5: Standardized Euclidean<br>• 6: Cosine | Only valid when OUTLIER_METHOD is 5 (DBSCAN) or 5 is specified as the value of VOTING_OUTLIER_METHOD. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| DBSCAN_NORMALI-ZATION | INTEGER | 1 | Specifies whether to take normalization of data before applying it to DBSCAN method.<br>• 0: Does not take normalization.<br>• 1: Takes normalization. | Only valid when OUT-LIER_METHOD is 5 (DBSCAN) or 5 is specified as the value of VOTING_OUT-LIER_METHOD. |
| DBSCAN_OUT-LIER_FROM_CLUSTER | INTEGER | 0 | Specifies how to take outliers from the DBSCAN result.<br>• 0: Takes the largest cluster as normal points and others as outliers.<br>• 1: Takes the points with CLUSTER_ID = -1 as outliers. | Only valid when OUT-LIER_METHOD is 5 (DBSCAN) or 5 is specified as the value of VOTING_OUT-LIER_METHOD. |
| RESIDUAL_USAGE | INTEGER | 0 | Specifies which residual to output.<br>• 0: Residual for outlier detection.<br>• 1: Residual for outlier correction. | |
| THREAD_RATIO | DOUBLE | -1 | The ratio of available threads.<br>• 0: single thread<br>• 0~1: percentage<br>• Others: heuristically determined | Only valid when AUTO is 1, or DE-TECT_SEASONALITY is 1, or OUT-LIER_METHOD is 4 or 5, or VOTING_OUT-LIER_METHOD is set to 4 or 5. |

When `AUTO` = 0, avoid the parameter combination {`DETECT_SEASONALITY` = 0, `WINDOW_SIZE` = 1}, because this parameter combination means that neither the seasonal decomposition or the median filter is applied to the original time series. The residual is always 0, and all the data points are treated as normal points.

When `SMOOTH_METHOD` = -1, we suggest not set `OUTLIER_METHOD` to 1 (Z2 score) or `VOTING_OUTLIER_METHOD` to 1 (Z2 score). For Z2 score, we assume the mean of the data is around 0 or 1. This is reasonable for residual, but not for raw data. For `SMOOTH_METHOD` = -1, no smoothing is done for the raw data. The input data of Z2 score is the raw data when `SMOOTH_METHOD` = -1.

### Output Tables

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| RESULT | 1st column | INTEGER | TIMESTAMP | ID of data |
| | 2nd column | DOUBLE | RAW_DATA | Original value |
| | 3rd column | DOUBLE | RESIDUAL | Residual |
| | 4th column | DOUBLE | OUTLIER_SCORE | Outlier score, empty for DBSCAN |
| | 5th column | INTEGER | IS_OUTLIER | • 0: normal<br>• 1: outlier |
| STATISTICS | 1st column | NVARCHAR | STAT_NAME | Name of statistics |
| | 2nd column | NCLOB | STAT_VALUE | Value of statistics |
| METRICS | 1st column | NVARCHAR | NAME | Name of metrics |
| | 2nd column | DOUBLE | VALUE | Value of metrics |

In the RESULT table, besides IS_OUTLIER column, we also provide RAW_DATA, RESIDUAL, OUTLIER_SCORE. This is because outlier detection is artificial. You can get more information from RESIDUAL and OUTLIER_SCORE columns, and decide whether to try with different choices of parameters. When `VOTING_OUTLIER_METHOD` is specified, the OUTLIER_SCORE column is empty.

For the RESIDUAL column, the default output is the residual for outlier detection. When `RESIDUAL_USAGE` = 1, the residual for outlier correction will be in the RESIDUAL column. These two residuals can be different in some cases. For example, when the outlier detection is from multiplicative seasonal decomposition, the residual for outlier detection is multiplicative, and the residual for outlier correction is transformed to be additive. When `AUTO` = 1, supplemented outliers may exist and logarithm may be taken in calculation. In such cases, the residual for outlier correction is different from the residual for outlier detection.

In the STATISTICS table, the result items depend on the parameter `AUTO`, `DETECT_SEASONALITY`, `OUTLIER_METHOD` and `VOTING_OUTLIER_METHOD`.

- When `AUTO` = 1, Auto Outlier Method, ID of Supplemented Outliers, DETECT_INTERMITTENT_TS, INTERMITTENT_FLAG, REMOVE_LEADING_AND_TRAILING_ZEROS, TAKING_LOG_FLAG, Periods, ACFs, Leading Zero Number and Trailing Zero Number may be in the table. Auto Outlier Method gives the method of getting the residual. If the median filter is the last step to get the final residual from deseasonalized time series or raw time series, super smoother is applied as a supplement. In such case, ID of Supplemented Outliers will be in the table and shows the supplemented outliers detected by the super smoother. DETECT_INTERMITTENT_TS and INTERMITTENT_FLAG will be in the table to show whether the time series detected as intermittent. REMOVE_LEADING_AND_TRAILING_ZEROS will be in the table and shows whether the leading and trailing zeros are removed during outlier detection. If yes, Leading Zero Number and Trailing Zero Number will be there and show the corresponding zero numbers. If the seasonal decomposition is applied, PAL shows TAKING_LOG_FLAG, Periods and ACFs. TAKING_LOG_FLAG shows whether we have taken logarithm of raw data before seasonal decomposition. Periods (seasonal cycles) and ACF (autocorrelation factors) are the process information of seasonal decomposition.
- When `AUTO` = 0, PAL shows the value of DETECT_SEASONALITY. If DETECT_SEASONALITY is 1 and valid, PAL also shows Type (decomposition type, the value would be additive, multiplicative or non-seasonal), Period (seasonal cycle) and ACF (autocorrelation factor). In the case of DETECT_SEASONALITY = 0 or not valid, when there is no value for PERIODS, ACF will be the maximum autocorrelation factor of all possible periods (between 2 and half of the series length); when PERIODS is specified between 2 and half of the

series length, ACF will be the autocorrelation factor of PERIODS; when PERIODS has other values, ACF will be -2.

- When `VOTING_OUTLIER_METHOD` is specified, the ID of outliers detected by each outlier method is shown in the table, such as ID of Z1 Outliers if 0 is specified as the value of `VOTING_OUTLIER_METHOD`. If super smoother is applied as a supplement, the ID of supplemented outliers detected by super smoother and each outlier method are shown in the table, such as ID of Z1 Supplemented Outliers. OutlierNum and HandleZero are also shown in the table. When `VOTING_OUTLIER_METHOD` is specified, ID of Supplemented Outliers is not shown in the table.
- When `VOTING_OUTLIER_METHOD` is not specified, for the different choices of parameter `OUTLIER_METHOD`, the result items are shown in the table below.

Result Items with Different Choices of OUTLIER_METHOD

| 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| OutlierNum | OutlierNum | OutlierNum | OutlierNum | OutlierNum | OutlierNum |
| Mean | Standard Deviation | Q1 | Median | HandleZero | HandleZero |
| Standard Deviation | HandleZero | Q3 | Median Absolute Deviation | | |
| HandleZero | | IQR | HandleZero | | |
| | | HandleZero | | | |

Here the value of HandleZero is 0 or 1. 1 for handling zero denominator or tiny denominator problem in the calculation, and 0 for not.

### Examples

Assume that:

- DM_PAL is a schema belonging to USER1;
- USER1 has been assigned the AFL__SYS_AFL_AFLPAL_EXECUTE or AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION role.

### Example 1: Automatic method

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_OUTLIER_DATA_TBL;
CREATE COLUMN TABLE PAL_OUTLIER_DATA_TBL (
    "ID" INT,
    "RAWDATA" DOUBLE
);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (1, 0.917022);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (2, 1.22032449);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (3, 10.50011437);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (4, 0.80233257);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (5, 0.64675589);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (6, 10.59233859);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (7, 0.68626021);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (8, 0.84556073);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (9, 10.89676747);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (10, 5.03881673);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (11, 0.91919451);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (12, 11.1852195);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (13, 0.70445225);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (14, 1.37811744);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (15, 10.52738759);
```

```
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (16, 1.17046751);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (17, 0.9173048);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (18, 11.05868983);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (19, 0.64038694);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (20, 0.69810149);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('AUTO', 1,NULL,NULL);
CALL "_SYS_AFL"."PAL_OUTLIER_DETECTION_FOR_TIME_SERIES"(PAL_OUTLIER_DATA_TBL,
#PAL_PARAMETER_TBL, ?, ?, ?);
```

**Expected Result**

RESULT:

| | TIMESTAMP | RAW_DATA | RESIDUAL | OUTLIER_SCORE | IS_OUTLIER |
|---|---|---|---|---|---|
| 1 | 1 | 0.917022 | -0.40119967155323444 | -0.6171619086623692 | 0 |
| 2 | 2 | 1.22032449 | 0.5503584118179408 | 0.44690969546399223 | 0 |
| 3 | 3 | 10.500114... | 0.07294272980278915 | -0.086956265505187... | 0 |
| 4 | 4 | 0.80233257 | -0.4257343867797525 | -0.6445976419550483 | 0 |
| 5 | 5 | 0.64675589 | 0.01234841075851456 | -0.154715339168834 | 0 |
| 6 | 6 | 10.592338... | 0.11042938446136219 | -0.045037137862908... | 0 |
| 7 | 7 | 0.68626021 | -0.6063237920449702 | -0.8465401838837576 | 0 |
| 8 | 8 | 0.84556073 | 0.06349161104324619 | -0.097524897389041... | 0 |
| 9 | 9 | 10.896767... | 0.2588438160027433 | 0.12092601772949499 | 0 |
| 10 | 10 | 5.03881673 | 3.5797404463683486 | 3.8344899781047603 | 1 |
| 11 | 11 | 0.91919451 | -0.006260579364621... | -0.175524681570030... | 0 |
| 12 | 12 | 11.1852195 | 0.4065744846495676 | 0.2861245613735919 | 0 |
| 13 | 13 | 0.70445225 | -0.7944701742875226 | -1.056933250016024 | 0 |
| 14 | 14 | 1.37811744 | 0.46172156263151365 | 0.34779230478998757 | 0 |
| 15 | 15 | 10.527387... | -0.15726583415965933 | -0.3443850028780142 | 0 |
| 16 | 16 | 1.17046751 | -0.26258861703724534 | -0.4621612947156716 | 0 |
| 17 | 17 | 0.9173048 | 0.14679529308242234 | -0.004371274792487... | 0 |
| 18 | 18 | 11.058689... | 0.49956842967410786 | 0.3901142368180312 | 0 |
| 19 | 19 | 0.64038694 | -0.5907021260252527 | -0.8290713908471448 | 0 |
| 20 | 20 | 0.69810149 | 0.0958177024288176 | -0.061376525033338... | 0 |

STATISTICS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | Auto Outlier Method | seasonal decomposition and super smoother |
| 2 | DETECT_INTERMITTENT_TS | 0 |
| 3 | INTERMITTENT_FLAG | 0 |
| 4 | REMOVE_LEADING_AND_... | 0 |
| 5 | TAKING_LOG_FLAG | 0 |
| 6 | Periods | 3 |
| 7 | ACFs | 0.786964 |
| 8 | OutlierNum | 1 |
| 9 | Mean | 0.150704 |
| 10 | Standard Deviation | 0.894261 |
| 11 | HandleZero | 0 |

**Example 2: Manual method without seasonal decomposition**

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_OUTLIER_DATA_TBL;
CREATE COLUMN TABLE PAL_OUTLIER_DATA_TBL (
    "ID" INT,
    "RAWDATA" DOUBLE
);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (1, 2.0);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (2, 2.5);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (3, 3.2);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (4, 2.8);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (5, 2.4);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (6, 2.9);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (7, 3.1);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (8, 8);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (9, 3.8);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (10, 4.2);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (11, 4.0);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (12, 4.3);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (13, 4.7);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (14, 5.1);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (15, 5.3);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (16, 10.0);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (17, 4.6);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (18, 4.4);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (19, 4.8);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (20, 5.1);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('AUTO', 0,NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('DETECT_SEASONALITY', 0,NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('OUTLIER_METHOD',0, NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('WINDOW_SIZE',3, NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES('THRESHOLD',NULL,3,NULL);
CALL "_SYS_AFL"."PAL_OUTLIER_DETECTION_FOR_TIME_SERIES"(PAL_OUTLIER_DATA_TBL,
#PAL_PARAMETER_TBL, ?, ?, ?);
```

**Expected Result**

RESULT:

| | TIMESTAMP | RAW_DATA | RESIDUAL | OUTLIER_SCORE | IS_OUTLIER |
|---|---|---|---|---|---|
| 1 | 1 | 2 | 0 | -0.2978496264349964 | 0 |
| 2 | 2 | 2.5 | 0 | -0.2978496264349964 | 0 |
| 3 | 3 | 3.2 | 0.40000... | -0.01076564914825... | 0 |
| 4 | 4 | 2.8 | 0 | -0.2978496264349964 | 0 |
| 5 | 5 | 2.4 | -0.39999... | -0.5849336037217399 | 0 |
| 6 | 6 | 2.9 | 0 | -0.2978496264349964 | 0 |
| 7 | 7 | 3.1 | 0 | -0.2978496264349964 | 0 |
| 8 | 8 | 8 | 4.2 | 2.7165321350758105 | 0 |
| 9 | 9 | 3.8 | -0.40000... | -0.5849336037217402 | 0 |
| 10 | 10 | 4.2 | 0.20000... | -0.1543076377916245 | 0 |
| 11 | 11 | 4 | -0.20000... | -0.44139161507836... | 0 |
| 12 | 12 | 4.3 | 0 | -0.2978496264349964 | 0 |
| 13 | 13 | 4.7 | 0 | -0.2978496264349964 | 0 |
| 14 | 14 | 5.1 | 0 | -0.2978496264349964 | 0 |
| 15 | 15 | 5.3 | 0 | -0.2978496264349964 | 0 |
| 16 | 16 | 10 | 4.7 | 3.0753871066842398 | 1 |
| 17 | 17 | 4.6 | 0 | -0.2978496264349964 | 0 |
| 18 | 18 | 4.4 | -0.19999... | -0.4413916150783676 | 0 |
| 19 | 19 | 4.8 | 0 | -0.2978496264349964 | 0 |
| 20 | 20 | 5.1 | 0 | -0.2978496264349964 | 0 |

STATISTICS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | DETECT_SEASONALITY | 0 |
| 2 | OutlierNum | 1 |
| 3 | Mean | 0.415 |
| 4 | Standard Deviation | 1.39332 |
| 5 | HandleZero | 0 |

**Example 3: Manual method with seasonal decomposition**

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_OUTLIER_DATA_TBL;
CREATE COLUMN TABLE PAL_OUTLIER_DATA_TBL (
    "ID" INT,
    "RAWDATA" DOUBLE
);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (1, 10);
```

```
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (2, 7);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (3, 17);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (4, 44);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (5, 9);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (6, 7);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (7, 18);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (8, 40);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (9, 27);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (10, 7);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (11, 27);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (12, 100);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (13, 93);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (14, -32);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (15, 159);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (16, 614);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (17, 548);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (18, 102);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (19, 21);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (20, 238);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('AUTO', 0,NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('DETECT_SEASONALITY', 1,NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('OUTLIER_METHOD',2, NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('WINDOW_SIZE',3, NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES('THRESHOLD',NULL,1.5,NULL);
CALL "_SYS_AFL"."PAL_OUTLIER_DETECTION_FOR_TIME_SERIES"(PAL_OUTLIER_DATA_TBL,
#PAL_PARAMETER_TBL, ?, ?, ?);
```

### Expected Result

RESULT:

| | TIMESTAMP | RAW_DATA | RESIDUAL | OUTLIER_SCORE | IS_OUTLIER |
|---|---|---|---|---|---|
| 1 | 1 | 10 | 0.8147277624485326 | 0 | 0 |
| 2 | 2 | 7 | 0.93455177345591 | 0 | 0 |
| 3 | 3 | 17 | 1.4675802857387537 | 0.43321514993... | 0 |
| 4 | 4 | 44 | 2.374211924930998 | 1.93960190514... | 1 |
| 5 | 5 | 9 | 0.6075541314259058 | 0 | 0 |
| 6 | 6 | 7 | 0.813704561371094 | 0 | 0 |
| 7 | 7 | 18 | 1.15464037186799 | 0 | 0 |
| 8 | 8 | 40 | 1.2590517783724988 | 0.08674082893... | 0 |
| 9 | 9 | 27 | 0.9245388956481174 | 0 | 0 |
| 10 | 10 | 7 | 0.3347153514859819 | -0.4490627881... | 0 |
| 11 | 11 | 27 | 0.6024210635832992 | -0.0042643478... | 0 |
| 12 | 12 | 100 | 1.0376800371201913 | 0 | 0 |
| 13 | 13 | 93 | 0.8860164416627792 | 0 | 0 |
| 14 | 14 | -32 | -0.32418925961248... | -1.5438460895... | 1 |
| 15 | 15 | 159 | 0.6553782030415365 | 0 | 0 |
| 16 | 16 | 614 | 1.2860480826777132 | 0.13159573305... | 0 |
| 17 | 17 | 548 | 1.4619104554947775 | 0.42379461262... | 0 |
| 18 | 18 | 102 | 0.6006083824168924 | -0.0072761541... | 0 |
| 19 | 19 | 21 | 0.11615543860707... | -0.8122044917... | 0 |
| 20 | 20 | 238 | 0.9013666140621298 | 0 | 0 |

STATISTICS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | DETECT_SEASONALITY | 1 |
| 2 | Type | multiplicative |
| 3 | Period | 4 |
| 4 | ACF | 0.520649 |
| 5 | OutlierNum | 2 |
| 6 | Q1 | 0.604987 |
| 7 | Q3 | 1.20684 |
| 8 | IQR | 0.601858 |
| 9 | HandleZero | 0 |

### Example 4: Residual for outlier correction

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_OUTLIER_DATA_TBL;
CREATE COLUMN TABLE PAL_OUTLIER_DATA_TBL (
    "ID" INT,
    "RAWDATA" DOUBLE
);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (1, 10);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (2, 7);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (3, 17);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (4, 44);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (5, 9);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (6, 7);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (7, 18);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (8, 40);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (9, 27);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (10, 7);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (11, 27);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (12, 100);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (13, 93);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (14, -32);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (15, 159);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (16, 614);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (17, 548);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (18, 102);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (19, 21);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (20, 238);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('AUTO', 0,NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('DETECT_SEASONALITY', 1,NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('OUTLIER_METHOD',2, NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('WINDOW_SIZE',3, NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES('THRESHOLD',NULL,1.5,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES('RESIDUAL_USAGE',1,NULL,NULL);
CALL "_SYS_AFL"."PAL_OUTLIER_DETECTION_FOR_TIME_SERIES"(PAL_OUTLIER_DATA_TBL,
#PAL_PARAMETER_TBL, ?, ?, ?);
```

### Expected Result

RESULT: The RESIDUAL column is different from example 3.

| | TIMESTAMP | RAW_DATA | RESIDUAL | OUTLIER_SCORE | IS_OUTLIER |
|---|---|---|---|---|---|
| 1 | 1 | 10 | -2.2740385941269707 | 0 | 0 |
| 2 | 2 | 7 | -0.49022172855599866 | 0 | 0 |
| 3 | 3 | 17 | 5.416306647617235 | 0.43321514993... | 0 |
| 4 | 4 | 44 | 25.46753474786006 | 1.93960190514... | 1 |
| 5 | 5 | 9 | -5.813494854980826 | 0 | 0 |
| 6 | 6 | 7 | -1.6026308961633244 | 0 | 0 |
| 7 | 7 | 18 | 2.410730441653101 | 0 | 0 |
| 8 | 8 | 40 | 8.230059567760101 | 0.08674082893... | 0 |
| 9 | 9 | 27 | -2.2037469998193444 | 0 | 0 |
| 10 | 10 | 7 | -13.913292351017741 | -0.4490627881... | 0 |
| 11 | 11 | 27 | -17.81914998024733 | -0.0042643478... | 0 |
| 12 | 12 | 100 | 3.6311806888723055 | 0 | 0 |
| 13 | 13 | 93 | -11.964192115292718 | 0 | 0 |
| 14 | 14 | -32 | -130.70777347235673 | -1.5438460895... | 1 |
| 15 | 15 | 159 | -83.60800750177361 | 0 | 0 |
| 16 | 16 | 614 | 136.56839517106152 | 0.13159573305... | 0 |
| 17 | 17 | 548 | 173.14803971729464 | 0.42379461262... | 0 |
| 18 | 18 | 102 | -67.82779958805185 | -0.0072761541... | 0 |
| 19 | 19 | 21 | -159.79222335027308 | -0.8122044917... | 0 |
| 20 | 20 | 238 | -26.043504925727177 | 0 | 0 |

STATISTICS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | DETECT_SEASONALITY | 1 |
| 2 | Type | multiplicative |
| 3 | Period | 4 |
| 4 | ACF | 0.520649 |
| 5 | OutlierNum | 2 |
| 6 | Q1 | 0.604987 |
| 7 | Q3 | 1.20684 |
| 8 | IQR | 0.601858 |
| 9 | HandleZero | 0 |

## _SYS_AFL.PAL_MASSIVE_OUTLIER_DETECTION_FOR_TIME_SERIES

This is a massive function of outlier detection for time series.

**Overview of All Tables**

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA Table>` | IN |
| 2 | `<PARAMETER Table>` | IN |
| 3 | `<RESULT Table>` | OUT |
| 4 | `<STATISTICS Table>` | OUT |
| 5 | `<METRICS Table>` | OUT |
| 6 | `<ERRORMSG table>` | OUT |

### Input Tables

| Table | Column | Data Type | Description |
|---|---|---|---|
| DATA | 1st column | INTEGER, VRACHAR or NVARCHAR | Group ID |
| | 2nd column | INTEGER | ID of data |
| | 3rd column | INTEGER, DOUBLE, DECIMAL (p,s) | Raw data |
| PARAMETER | 1st column | INTEGER, VRACHAR or NVARCHAR | Group ID |
| | 2nd column | VARCHAR or NVARCHAR | Parameter name |
| | 3rd column | INTEGER | Int value |
| | 4th column | DOUBLE | Double value |
| | 5th column | VARCHAR or NVARCHAR | String value |

### Parameter Table

See the Parameter Table in _SYS_AFL.PAL_OUTLIER_DETECTION_FOR_TIME_SERIES.

### Output Tables

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| | 1st column | NVARCHAR | GROUP_ID | Group ID |
| RESULT | 2nd column | INTEGER | TIMESTAMP | ID of data |
| | 3rd column | DOUBLE | RAW_DATA | Original value |
| | 4th column | DOUBLE | RESIDUAL | Residual |
| | 5th column | DOUBLE | OUTLIER_SCORE | Outlier score, empty for DBSCAN |
| | 6th column | INTEGER | IS_OUTLIER | • 0: normal<br>• 1: outlier |
| STATISTICS | 1st column | NVARCHAR | GROUP_ID | Group ID |
| | 2nd column | NVARCHAR | STAT_NAME | Name of statistics |
| | 3rd column | NCLOB | STAT_VALUE | Value of statistics |

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| METRICS | 1st column | NVARCHAR | GROUP_ID | GROUP_ID |
| | 2nd column | NVARCHAR | NAME | Name of metrics |
| | 3rd column | DOUBLE | VALUE | Value of metrics |
| ERRORMSG | 1st column | NVARCHAR | GROUP_ID | Group ID |
| | 2nd column | NVARCHAR | ERROR_TIMESTAMP | Error ID |
| | 3rd column | INTEGER | ERRORCODE | Error code |
| | 4th column | NVARCHAR | MESSAGE | Error message |

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA DM_PAL;
DROP TABLE #PAL_MASSIVE_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_MASSIVE_PARAMETER_TBL (
    "GROUPID" VARCHAR(100),
    "PARAM_NAME " VARCHAR(100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (100)
);
DROP TABLE PAL_MASSIVE_OUTLIER_DATA_TBL;
CREATE COLUMN TABLE PAL_MASSIVE_OUTLIER_DATA_TBL (
    "GROUPID" VARCHAR(100),
    "ID" INT,
    "RAWDATA" DOUBLE
);
INSERT INTO PAL_MASSIVE_OUTLIER_DATA_TBL VALUES ('100', 1, 0.917022);
INSERT INTO PAL_MASSIVE_OUTLIER_DATA_TBL VALUES ('100', 2, 1.22032449);
INSERT INTO PAL_MASSIVE_OUTLIER_DATA_TBL VALUES ('100', 3, 10.50011437);
INSERT INTO PAL_MASSIVE_OUTLIER_DATA_TBL VALUES ('100', 4, 0.80233257);
INSERT INTO PAL_MASSIVE_OUTLIER_DATA_TBL VALUES ('100', 5, 0.64675589);
INSERT INTO PAL_MASSIVE_OUTLIER_DATA_TBL VALUES ('100', 6, 10.59233859);
INSERT INTO PAL_MASSIVE_OUTLIER_DATA_TBL VALUES ('100', 7, 0.68626021);
INSERT INTO PAL_MASSIVE_OUTLIER_DATA_TBL VALUES ('100', 8, 0.84556073);
INSERT INTO PAL_MASSIVE_OUTLIER_DATA_TBL VALUES ('100', 9, 10.89676747);
INSERT INTO PAL_MASSIVE_OUTLIER_DATA_TBL VALUES ('100', 10, 5.03881673);
INSERT INTO PAL_MASSIVE_OUTLIER_DATA_TBL VALUES ('100', 11, 0.91919451);
INSERT INTO PAL_MASSIVE_OUTLIER_DATA_TBL VALUES ('100', 12, 11.1852195);
INSERT INTO PAL_MASSIVE_OUTLIER_DATA_TBL VALUES ('100', 13, 0.70445225);
INSERT INTO PAL_MASSIVE_OUTLIER_DATA_TBL VALUES ('100', 14, 1.37811744);
INSERT INTO PAL_MASSIVE_OUTLIER_DATA_TBL VALUES ('100', 15, 10.52738759);
INSERT INTO PAL_MASSIVE_OUTLIER_DATA_TBL VALUES ('100', 16, 1.17046751);
INSERT INTO PAL_MASSIVE_OUTLIER_DATA_TBL VALUES ('100', 17, 0.9173048);
INSERT INTO PAL_MASSIVE_OUTLIER_DATA_TBL VALUES ('100', 18, 11.05868983);
INSERT INTO PAL_MASSIVE_OUTLIER_DATA_TBL VALUES ('100', 19, 0.64038694);
INSERT INTO PAL_MASSIVE_OUTLIER_DATA_TBL VALUES ('100', 20, 0.69810149);
INSERT INTO PAL_MASSIVE_OUTLIER_DATA_TBL VALUES ('200', 1, 2.0);
INSERT INTO PAL_MASSIVE_OUTLIER_DATA_TBL VALUES ('200', 2, 2.5);
INSERT INTO PAL_MASSIVE_OUTLIER_DATA_TBL VALUES ('200', 3, 3.2);
INSERT INTO PAL_MASSIVE_OUTLIER_DATA_TBL VALUES ('200', 4, 2.8);
INSERT INTO PAL_MASSIVE_OUTLIER_DATA_TBL VALUES ('200', 5, 2.4);
INSERT INTO PAL_MASSIVE_OUTLIER_DATA_TBL VALUES ('200', 6, 2.9);
INSERT INTO PAL_MASSIVE_OUTLIER_DATA_TBL VALUES ('200', 7, 3.1);
INSERT INTO PAL_MASSIVE_OUTLIER_DATA_TBL VALUES ('200', 8, 8);
INSERT INTO PAL_MASSIVE_OUTLIER_DATA_TBL VALUES ('200', 9, 3.8);
```

```
INSERT INTO PAL_MASSIVE_OUTLIER_DATA_TBL VALUES ('200', 10, 4.2);
INSERT INTO PAL_MASSIVE_OUTLIER_DATA_TBL VALUES ('200', 11, 4.0);
INSERT INTO PAL_MASSIVE_OUTLIER_DATA_TBL VALUES ('200', 12, 4.3);
INSERT INTO PAL_MASSIVE_OUTLIER_DATA_TBL VALUES ('200', 13, 4.7);
INSERT INTO PAL_MASSIVE_OUTLIER_DATA_TBL VALUES ('200', 14, 5.1);
INSERT INTO PAL_MASSIVE_OUTLIER_DATA_TBL VALUES ('200', 15, 5.3);
INSERT INTO PAL_MASSIVE_OUTLIER_DATA_TBL VALUES ('200', 16, 10.0);
INSERT INTO PAL_MASSIVE_OUTLIER_DATA_TBL VALUES ('200', 17, 4.6);
INSERT INTO PAL_MASSIVE_OUTLIER_DATA_TBL VALUES ('200', 18, 4.4);
INSERT INTO PAL_MASSIVE_OUTLIER_DATA_TBL VALUES ('200', 19, 4.8);
INSERT INTO PAL_MASSIVE_OUTLIER_DATA_TBL VALUES ('200', 20, 5.1);
INSERT INTO #PAL_MASSIVE_PARAMETER_TBL VALUES ('100', 'AUTO', 1, NULL, NULL);
INSERT INTO #PAL_MASSIVE_PARAMETER_TBL VALUES ('200', 'AUTO', 0, NULL, NULL);
INSERT INTO #PAL_MASSIVE_PARAMETER_TBL VALUES ('200', 'DETECT_SEASONALITY', 0,
NULL, NULL);
INSERT INTO #PAL_MASSIVE_PARAMETER_TBL VALUES ('200', 'OUTLIER_METHOD', 0, NULL,
NULL);
INSERT INTO #PAL_MASSIVE_PARAMETER_TBL VALUES ('200', 'WINDOW_SIZE', 3, NULL,
NULL);
INSERT INTO #PAL_MASSIVE_PARAMETER_TBL VALUES ('200', 'THRESHOLD', NULL, 3,
NULL);
CALL
_SYS_AFL.PAL_MASSIVE_OUTLIER_DETECTION_FOR_TIME_SERIES(PAL_MASSIVE_OUTLIER_DATA_T
BL, #PAL_MASSIVE_PARAMETER_TBL, ?, ?, ?, ?);
```

### Expected Result

RESULT:

| | GROUP_ID | TIMESTAMP | RAW_DATA | RESIDUAL | OUTLIER_SCORE | IS_OUTLIER |
|---|---|---|---|---|---|---|
| 1 | 200 | 1 | 2 | 0 | -0.2978496264349964 | 0 |
| 2 | 200 | 2 | 2.5 | 0 | -0.2978496264349964 | 0 |
| 3 | 200 | 3 | 3.2 | 0.40000... | -0.01076564914825... | 0 |
| 4 | 200 | 4 | 2.8 | 0 | -0.2978496264349964 | 0 |
| 5 | 200 | 5 | 2.4 | -0.39999... | -0.5849336037217399 | 0 |
| 6 | 200 | 6 | 2.9 | 0 | -0.2978496264349964 | 0 |
| 7 | 200 | 7 | 3.1 | 0 | -0.2978496264349964 | 0 |
| 8 | 200 | 8 | 8 | 4.2 | 2.7165321350758105 | 0 |
| 9 | 200 | 9 | 3.8 | -0.40000... | -0.5849336037217402 | 0 |
| 10 | 200 | 10 | 4.2 | 0.20000... | -0.1543076377916245 | 0 |
| 11 | 200 | 11 | 4 | -0.20000... | -0.44139161507836... | 0 |
| 12 | 200 | 12 | 4.3 | 0 | -0.2978496264349964 | 0 |
| 13 | 200 | 13 | 4.7 | 0 | -0.2978496264349964 | 0 |
| 14 | 200 | 14 | 5.1 | 0 | -0.2978496264349964 | 0 |
| 15 | 200 | 15 | 5.3 | 0 | -0.2978496264349964 | 0 |
| 16 | 200 | 16 | 10 | 4.7 | 3.0753871066842398 | 1 |
| 17 | 200 | 17 | 4.6 | 0 | -0.2978496264349964 | 0 |
| 18 | 200 | 18 | 4.4 | -0.19999... | -0.4413916150783676 | 0 |
| 19 | 200 | 19 | 4.8 | 0 | -0.2978496264349964 | 0 |
| 20 | 200 | 20 | 5.1 | 0 | -0.2978496264349964 | 0 |
| 21 | 100 | 1 | 0.917022 | -0.40119... | -0.6171619086623692 | 0 |
| 22 | 100 | 2 | 1.22032449 | 0.55035... | 0.446909695463992... | 0 |
| 23 | 100 | 3 | 10.500114... | 0.07294... | -0.08695626550518... | 0 |
| 24 | 100 | 4 | 0.80233257 | -0.42573... | -0.6445976419550483 | 0 |
| 25 | 100 | 5 | 0.64675589 | 0.01234... | -0.154715339168834 | 0 |
| 26 | 100 | 6 | 10.592338... | 0.11042... | -0.04503713786290... | 0 |
| 27 | 100 | 7 | 0.68626021 | -0.60632... | -0.8465401838837576 | 0 |
| 28 | 100 | 8 | 0.84556073 | 0.06349... | -0.09752489738904... | 0 |
| 29 | 100 | 9 | 10.896767... | 0.25884... | 0.120926017729494... | 0 |
| 30 | 100 | 10 | 5.03881673 | 3.57974... | 3.8344899781047603 | 1 |
| 31 | 100 | 11 | 0.91919451 | -0.00626... | -0.17552468157003... | 0 |
| 32 | 100 | 12 | 11.1852195 | 0.40657... | 0.2861245613735919 | 0 |
| 33 | 100 | 13 | 0.70445225 | -0.79447... | -1.056933250016024 | 0 |
| 34 | 100 | 14 | 1.37811744 | 0.46172... | 0.347792304789987... | 0 |
| 35 | 100 | 15 | 10.527387... | -0.15726... | -0.3443850028780142 | 0 |
| 36 | 100 | 16 | 1.17046751 | -0.26258... | -0.4621612947156716 | 0 |
| 37 | 100 | 17 | 0.9173048 | 0.14679... | -0.00437127479248... | 0 |
| 38 | 100 | 18 | 11.058689... | 0.49956... | 0.3901142368180312 | 0 |
| 39 | 100 | 19 | 0.64038694 | -0.59070... | -0.8290713908471448 | 0 |
| 40 | 100 | 20 | 0.69810149 | 0.09581... | -0.06137652503333... | 0 |

STATISTICS:

| | GROUP_ID | STAT_NAME | STAT_VALUE |
|---|---|---|---|
| 1 | 200 | DETECT_SEASONALITY | 0 |
| 2 | 200 | OutlierNum | 1 |
| 3 | 200 | Mean | 0.415 |
| 4 | 200 | Standard Deviation | 1.39332 |
| 5 | 200 | HandleZero | 0 |
| 6 | 100 | Auto Outlier Method | seasonal decomposition and super smoother |
| 7 | 100 | DETECT_INTERMITTE... | 0 |
| 8 | 100 | INTERMITTENT_FLAG | 0 |
| 9 | 100 | REMOVE_LEADING_A... | 0 |
| 10 | 100 | TAKING_LOG_FLAG | 0 |
| 11 | 100 | Periods | 3 |
| 12 | 100 | ACFs | 0.786964 |
| 13 | 100 | OutlierNum | 1 |
| 14 | 100 | Mean | 0.150704 |
| 15 | 100 | Standard Deviation | 0.894261 |
| 16 | 100 | HandleZero | 0 |

METRICS:

| | GROUP_ID | NAME | VALUE |
|---|---|---|---|
| | | | |
| | | | |

ERRORMSG:

| | GROUP_ID | ERROR_TIMESTAMP | ERRORCODE | MESSAGE |
|---|---|---|---|---|
| | | | | |
| | | | | |

## Related Information

Seasonality Test [page 934]
Isolation Forest [page 1078]
DBSCAN [page 111]
Methods to Get Residual in Outlier Detection [page 846]
Detect Outlier from Residual [page 866]

## 5.5.24.1 Methods to Get Residual in Outlier Detection

PAL provides an automatic method and five manual methods to get the residual. This topic introduces the applicable scenarios and parameter choices of each method. At least one example is provided for each method.

You can also skip the step of smoothing and treat the raw data as residual. An example is provided with such choice.

When `AUTO` = 0 and `SMOOTH_METHOD` = 0, `WINDOW_SIZE` and `DETECT_SEASONALITY` are two important parameters to achieve the manual methods related to median filter and seasonal decomposition. The parameter combination {`DETECT_SEASONALITY` = 0, `WINDOW_SIZE` = 1} should be avoided when `AUTO` = 0 and `SMOOTH_METHOD` = 0, since PAL neither applies the median filter nor does the seasonal decomposition to the original time series. The residual is always 0.

### Automatic Method: Residual from Automatic Procedure

This method is applicable to the time series which is smooth.

**Relevant Parameters**

- `AUTO`
  To use manual method 1, you need to set `AUTO` = 1 (default value).
- `DETECT_INTERMITTENT_TS`
  If `DETECT_INTERMITTENT_TS` = 0, the algorithm does outlier detection as normal. If `DETECT_INTERMITTENT_TS` = 1, the algorithm detects whether the time series is intermittent. If yes, the algorithm doesn't do outlier detection. If not, the algorithm does outlier detection as normal. The default value is 0.
- `ALPHA`
  The criterion for the autocorrelation coefficient. The default value is 0.4 when `AUTO` = 1. The default value is recommended.
- `EXTRAPOLATION`
  Specifies whether to extrapolate the endpoints. The default value is 1 when `AUTO` = 1. The default value is recommended.
- `THREAD_RATIO`
  `THREAD_RATIO` is valid when `AUTO` = 1.

**Examples**

**Example 1: The time series is not intermittent**

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_OUTLIER_DATA_TBL;
CREATE COLUMN TABLE PAL_OUTLIER_DATA_TBL (
    "ID" INT,
    "RAWDATA" DOUBLE
);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (1, 0.917022);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (2, 1.22032449);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (3, 10.50011437);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (4, 0.80233257);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (5, 0.64675589);
```

```
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (6, 10.59233859);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (7, 0.68626021);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (8, 0.84556073);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (9, 10.89676747);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (10, 5.03881673);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (11, 0.91919451);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (12, 11.1852195);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (13, 0.70445225);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (14, 1.37811744);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (15, 10.52738759);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (16, 1.17046751);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (17, 0.9173048);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (18, 11.05868983);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (19, 0.64038694);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (20, 0.69810149);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('AUTO', 1,NULL,NULL);
CALL "_SYS_AFL"."PAL_OUTLIER_DETECTION_FOR_TIME_SERIES"(PAL_OUTLIER_DATA_TBL,
#PAL_PARAMETER_TBL, ?, ?, ?);
```

**Expected Result**

RESULT:

| | TIMESTAMP | RAW_DATA | RESIDUAL | OUTLIER_SCORE | IS_OUTLIER |
|---|---|---|---|---|---|
| 1 | 1 | 0.917022 | -0.40119967155323444 | -0.6171619086623692 | 0 |
| 2 | 2 | 1.22032449 | 0.5503584118179408 | 0.44690969546399223 | 0 |
| 3 | 3 | 10.500114... | 0.07294272980278915 | -0.086956265505187... | 0 |
| 4 | 4 | 0.80233257 | -0.4257343867797525 | -0.6445976419550483 | 0 |
| 5 | 5 | 0.64675589 | 0.01234841075851456 | -0.154715339168834 | 0 |
| 6 | 6 | 10.592338... | 0.11042938446136219 | -0.045037137862908... | 0 |
| 7 | 7 | 0.68626021 | -0.6063237920449702 | -0.8465401838837576 | 0 |
| 8 | 8 | 0.84556073 | 0.06349161104324619 | -0.097524897389041... | 0 |
| 9 | 9 | 10.896767... | 0.2588438160027433 | 0.12092601772949499 | 0 |
| 10 | 10 | 5.03881673 | 3.5797404463683486 | 3.8344899781047603 | 1 |
| 11 | 11 | 0.91919451 | -0.006260579364621... | -0.175524681570030... | 0 |
| 12 | 12 | 11.1852195 | 0.4065744846495676 | 0.2861245613735919 | 0 |
| 13 | 13 | 0.70445225 | -0.7944701742875226 | -1.056933250016024 | 0 |
| 14 | 14 | 1.37811744 | 0.46172156263151365 | 0.34779230478998757 | 0 |
| 15 | 15 | 10.527387... | -0.15726583415965933 | -0.3443850028780142 | 0 |
| 16 | 16 | 1.17046751 | -0.26258861703724534 | -0.4621612947156716 | 0 |
| 17 | 17 | 0.9173048 | 0.14679529308242234 | -0.004371274792487... | 0 |
| 18 | 18 | 11.058689... | 0.49956842967410786 | 0.3901142368180312 | 0 |
| 19 | 19 | 0.64038694 | -0.5907021260252527 | -0.8290713908471448 | 0 |
| 20 | 20 | 0.69810149 | 0.0958177024288176 | -0.061376525033338... | 0 |

STATISTICS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | Auto Outlier Method | seasonal decomposition and super smoother |
| 2 | DETECT_INTERMITTENT_TS | 0 |
| 3 | INTERMITTENT_FLAG | 0 |
| 4 | REMOVE_LEADING_AND_... | 0 |
| 5 | TAKING_LOG_FLAG | 0 |
| 6 | Periods | 3 |
| 7 | ACFs | 0.786964 |
| 8 | OutlierNum | 1 |
| 9 | Mean | 0.150704 |
| 10 | Standard Deviation | 0.894261 |
| 11 | HandleZero | 0 |

**Example 2: The time series is intermittent**

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_OUTLIER_DATA_TBL;
CREATE COLUMN TABLE PAL_OUTLIER_DATA_TBL (
    "ID" INT,
    "RAWDATA" DOUBLE
);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (1, 0);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (2, 0);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (3, 0);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (4, 0);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (5, 2);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (6, 0);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (7, 0);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (8, 0);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (9, 3);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (10, 5);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (11, 0);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (12, 0);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (13, 0);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (14, 0);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (15, 0);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (16, 0);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (17, 4);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (18, 0);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (19, 1);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (20, 0);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('AUTO', 1,NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('DETECT_INTERMITTENT_TS', 1,NULL,NULL);
CALL "_SYS_AFL"."PAL_OUTLIER_DETECTION_FOR_TIME_SERIES"(PAL_OUTLIER_DATA_TBL,
#PAL_PARAMETER_TBL, ?, ?, ?);
```

**Expected Result**

RESULT:

| | TIMESTAMP | RAW_DATA | RESIDUAL | OUTLIER_SCORE | IS_OUTLIER |
|---|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 0 | 0 |
| 2 | 2 | 0 | 0 | 0 | 0 |
| 3 | 3 | 0 | 0 | 0 | 0 |
| 4 | 4 | 0 | 0 | 0 | 0 |
| 5 | 5 | 2 | 0 | 0 | 0 |
| 6 | 6 | 0 | 0 | 0 | 0 |
| 7 | 7 | 0 | 0 | 0 | 0 |
| 8 | 8 | 0 | 0 | 0 | 0 |
| 9 | 9 | 3 | 0 | 0 | 0 |
| 10 | 10 | 5 | 0 | 0 | 0 |
| 11 | 11 | 0 | 0 | 0 | 0 |
| 12 | 12 | 0 | 0 | 0 | 0 |
| 13 | 13 | 0 | 0 | 0 | 0 |
| 14 | 14 | 0 | 0 | 0 | 0 |
| 15 | 15 | 0 | 0 | 0 | 0 |
| 16 | 16 | 0 | 0 | 0 | 0 |
| 17 | 17 | 4 | 0 | 0 | 0 |
| 18 | 18 | 0 | 0 | 0 | 0 |
| 19 | 19 | 1 | 0 | 0 | 0 |
| 20 | 20 | 0 | 0 | 0 | 0 |

STATISTICS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | Auto Outlier Method | intermittent time series, no outlier detection. |
| 2 | DETECT_INTERMITTENT_TS | 1 |
| 3 | INTERMITTENT_FLAG | 1 |
| 4 | REMOVE_LEADING_AND_... | 0 |
| 5 | OutlierNum | 0 |
| 6 | Mean | 0 |
| 7 | Standard Deviation | 1.000000e-08 |
| 8 | HandleZero | 1 |

## Manual Method 1: Residual from Median Filter

This method is applicable to the time series which is smooth.

### Relevant Parameters

- `AUTO`
  To use manual method 1, set `AUTO` to 0.
- `SMOOTH_METHOD`

To use manual method 1, set `SMOOTH_METHOD` to 0 (default value).

- `WINDOW_SIZE`
  To use manual method 1, set `WINDOW_SIZE` to be not less than 3.
- `DETECT_SEASONALITY`
  To use manual method 1, set `DETECT_SEASONALITY` to 0.

### Example

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_OUTLIER_DATA_TBL;
CREATE COLUMN TABLE PAL_OUTLIER_DATA_TBL (
    "ID" INT,
    "RAWDATA" DOUBLE
);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (1, 2.0);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (2, 2.5);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (3, 3.2);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (4, 3.8);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (5, 3.4);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (6, 2.9);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (7, 3.1);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (8, -5);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (9, 3.5);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (10, 3.9);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (11, 4.0);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (12, 4.3);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (13, 4.7);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (14, 5.1);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (15, 5.3);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (16, 13.0);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (17, 4.6);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (18, 4.4);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (19, 4.1);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (20, 3.7);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('AUTO', 0,NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('DETECT_SEASONALITY', 0,NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('OUTLIER_METHOD',0, NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('WINDOW_SIZE',3, NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES('THRESHOLD',NULL,3,NULL);
CALL "_SYS_AFL"."PAL_OUTLIER_DETECTION_FOR_TIME_SERIES"(PAL_OUTLIER_DATA_TBL,
#PAL_PARAMETER_TBL, ?, ?, ?);
```

### Expected Result

RESULT:

| | TIMESTAMP | RAW_DATA | RESIDUAL | OUTLIER_SCORE | IS_OUTLIER |
|---|---|---|---|---|---|
| 1 | 1 | 2 | 0 | -0.00000000000000001730409721395334 2 | 0 |
| 2 | 2 | 2.5 | 0 | -0.00000000000000001730409721395334 2 | 0 |
| 3 | 3 | 3.2 | 0 | -0.00000000000000001730409721395334 2 | 0 |
| 4 | 4 | 3.8 | 0.39999... | 0.15586145152948613 | 0 |
| 5 | 5 | 3.4 | 0 | -0.00000000000000001730409721395334 2 | 0 |
| 6 | 6 | 2.9 | -0.20000... | -0.07793072576474319 | 0 |
| 7 | 7 | 3.1 | 0.20000... | 0.07793072576474315 | 0 |
| 8 | 8 | -5 | -8.1 | -3.1561943934720955 | 1 |
| 9 | 9 | 3.5 | 0 | -0.00000000000000001730409721395334 2 | 0 |
| 10 | 10 | 3.9 | 0 | -0.00000000000000001730409721395334 2 | 0 |
| 11 | 11 | 4 | 0 | -0.00000000000000001730409721395334 2 | 0 |
| 12 | 12 | 4.3 | 0 | -0.00000000000000001730409721395334 2 | 0 |
| 13 | 13 | 4.7 | 0 | -0.00000000000000001730409721395334 2 | 0 |
| 14 | 14 | 5.1 | 0 | -0.00000000000000001730409721395334 2 | 0 |
| 15 | 15 | 5.3 | 0 | -0.00000000000000001730409721395334 2 | 0 |
| 16 | 16 | 13 | 7.7 | 3.0003329419426095 | 1 |
| 17 | 17 | 4.6 | 0 | -0.00000000000000001730409721395334 2 | 0 |
| 18 | 18 | 4.4 | 0 | -0.00000000000000001730409721395334 2 | 0 |
| 19 | 19 | 4.1 | 0 | -0.00000000000000001730409721395334 2 | 0 |
| 20 | 20 | 3.7 | 0 | -0.00000000000000001730409721395334 2 | 0 |

STATISTICS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | DETECT_SEASONALITY | 0 |
| 2 | OutlierNum | 2 |
| 3 | Mean | 4.44089e-17 |
| 4 | Standard Deviation | 2.56638 |
| 5 | HandleZero | 0 |

## Manual Method 2: Seasonal Decomposition Residual

This method is applicable to the time series which is not smooth, but with strong seasonality.

If the time series is not seasonal, that is, the autocorrelation is below ALPHA, the method will fail to detect outliers. In such case, seasonal decomposition is not applied to the original time series and the residuals are all 0.

**Relevant Parameters**

- AUTO
  To use manual method 2, set AUTO to 0.
- SMOOTH_METHOD
  To use manual method 2, set SMOOTH_METHOD to 0 (default value).
- WINDOW_SIZE
  To use manual method 2, set WINDOW_SIZE to 1.
- DETECT_SEASONALITY
  To use manual method 2, set DETECT_SEASONALITY to 1.
- PERIODS
  If you know the seasonal period, specifying PERIODS can speed up the calculation. If not, do not specify it.
- ALPHA
  The criterion for the autocorrelation coefficient. The value range is (0, 1). A larger value indicates stricter requirements for seasonality. The default value is 0.2 when AUTO = 0.
- EXTRAPOLATION
  It specifies whether to extrapolate the endpoints. The default value 0 which means no. When there's an end-point issue, set EXTRAPOLATION to 1.
- THREAD_RATIO
  THREAD_RATIO is valid when DETECT_SEASONALITY is valid with value 1.

### Examples

#### Example 1: With seasonal decomposition

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_OUTLIER_DATA_TBL;
CREATE COLUMN TABLE PAL_OUTLIER_DATA_TBL (
    "ID" INT,
    "RAWDATA" DOUBLE
);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (1, 0.917022);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (2, 1.22032449);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (3, 10.50011437);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (4, 0.80233257);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (5, 0.64675589);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (6, 10.59233859);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (7, 0.68626021);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (8, 0.84556073);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (9, 10.89676747);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (10, 5.03881673);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (11, 0.91919451);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (12, 11.1852195);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (13, 0.70445225);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (14, 1.37811744);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (15, 10.52738759);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (16, 1.17046751);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (17, 0.9173048);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (18, 11.05868983);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (19, 0.64038694);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (20, 0.69810149);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('AUTO', 0,NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('DETECT_SEASONALITY', 1,NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('OUTLIER_METHOD',0, NULL,NULL);
```

```
INSERT INTO #PAL_PARAMETER_TBL VALUES ('WINDOW_SIZE',1, NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES('THRESHOLD',NULL,3,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('EXTRAPOLATION',1, NULL,NULL);
CALL "_SYS_AFL"."PAL_OUTLIER_DETECTION_FOR_TIME_SERIES"(PAL_OUTLIER_DATA_TBL,
#PAL_PARAMETER_TBL, ?, ?, ?);
```

**Expected Result**

RESULT:

| | TIMESTAMP | RAW_DATA | RESIDUAL | OUTLIER_SCORE | IS_OUTLIER |
|---|---|---|---|---|---|
| 1 | 1 | 0.917022 | -0.5377799990740781 | -0.6807061659558097 | 0 |
| 2 | 2 | 1.22032449 | 0.46934951148148185 | 0.628980852175428 | 0 |
| 3 | 3 | 10.500114... | -0.033542661296295... | -0.02498802005387... | 0 |
| 4 | 4 | 0.80233257 | -0.2828471268518524 | -0.3491874588332469 | 0 |
| 5 | 5 | 0.64675589 | 0.09445884814814764 | 0.141467151265767... | 0 |
| 6 | 6 | 10.592338... | 0.2578204720370385 | 0.3539051702352405 | 0 |
| 7 | 7 | 0.68626021 | -0.4572383868518517 | -0.5759685896647304 | 0 |
| 8 | 8 | 0.84556073 | 0.1642099014814815 | 0.2321725154659036 | 0 |
| 9 | 9 | 10.896767... | -1.0563473946296291 | -1.3550593438411322 | 0 |
| 10 | 10 | 5.03881673 | 2.3184450731481476 | 3.0335737342398468 | 1 |
| 11 | 11 | 0.91919451 | -1.3337037618518521 | -1.7157379157028656 | 0 |
| 12 | 12 | 11.1852195 | 0.5561975253703713 | 0.741919372145929 | 0 |
| 13 | 13 | 0.70445225 | -0.8202562335185188 | -1.0480426934878075 | 0 |
| 14 | 14 | 1.37811744 | 0.6363103214814814 | 0.8460993092376714 | 0 |
| 15 | 15 | 10.527387... | -0.190669811296295... | -0.22931863135473... | 0 |
| 16 | 16 | 1.17046751 | -0.1366978768518523 | -0.15913268094372... | 0 |
| 17 | 17 | 0.9173048 | -0.003337271851851... | 0.014291542324519... | 0 |
| 18 | 18 | 11.058689... | 0.49382941870370445 | 0.6608149076350047 | 0 |
| 19 | 19 | 0.64038694 | -0.5941179001851857 | -0.7539688565439121 | 0 |
| 20 | 20 | 0.69810149 | 0.16937231703709532 | 0.238885801656531... | 0 |

STATISTICS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | DETECT_SEASONALITY | 1 |
| 2 | Type | additive |
| 3 | Period | 3 |
| 4 | ACF | 0.786964 |
| 5 | OutlierNum | 1 |
| 6 | Mean | -0.0143272 |
| 7 | Standard Deviation | 0.768984 |
| 8 | HandleZero | 0 |

**Example 2: Fails to do seasonal decomposition**

For the same time series, if we set the wrong PERIODS, PAL will fail to do seasonal decomposition, and fail to detect the outlier. In example 2, we use the same time series as example 1 and set PERIODS = 4, which is not the real seasonal period. In the RESULT table, all residuals are 0 and we fail to detect the outlier.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_OUTLIER_DATA_TBL;
CREATE COLUMN TABLE PAL_OUTLIER_DATA_TBL (
    "ID" INT,
    "RAWDATA" DOUBLE
);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (1, 0.917022);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (2, 1.22032449);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (3, 10.50011437);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (4, 0.80233257);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (5, 0.64675589);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (6, 10.59233859);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (7, 0.68626021);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (8, 0.84556073);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (9, 10.89676747);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (10, 5.03881673);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (11, 0.91919451);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (12, 11.1852195);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (13, 0.70445225);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (14, 1.37811744);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (15, 10.52738759);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (16, 1.17046751);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (17, 0.9173048);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (18, 11.05868983);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (19, 0.64038694);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (20, 0.69810149);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('AUTO', 0,NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('DETECT_SEASONALITY', 1,NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('OUTLIER_METHOD',0, NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('WINDOW_SIZE',1, NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES('THRESHOLD',NULL,3,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('EXTRAPOLATION',1, NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PERIODS',4, NULL,NULL);
```

```
CALL "_SYS_AFL"."PAL_OUTLIER_DETECTION_FOR_TIME_SERIES"(PAL_OUTLIER_DATA_TBL,
#PAL_PARAMETER_TBL, ?, ?, ?);
```

**Expected Result**

RESULT:

| | TIMESTAMP | RAW_DATA | RESIDUAL | OUTLIER_SCORE | IS_OUTLIER |
|---|---|---|---|---|---|
| 1 | 1 | 0.917022 | 0 | 0 | 0 |
| 2 | 2 | 1.22032449 | 0 | 0 | 0 |
| 3 | 3 | 10.500114... | 0 | 0 | 0 |
| 4 | 4 | 0.80233257 | 0 | 0 | 0 |
| 5 | 5 | 0.64675589 | 0 | 0 | 0 |
| 6 | 6 | 10.592338... | 0 | 0 | 0 |
| 7 | 7 | 0.68626021 | 0 | 0 | 0 |
| 8 | 8 | 0.84556073 | 0 | 0 | 0 |
| 9 | 9 | 10.896767... | 0 | 0 | 0 |
| 10 | 10 | 5.03881673 | 0 | 0 | 0 |
| 11 | 11 | 0.91919451 | 0 | 0 | 0 |
| 12 | 12 | 11.1852195 | 0 | 0 | 0 |
| 13 | 13 | 0.70445225 | 0 | 0 | 0 |
| 14 | 14 | 1.37811744 | 0 | 0 | 0 |
| 15 | 15 | 10.527387... | 0 | 0 | 0 |
| 16 | 16 | 1.17046751 | 0 | 0 | 0 |
| 17 | 17 | 0.9173048 | 0 | 0 | 0 |
| 18 | 18 | 11.058689... | 0 | 0 | 0 |
| 19 | 19 | 0.64038694 | 0 | 0 | 0 |
| 20 | 20 | 0.69810149 | 0 | 0 | 0 |

STATISTICS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | DETECT_SEASONALITY | 1 |
| 2 | Type | non-seasonal |
| 3 | Period | 0 |
| 4 | ACF | -0.339406 |
| 5 | OutlierNum | 0 |
| 6 | Mean | 0 |
| 7 | Standard Deviation | 1.000000e-... |
| 8 | HandleZero | 1 |

## Manual Method 3: Residual from Median Filter and Seasonal Decomposition

This method is applicable to the time series which is both smooth and seasonal.

If the time series is not seasonal, that is, the autocorrelation is below ALPHA, the method only applies the median filter to the original time series.

### Relevant Parameters

- `AUTO`
  To use manual method 3, set `AUTO` to 0.
- `SMOOTH_METHOD`
  To use manual method 3, set `SMOOTH_METHOD` to 0 (default value).
- `WINDOW_SIZE`
  To use method 3, set `WINDOW_SIZE` to be not less than 3.
- `DETECT_SEASONALITY`
  To use method 3, set `DETECT_SEASONALITY` to 1.
- `PERIODS`
  If you know the seasonal period, specifying PERIODS can speed up the calculation. If not, do not specify it.
- `ALPHA`
  The criterion for the autocorrelation coefficient. The value range is (0, 1).
  A larger value indicates stricter requirements for seasonality. The default value is 0.2.
- `EXTRAPOLATION`
  It specifies whether to extrapolate the endpoints. The default value 0 which means no. When there's an end-point issue, set `EXTRAPOLATION` to 1.
- `THREAD_RATIO`
  `THREAD_RATIO` is valid when `DETECT_SEASONALITY` is valid with value 1.

### Example

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_OUTLIER_DATA_TBL;
CREATE COLUMN TABLE PAL_OUTLIER_DATA_TBL (
    "ID" INT,
```

```
    "RAWDATA" DOUBLE
);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (1, 0.9834044);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (2, 2.7196354);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (3, 3.80213591);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (4, 4.36257955);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (5, 4.10492168);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (6, 3.41846772);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (7, 2.76168154);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (8, 2.56699911);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (9, 3.07724046);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (10, 4.33219284);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (11, 5.9838389);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (12, 7.7126144);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (13, 8.84300348);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (14, 9.47773652);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (15, 9.08104802);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (16, 8.5340935);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (17, 7.80789046);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (18, 7.60962493);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (19, 3.02596436);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (20, 9.26404979);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (21, 11.06014891);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (22, 12.76922282);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (23, 13.86479787);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (24, 14.44057756);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (25, 14.25084834);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (26, 13.57892133);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (27, 12.74143834);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (28, 12.50569792);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (29, 13.03185305);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (30, 14.400058);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (31, 22.91966937);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (32, 17.65979203);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (33, 18.99369094);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (34, 19.40874609);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (35, 19.21394593);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (36, 18.46310313);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (37, 17.86172968);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (38, 17.6648121);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (39, 18.00154462);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (40, 19.37445836);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('AUTO', 0,NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('DETECT_SEASONALITY', 1,NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('OUTLIER_METHOD',0, NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('WINDOW_SIZE',3, NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES('THRESHOLD',NULL,3,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('EXTRAPOLATION',1, NULL,NULL);
CALL "_SYS_AFL"."PAL_OUTLIER_DETECTION_FOR_TIME_SERIES"(PAL_OUTLIER_DATA_TBL,
#PAL_PARAMETER_TBL, ?, ?, ?);
```

### Expected Result

RESULT:

| | TIMESTAMP | RAW_DATA | RESIDUAL | OUTLIER_SCORE | IS_OUTLIER |
|---|---|---|---|---|---|
| 1 | 1 | 0.9834044 | -0.415655666959084 | -0.2876611693624525 | 0 |
| 2 | 2 | 2.7196354 | -0.30209087437953985 | -0.198405090064265 | 0 |
| 3 | 3 | 3.80213591 | -0.04168524267499607 | 0.0062603276692763 | 0 |
| 4 | 4 | 4.36257955 | 0.23479893677954822 | 0.223562656019006... | 0 |
| 5 | 5 | 4.10492168 | -0.020756734015907... | 0.022709057826204... | 0 |
| 6 | 6 | 3.41846772 | 0.0012417926340967... | 0.039998767026709... | 0 |
| 7 | 7 | 2.76168154 | 0.09087760428182623 | 0.110447899461688... | 0 |
| 8 | 8 | 2.56699911 | -0.03385992757044498 | 0.012410622400945... | 0 |
| 9 | 9 | 3.07724046 | 0.2965925374522853 | 0.272129244276570... | 0 |
| 10 | 10 | 4.33219284 | 0.14778461922501562 | 0.155173882955514... | 0 |
| 11 | 11 | 5.9838389 | -0.3924260104681922 | -0.2694038555002497 | 0 |
| 12 | 12 | 7.7126144 | -0.29776855716137485 | -0.19500797129520... | 0 |
| 13 | 13 | 8.84300348 | 0.0124579247704434... | 0.048814069345688... | 0 |
| 14 | 14 | 9.47773652 | 0.39085500695226294 | 0.3462146915479148 | 0 |
| 15 | 15 | 9.08104802 | 0.0196965308840812... | 0.054503241021272... | 0 |
| 16 | 16 | 8.5340935 | 0.1670184711340966 | 0.1702907001680677 | 0 |
| 17 | 17 | 7.80789046 | 0.18026217628182684 | 0.180699570470363... | 0 |
| 18 | 18 | 7.60962493 | 0.0497137324295553... | 0.078095215792643... | 0 |
| 19 | 19 | 3.02596436 | -4.722465762547715 | -3.672592138569674 | 1 |
| 20 | 20 | 9.26404979 | 0.09249197122501474 | 0.111716708752179... | 0 |
| 21 | 21 | 11.060148... | -0.3094093999681908 | -0.20415707433200... | 0 |
| 22 | 22 | 12.769222... | -0.2252805871613739 | -0.13803615942003... | 0 |
| 23 | 23 | 13.864797... | 0.0415714157704447... | 0.071695774162108... | 0 |
| 24 | 24 | 14.440577... | 0.3023294644522636 | 0.2766381734931611 | 0 |
| 25 | 25 | 14.250848... | 0.08343052288408082 | 0.104594876947587... | 0 |
| 26 | 26 | 13.578921... | 0.0194057546340971... | 0.054274705861412... | 0 |
| 27 | 27 | 12.741438... | -0.21319013071817272 | -0.1285336832165048 | 0 |
| 28 | 28 | 12.505697... | -0.44496138357044357 | -0.3106939536113349 | 0 |
| 29 | 29 | 13.031853... | -0.10234497604771331 | -0.04141509927547... | 0 |
| 30 | 30 | 14.400058 | -0.13659744927498285 | -0.06833577950646... | 0 |
| 31 | 31 | 22.919669... | 6.203380867531809 | 4.914560719840693 | 1 |
| 32 | 32 | 17.659792... | -0.6827464411613744 | -0.49758076486839... | 0 |
| 33 | 33 | 18.993690... | -0.1863316477295529 | -0.10742429858233... | 0 |
| 34 | 34 | 19.408746... | -0.06309520804773605 | -0.01056680267345... | 0 |
| 35 | 35 | 19.213945... | -0.2563578696159148 | -0.1624613032003266 | 0 |
| 36 | 36 | 18.463103... | -0.3616535682659513 | -0.24521830046145... | 0 |
| 37 | 37 | 17.861729... | -0.2319371997091415 | -0.14326791412080... | 0 |
| 38 | 38 | 17.6648121 | -0.37222040115232913 | -0.2535232870036211 | 0 |
| 39 | 39 | 18.001544... | -0.2294299187205171 | -0.1412973203684657 | 0 |
| 40 | 40 | 19.374458... | -0.2776666910387082 | -0.17920893960649... | 0 |

STATISTICS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | DETECT_SEASONALITY | 1 |
| 2 | Type | additive |
| 3 | Period | 10 |
| 4 | ACF | 0.705896 |
| 5 | OutlierNum | 2 |
| 6 | Mean | -0.0496505 |
| 7 | Standard Deviation | 1.27234 |
| 8 | HandleZero | 0 |

## Manual Method 4:Residual from LOESS

This method is applicable to the time series which is non-seasonal.

### Relevant Parameters

- `AUTO`
  To use manual method 4, set `AUTO` to 0.
- `SMOOTH_METHOD`
  To use manual method 4, set `SMOOTH_METHOD` to 1.
- `LOESS_LAG`
  To use manual method 4, set `LOESS_LAG` to be not less than 3. The default value is 7.
- `CURRENT_VALUE_FLAG`
  Whether to take the current data point when using LOESS smoothing method. 0 for not taking the current data point, 1 for taking the current data point. The default value is 0.

### Example

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_OUTLIER_DATA_TBL;
CREATE COLUMN TABLE PAL_OUTLIER_DATA_TBL (
    "ID" INT,
    "RAWDATA" DOUBLE
);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (1, 2.0);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (2, 2.5);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (3, 3.2);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (4, 3.8);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (5, 3.4);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (6, 2.9);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (7, 3.1);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (8, -5);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (9, 3.5);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (10, 3.9);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (11, 4.0);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (12, 4.3);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (13, 4.7);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (14, 5.1);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (15, 5.3);
```

```
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (16, 13.0);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (17, 4.6);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (18, 4.4);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (19, 4.1);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (20, 3.7);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('AUTO', 0,NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SMOOTH_METHOD', 1,NULL,NULL);
CALL "_SYS_AFL"."PAL_OUTLIER_DETECTION_FOR_TIME_SERIES"(PAL_OUTLIER_DATA_TBL,
#PAL_PARAMETER_TBL, ?, ?, ?);
```

**Expected Result**

RESULT:

| | TIMESTAMP | RAW_DATA | RESIDUAL | OUTLIER_SCORE | IS_OUTLIER |
|---|---|---|---|---|---|
| 1 | 1 | 2 | -0.3738874649562134 | -0.1433870910836566 | 0 |
| 2 | 2 | 2.5 | -0.11012714518240685 | -0.04506165519377258 | 0 |
| 3 | 3 | 3.2 | 0.30049249731243055 | 0.10801048002306801 | 0 |
| 4 | 4 | 3.8 | 0.6666158152658634 | 0.2444951296401667 | 0 |
| 5 | 5 | 3.4 | 0.12638379485286233 | 0.04310566766827481 | 0 |
| 6 | 6 | 2.9 | -0.35846928672599043 | -0.1376394521043944 | 0 |
| 7 | 7 | 3.1 | -0.03889119898099613 | -0.018506085852301... | 0 |
| 8 | 8 | -5 | -8.344335519154058 | -3.1146368220468212 | 1 |
| 9 | 9 | 3.5 | -0.12117982863382881 | -0.049181910731246... | 0 |
| 10 | 10 | 3.9 | 0.14832570027743275 | 0.05128524303797281 | 0 |
| 11 | 11 | 4 | -0.10542042471052504 | -0.043307068520513... | 0 |
| 12 | 12 | 4.3 | -0.08869526947989304 | -0.037072210374344... | 0 |
| 13 | 13 | 4.7 | 0.0080439246518215... | -0.001009456739551... | 0 |
| 14 | 14 | 5.1 | 0.10571793989297085 | 0.0354017825702016 | 0 |
| 15 | 15 | 5.3 | 0.39623518759872844 | 0.14370175369055346 | 0 |
| 16 | 16 | 13 | 8.124548931061458 | 3.0246878632441603 | 1 |
| 17 | 17 | 4.6 | -0.09999997161233676 | -0.041286414261157... | 0 |
| 18 | 18 | 4.4 | 0.0445628561016402... | 0.01260419013544254 | 0 |
| 19 | 19 | 4.1 | 0.06307408719001639 | 0.01950486748973088 | 0 |
| 20 | 20 | 3.7 | -0.12795829759485144 | -0.05170881059181014 | 0 |

STATISTICS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | DETECT_SEASONALITY | 0 |
| 2 | OutlierNum | 2 |
| 3 | Mean | 0.0107518 |
| 4 | Standard Deviation | 2.68252 |
| 5 | HandleZero | 0 |

## Manual Method 5: Residual from Super Smoother

This method is applicable to the time series which is non-seasonal.

### Relevant Parameters

- `AUTO`
  To use manual method 5, set `AUTO` to 0.
- `SMOOTH_METHOD`
  To use manual method 5, set `SMOOTH_METHOD` to 2.

### Example

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_OUTLIER_DATA_TBL;
CREATE COLUMN TABLE PAL_OUTLIER_DATA_TBL (
    "ID" INT,
    "RAWDATA" DOUBLE
);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (1, 2.0);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (2, 2.5);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (3, 3.2);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (4, 3.8);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (5, 3.4);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (6, 2.9);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (7, 3.1);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (8, -5);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (9, 3.5);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (10, 3.9);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (11, 4.0);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (12, 4.3);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (13, 4.7);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (14, 5.1);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (15, 5.3);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (16, 13.0);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (17, 4.6);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (18, 4.4);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (19, 4.1);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (20, 3.7);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('AUTO', 0,NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SMOOTH_METHOD', 2,NULL,NULL);
CALL "_SYS_AFL"."PAL_OUTLIER_DETECTION_FOR_TIME_SERIES"(PAL_OUTLIER_DATA_TBL,
#PAL_PARAMETER_TBL, ?, ?, ?);
```

### Expected Result

RESULT:

| | TIMESTAMP | RAW_DATA | RESIDUAL | OUTLIER_SCORE | IS_OUTLIER |
|---|---|---|---|---|---|
| 1 | 1 | 2 | 0.08669801848758629 | 0.03549036248620932 | 0 |
| 2 | 2 | 2.5 | 0.007392351518096696 | 0.005851297502800426 | 0 |
| 3 | 3 | 3.2 | 0.16458703428706345 | 0.06459997960427243 | 0 |
| 4 | 4 | 3.8 | 0.528646798712102 | 0.20066076098310529 | 0 |
| 5 | 5 | 3.4 | 0.13303986655831768 | 0.05280979373124372 | 0 |
| 6 | 6 | 2.9 | -0.2859393852478975 | -0.1037761571285095 | 0 |
| 7 | 7 | 3.1 | -0.11856289120078056 | -0.0412222064043298... | 0 |
| 8 | 8 | -5 | -8.410964713988811 | -3.140358067511314 | 1 |
| 9 | 9 | 3.5 | -0.12777804561208495 | -0.04466620442951358 | 0 |
| 10 | 10 | 3.9 | 0.057481317481012706 | 0.024571146586192843 | 0 |
| 11 | 11 | 4 | -0.09816034589151457 | -0.0335971224808704... | 0 |
| 12 | 12 | 4.3 | -0.09007649371817372 | -0.0305759282959190... | 0 |
| 13 | 13 | 4.7 | -0.0194765073282523... | -0.0041904546671089... | 0 |
| 14 | 14 | 5.1 | 0.0000000557914718... | 0.0030885600870454... | 0 |
| 15 | 15 | 5.3 | 0.12943763923003537 | 0.05146352613211936 | 0 |
| 16 | 16 | 13 | 8.049472652211092 | 3.0114339972326167 | 1 |
| 17 | 17 | 4.6 | -0.0570299700541943 | -0.01822538514079662 | 0 |
| 18 | 18 | 4.4 | -0.0283831210416511... | -0.0075191411623011... | 0 |
| 19 | 19 | 4.1 | -0.0066080825447176... | 0.0006188873757494... | 0 |
| 20 | 20 | 3.7 | -0.07905714329895774 | -0.0264576445006922... | 0 |

STATISTICS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | DETECT_SEASONALITY | 0 |
| 2 | OutlierNum | 2 |
| 3 | Mean | -0.00826404 |
| 4 | Standard Deviation | 2.67571 |
| 5 | HandleZero | 0 |

## No Smoothing: Skip Step 1

This method is applicable to the time series with neither seasonality nor trend.

### Relevant Parameters

- `AUTO`
  To skip step 1, set `AUTO` to 0.
- `SMOOTH_METHOD`
  To skip step 1, set `SMOOTH_METHOD` to -1.

### Example

```
SET SCHEMA DM_PAL;
```

```
DROP TABLE PAL_OUTLIER_DATA_TBL;
CREATE COLUMN TABLE PAL_OUTLIER_DATA_TBL (
    "ID" INT,
    "RAWDATA" DOUBLE
);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (1, 2.0);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (2, 2.1);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (3, 2.3);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (4, 1.9);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (5, 1.8);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (6, 2.1);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (7, 2.0);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (8, 2.2);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (9, 6.0);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (10, 1.9);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (11, 1.7);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (12, 1.9);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (13, 2.1);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (14, 2.0);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (15, 2.3);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (16, 2.2);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (17, 1.8);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (18, 1.9);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (19, -2.0);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (20, 2.1);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('AUTO', 0,NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SMOOTH_METHOD', -1,NULL,NULL);
CALL "_SYS_AFL"."PAL_OUTLIER_DETECTION_FOR_TIME_SERIES"(PAL_OUTLIER_DATA_TBL,
#PAL_PARAMETER_TBL, ?, ?, ?);
```

### Expected Result

RESULT:

| | TIMESTAMP | RAW_DATA | RESIDUAL | OUTLIER_SCORE | IS_OUTLIER |
|---|---|---|---|---|---|
| 1 | 1 | 2 | 2 | -0.01146805148880689 | 0 |
| 2 | 2 | 2.1 | 2.1 | 0.06498562510324075 | 0 |
| 3 | 3 | 2.3 | 2.3 | 0.2178929782873357 | 0 |
| 4 | 4 | 1.9 | 1.9 | -0.08792172808085454 | 0 |
| 5 | 5 | 1.8 | 1.8 | -0.164375404672902 | 0 |
| 6 | 6 | 2.1 | 2.1 | 0.06498562510324075 | 0 |
| 7 | 7 | 2 | 2 | -0.01146805148880689 | 0 |
| 8 | 8 | 2.2 | 2.2 | 0.1414393016952884 | 0 |
| 9 | 9 | 6 | 6 | 3.046679012193096 | 1 |
| 10 | 10 | 1.9 | 1.9 | -0.08792172808085454 | 0 |
| 11 | 11 | 1.7 | 1.7 | -0.24082908126494965 | 0 |
| 12 | 12 | 1.9 | 1.9 | -0.08792172808085454 | 0 |
| 13 | 13 | 2.1 | 2.1 | 0.06498562510324075 | 0 |
| 14 | 14 | 2 | 2 | -0.01146805148880689 | 0 |
| 15 | 15 | 2.3 | 2.3 | 0.2178929782873357 | 0 |
| 16 | 16 | 2.2 | 2.2 | 0.1414393016952884 | 0 |
| 17 | 17 | 1.8 | 1.8 | -0.164375404672902 | 0 |
| 18 | 18 | 1.9 | 1.9 | -0.08792172808085454 | 0 |
| 19 | 19 | -2 | -2 | -3.0696151151707096 | 1 |
| 20 | 20 | 2.1 | 2.1 | 0.06498562510324075 | 0 |

STATISTICS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | OutlierNum | 2 |
| 2 | Mean | 2.01499 |
| 3 | Standard Deviation | 1.30798 |
| 4 | HandleZero | 0 |

## Parameter Combination Needs to Be Avoided

When `SMOOTH_METHOD` is valid with value 0, the parameter combination {`DETECT_SEASONALITY` = 0, `WINDOW_SIZE` = 1} should be avoided. With such parameter combination, all residuals are 0, no matter what the time series is.

### Example

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_OUTLIER_DATA_TBL;
CREATE COLUMN TABLE PAL_OUTLIER_DATA_TBL (
    "ID" INT,
    "RAWDATA" DOUBLE
);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (1, 2.0);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (2, 2.5);
```

```
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (3, 3.2);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (4, 3.8);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (5, 3.4);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (6, 2.9);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (7, 3.1);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (8, -5);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (9, 3.5);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (10, 3.9);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('AUTO', 0,NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('DETECT_SEASONALITY', 0,NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('OUTLIER_METHOD',0, NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('WINDOW_SIZE',1, NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES('THRESHOLD',NULL,3,NULL);
CALL "_SYS_AFL"."PAL_OUTLIER_DETECTION_FOR_TIME_SERIES"(PAL_OUTLIER_DATA_TBL,
#PAL_PARAMETER_TBL, ?, ?, ?);
```

**Expected Result**

RESULT:

| | TIMESTAMP | RAW_DATA | RESIDUAL | OUTLIER_SCORE | IS_OUTLIER |
|---|---|---|---|---|---|
| 1 | 1 | 2 | 0 | 0 | 0 |
| 2 | 2 | 2.5 | 0 | 0 | 0 |
| 3 | 3 | 3.2 | 0 | 0 | 0 |
| 4 | 4 | 3.8 | 0 | 0 | 0 |
| 5 | 5 | 3.4 | 0 | 0 | 0 |
| 6 | 6 | 2.9 | 0 | 0 | 0 |
| 7 | 7 | 3.1 | 0 | 0 | 0 |
| 8 | 8 | -5 | 0 | 0 | 0 |
| 9 | 9 | 3.5 | 0 | 0 | 0 |
| 10 | 10 | 3.9 | 0 | 0 | 0 |

STATISTICS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | DETECT_SEASONALITY | 0 |
| 2 | OutlierNum | 0 |
| 3 | Mean | 0 |
| 4 | Standard Deviation | 1.000000e-... |
| 5 | HandleZero | 1 |

## 5.5.24.2 Detect Outlier from Residual

After getting the residual series, we detect outliers from the residual series.

In PAL, we have 6 methods. They are Z1 score, Z2 score, IQR (Inter-Quartile Range) score, MAD (Median Absolute Deviation) score, isolation forest score and DBSCAN (Density-Based Spatial Clustering of Applications with Noise). In this file, we will give the features of each method and the corresponding parameter choice. For each method, we will also provide at least one example.

You can also use some of the previous methods and get the outlier result by voting.

### Z1 score

Z1 score method is a general outlier detection method in statistic. It needs a threshold. Usually we set the threshold to 3. Z1 score method usually works well. In the file Methods to Get Residual in Outlier Detection [page 846], we use Z1 score in all the examples.

In Z1 score method, we use the mean of the residual. However, may Outlier points may deviate the mean from the statistical mean, and cause outlier detection mistakes. Please see the example below.

**Relevant Parameters**

- OUTLIER_METHOD
  To use Z1 score method, you need to set OUTLIER_METHOD = 0 (default value).
- THRESHOLD
  To use Z1 score method, we usually set THRESHOLD = 3 (default value for Z1 score). A larger value indicates stricter requirement for outlier, that is fewer outlier points.

**Example**

In the example below, the mean in STATISTICS table deviates from 0. It leads the miss detection of outliers.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_OUTLIER_DATA_TBL;
CREATE COLUMN TABLE PAL_OUTLIER_DATA_TBL (
    "ID" INT,
    "RAWDATA" DOUBLE
);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (1, 2.0);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (2, 2.5);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (3, 3.2);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (4, 3.8);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (5, 3.4);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (6, 2.9);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (7, 3.1);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (8, 11);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (9, 3.5);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (10, 3.9);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (11, 4.0);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (12, 4.3);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (13, 4.7);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (14, 5.1);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (15, 5.3);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (16, 13.0);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (17, 4.6);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (18, 4.4);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (19, 4.1);
```

```
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (20, 3.7);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('AUTO', 0,NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('DETECT_SEASONALITY', 0,NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('OUTLIER_METHOD',0, NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('WINDOW_SIZE',3, NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES('THRESHOLD',NULL,3,NULL);
CALL "_SYS_AFL"."PAL_OUTLIER_DETECTION_FOR_TIME_SERIES"(PAL_OUTLIER_DATA_TBL,
#PAL_PARAMETER_TBL, ?, ?, ?);
```

**Expected Result**

RESULT:

| | TIMESTAMP | RAW_DATA | RESIDUAL | OUTLIER_SCORE | IS_OUTLIER |
|---|---|---|---|---|---|
| 1 | 1 | 2 | 0 | -0.31957179902931104 | 0 |
| 2 | 2 | 2.5 | 0 | -0.31957179902931104 | 0 |
| 3 | 3 | 3.2 | 0 | -0.31957179902931104 | 0 |
| 4 | 4 | 3.8 | 0.39999... | -0.14913350621367852 | 0 |
| 5 | 5 | 3.4 | 0 | -0.31957179902931104 | 0 |
| 6 | 6 | 2.9 | -0.20000... | -0.4047909454371274 | 0 |
| 7 | 7 | 3.1 | 0 | -0.31957179902931104 | 0 |
| 8 | 8 | 11 | 7.5 | 2.8761461912637993 | 0 |
| 9 | 9 | 3.5 | -0.39999... | -0.49001009184494354 | 0 |
| 10 | 10 | 3.9 | 0 | -0.31957179902931104 | 0 |
| 11 | 11 | 4 | 0 | -0.31957179902931104 | 0 |
| 12 | 12 | 4.3 | 0 | -0.31957179902931104 | 0 |
| 13 | 13 | 4.7 | 0 | -0.31957179902931104 | 0 |
| 14 | 14 | 5.1 | 0 | -0.31957179902931104 | 0 |
| 15 | 15 | 5.3 | 0 | -0.31957179902931104 | 0 |
| 16 | 16 | 13 | 7.7 | 2.961365337671616 | 0 |
| 17 | 17 | 4.6 | 0 | -0.31957179902931104 | 0 |
| 18 | 18 | 4.4 | 0 | -0.31957179902931104 | 0 |
| 19 | 19 | 4.1 | 0 | -0.31957179902931104 | 0 |
| 20 | 20 | 3.7 | 0 | -0.31957179902931104 | 0 |

STATISTICS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | DETECT_SEASONALITY | 0 |
| 2 | OutlierNum | 0 |
| 3 | Mean | 0.75 |
| 4 | Standard Deviation | 2.34689 |
| 5 | HandleZero | 0 |

## Z2 Score

Z2 score method is a modification of Z1 score method. It replaces the sample mean by the statistical mean (0 for additive residual, 1 for multiplicative residual). It also needs a threshold. Usually we set the threshold to 3.

### Relevant Parameters

- OUTLIER_METHOD
  To use Z2 score method, you need to set OUTLIER_METHOD = 1.
- THRESHOLD
  To use Z2 score method, we usually set THRESHOLD = 3 (default value for Z2 score). A larger value indicates stricter requirement for outlier, that is, less outlier points.

### Example

In the example below, we use the same time series as the example of Z1 score. With Z2 score, we find outliers correctly.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_OUTLIER_DATA_TBL;
CREATE COLUMN TABLE PAL_OUTLIER_DATA_TBL (
    "ID" INT,
    "RAWDATA" DOUBLE
);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (1, 2.0);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (2, 2.5);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (3, 3.2);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (4, 3.8);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (5, 3.4);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (6, 2.9);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (7, 3.1);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (8, 11);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (9, 3.5);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (10, 3.9);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (11, 4.0);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (12, 4.3);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (13, 4.7);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (14, 5.1);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (15, 5.3);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (16, 13.0);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (17, 4.6);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (18, 4.4);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (19, 4.1);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (20, 3.7);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
```

```
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('AUTO', 0,NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('DETECT_SEASONALITY', 0,NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('OUTLIER_METHOD',1, NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('WINDOW_SIZE',3, NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES('THRESHOLD',NULL,3,NULL);
CALL "_SYS_AFL"."PAL_OUTLIER_DETECTION_FOR_TIME_SERIES"(PAL_OUTLIER_DATA_TBL,
#PAL_PARAMETER_TBL, ?, ?, ?);
```

**Expected Result**

RESULT:

| | TIMESTAMP | RAW_DATA | RESIDUAL | OUTLIER_SCORE | IS_OUTLIER |
|---|---|---|---|---|---|
| 1 | 1 | 2 | 0 | 0 | 0 |
| 2 | 2 | 2.5 | 0 | 0 | 0 |
| 3 | 3 | 3.2 | 0 | 0 | 0 |
| 4 | 4 | 3.8 | 0.39999... | 0.17043829281... | 0 |
| 5 | 5 | 3.4 | 0 | 0 | 0 |
| 6 | 6 | 2.9 | -0.20000... | -0.0852191464... | 0 |
| 7 | 7 | 3.1 | 0 | 0 | 0 |
| 8 | 8 | 11 | 7.5 | 3.19571799029... | 1 |
| 9 | 9 | 3.5 | -0.39999... | -0.1704382928... | 0 |
| 10 | 10 | 3.9 | 0 | 0 | 0 |
| 11 | 11 | 4 | 0 | 0 | 0 |
| 12 | 12 | 4.3 | 0 | 0 | 0 |
| 13 | 13 | 4.7 | 0 | 0 | 0 |
| 14 | 14 | 5.1 | 0 | 0 | 0 |
| 15 | 15 | 5.3 | 0 | 0 | 0 |
| 16 | 16 | 13 | 7.7 | 3.28093713670... | 1 |
| 17 | 17 | 4.6 | 0 | 0 | 0 |
| 18 | 18 | 4.4 | 0 | 0 | 0 |
| 19 | 19 | 4.1 | 0 | 0 | 0 |
| 20 | 20 | 3.7 | 0 | 0 | 0 |

STATISTICS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | DETECT_SEASONALITY | 0 |
| 2 | OutlierNum | 2 |
| 3 | Standard Deviation | 2.34689 |
| 4 | HandleZero | 0 |

## IQR Score

IQR score method is a common statistic method to detect outlier. Unlike the mean and standard deviation in Z1 score method and Z2 score method, Q1, Q3 and IQR are not affected by the extreme (highest and lowest) values in the residual. Q1, Q3 and IQR are robust statistic and outliers can not affect their values. In this way, it is robust in detecting outliers. It also needs a threshold. Usually we set the threshold to 1.5 (default value for IQR score).

However, IQR score method also has its shortcoming. There may be a zero denominator problem. When the time series is very smooth and we only apply median filter, there will be many 0 residual points and IQR may be 0. In this case, zero denominator problem happens. All data points with residual not equal to 0 will be considered as outliers. We suggest using the IQR score method with the time series which is not so smooth, or the case with seasonal decomposition.

### Relevant Parameters

- OUTLIER_METHOD
  To use IQR score method, you need to set OUTLIER_METHOD = 2.
- THRESHOLD
  To use IQR score method, we usually set THRESHOLD = 1.5 (default value for IQR score). A larger value indicates stricter requirement for outlier, that is, fewer outlier points.

### Examples

#### Example 1: Smooth series without seasonal decomposition

In the example below, we use the same time series as the example of Z1 score and Z2 score and only apply the median filter. The series is quite smooth and we find that IQR is 0 and there are many false alarms.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_OUTLIER_DATA_TBL;
CREATE COLUMN TABLE PAL_OUTLIER_DATA_TBL (
    "ID" INT,
    "RAWDATA" DOUBLE
);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (1, 2.0);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (2, 2.5);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (3, 3.2);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (4, 3.8);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (5, 3.4);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (6, 2.9);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (7, 3.1);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (8, 11);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (9, 3.5);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (10, 3.9);
```

```
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (11, 4.0);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (12, 4.3);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (13, 4.7);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (14, 5.1);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (15, 5.3);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (16, 13.0);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (17, 4.6);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (18, 4.4);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (19, 4.1);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (20, 3.7);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('AUTO', 0,NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('DETECT_SEASONALITY', 0,NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('OUTLIER_METHOD',2, NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('WINDOW_SIZE',3, NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES('THRESHOLD',NULL,1.5,NULL);
CALL "_SYS_AFL"."PAL_OUTLIER_DETECTION_FOR_TIME_SERIES"(PAL_OUTLIER_DATA_TBL,
#PAL_PARAMETER_TBL, ?, ?, ?);
```

### Expected Result

RESULT:

| | TIMESTAMP | RAW_DATA | RESIDUAL | OUTLIER_SCORE | IS_OUTLIER |
|---|---|---|---|---|---|
| 1 | 1 | 2 | 0 | 0 | 0 |
| 2 | 2 | 2.5 | 0 | 0 | 0 |
| 3 | 3 | 3.2 | 0 | 0 | 0 |
| 4 | 4 | 3.8 | 0.39999... | 39,999,999.999... | 1 |
| 5 | 5 | 3.4 | 0 | 0 | 0 |
| 6 | 6 | 2.9 | -0.20000... | -20,000,000.00... | 1 |
| 7 | 7 | 3.1 | 0 | 0 | 0 |
| 8 | 8 | 11 | 7.5 | 750,000,000 | 1 |
| 9 | 9 | 3.5 | -0.39999... | -39,999,999.99... | 1 |
| 10 | 10 | 3.9 | 0 | 0 | 0 |
| 11 | 11 | 4 | 0 | 0 | 0 |
| 12 | 12 | 4.3 | 0 | 0 | 0 |
| 13 | 13 | 4.7 | 0 | 0 | 0 |
| 14 | 14 | 5.1 | 0 | 0 | 0 |
| 15 | 15 | 5.3 | 0 | 0 | 0 |
| 16 | 16 | 13 | 7.7 | 770,000,000 | 1 |
| 17 | 17 | 4.6 | 0 | 0 | 0 |
| 18 | 18 | 4.4 | 0 | 0 | 0 |
| 19 | 19 | 4.1 | 0 | 0 | 0 |
| 20 | 20 | 3.7 | 0 | 0 | 0 |

STATISTICS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | DETECT_SEASONALITY | 0 |
| 2 | OutlierNum | 5 |
| 3 | Q1 | 0 |
| 4 | Q3 | 0 |
| 5 | IQR | 1.000000e-... |
| 6 | HandleZero | 1 |

**Example 2: Series with seasonal decomposition**

In the example below, we take a time series with seasonality and do the seasonal decomposition. We find that the IQR score method works well.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_OUTLIER_DATA_TBL;
```

```
CREATE COLUMN TABLE PAL_OUTLIER_DATA_TBL (
    "ID" INT,
    "RAWDATA" DOUBLE
);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (1, 0.917022);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (2, 1.22032449);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (3, 10.50011437);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (4, 0.80233257);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (5, 0.64675589);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (6, 10.59233859);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (7, 0.68626021);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (8, 0.84556073);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (9, 10.89676747);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (10, 5.03881673);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (11, 0.91919451);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (12, 11.1852195);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (13, 0.70445225);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (14, 1.37811744);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (15, 10.52738759);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (16, 1.17046751);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (17, 0.9173048);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (18, 11.05868983);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (19, 0.64038694);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (20, 0.69810149);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('AUTO', 0,NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('DETECT_SEASONALITY', 1,NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('OUTLIER_METHOD',2, NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('WINDOW_SIZE',1, NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES('THRESHOLD',NULL,1.5,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('EXTRAPOLATION',1, NULL,NULL);
CALL "_SYS_AFL"."PAL_OUTLIER_DETECTION_FOR_TIME_SERIES"(PAL_OUTLIER_DATA_TBL,
#PAL_PARAMETER_TBL, ?, ?, ?);
```

### Expected Result

RESULT:

| | TIMESTAMP | RAW_DATA | RESIDUAL | OUTLIER_SCORE | IS_OUTLIER |
|---|---|---|---|---|---|
| 1 | 1 | 0.917022 | -0.5377799990740781 | -0.04676701669295784 | 0 |
| 2 | 2 | 1.22032449 | 0.46934951148148185 | 0.12282572754377569 | 0 |
| 3 | 3 | 10.500114... | -0.033542661296295... | 0 | 0 |
| 4 | 4 | 0.80233257 | -0.2828471268518524 | 0 | 0 |
| 5 | 5 | 0.64675589 | 0.09445884814814764 | 0 | 0 |
| 6 | 6 | 10.592338... | 0.2578204720370385 | 0 | 0 |
| 7 | 7 | 0.68626021 | -0.4572383868518517 | 0 | 0 |
| 8 | 8 | 0.84556073 | 0.1642099014814815 | 0 | 0 |
| 9 | 9 | 10.896767... | -1.0563473946296291 | -0.6489861522487651 | 0 |
| 10 | 10 | 5.03881673 | 2.3184450731481476 | 2.270204718685324 | 1 |
| 11 | 11 | 0.91919451 | -1.3337037618518521 | -0.9710837487058747 | 0 |
| 12 | 12 | 11.1852195 | 0.5561975253703713 | 0.22368346811358938 | 0 |
| 13 | 13 | 0.70445225 | -0.8202562335185188 | -0.37481038228084984 | 0 |
| 14 | 14 | 1.37811744 | 0.6363103214814814 | 0.3167195117107868 | 0 |
| 15 | 15 | 10.527387... | -0.190669811296295... | 0 | 0 |
| 16 | 16 | 1.17046751 | -0.1366978768518523 | 0 | 0 |
| 17 | 17 | 0.9173048 | -0.003337271851851... | 0 | 0 |
| 18 | 18 | 11.058689... | 0.49382941870370445 | 0.15125456570869653 | 0 |
| 19 | 19 | 0.64038694 | -0.5941179001851857 | -0.11219296208972217 | 0 |
| 20 | 20 | 0.69810149 | 0.16937231703709532 | 0 | 0 |

STATISTICS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | DETECT_SEASONALITY | 1 |
| 2 | Type | additive |
| 3 | Period | 3 |
| 4 | ACF | 0.786964 |
| 5 | OutlierNum | 1 |
| 6 | Q1 | -0.497509 |
| 7 | Q3 | 0.363584 |
| 8 | IQR | 0.861094 |
| 9 | HandleZero | 0 |

## MAD Score

MAD score method is a common statistic method to detect outlier. Similar to Q1, Q3 and IQR in IQR score, median and MAD are also robust statistic and outliers can not affect their values. It also needs a threshold. Usually we set the threshold to 3.

Similar to IQR score, there may also be a zero denominator problem in the MAD score method. The MAD score method is also not suitable for smooth time series only with median filter. We suggest using the MAD score method with the time series which is not so smooth, or the case with seasonal decomposition.

### Relevant Parameters

- OUTLIER_METHOD
  To use MAD score method, you need to set OUTLIER_METHOD = 3.
- THRESHOLD
  To use MAD score method, we usually set THRESHOLD = 3 (default value for MAD score). A larger value indicates stricter requirement for outlier, that is, fewer outlier points.

### Examples

#### Example 1: Smooth series without seasonal decomposition

In the example below, we use the same time series as the example of Z1 score and Z2 score and only apply the median filter. The series is quite smooth and we find that MAD is 0 and there are many false alarms.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_OUTLIER_DATA_TBL;
CREATE COLUMN TABLE PAL_OUTLIER_DATA_TBL (
    "ID" INT,
    "RAWDATA" DOUBLE
);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (1, 2.0);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (2, 2.5);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (3, 3.2);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (4, 3.8);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (5, 3.4);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (6, 2.9);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (7, 3.1);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (8, 11);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (9, 3.5);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (10, 3.9);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (11, 4.0);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (12, 4.3);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (13, 4.7);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (14, 5.1);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (15, 5.3);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (16, 13.0);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (17, 4.6);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (18, 4.4);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (19, 4.1);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (20, 3.7);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('AUTO', 0,NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('DETECT_SEASONALITY', 0,NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('OUTLIER_METHOD',3, NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('WINDOW_SIZE',3, NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES('THRESHOLD',NULL,3,NULL);
```

```
CALL "_SYS_AFL"."PAL_OUTLIER_DETECTION_FOR_TIME_SERIES"(PAL_OUTLIER_DATA_TBL,
#PAL_PARAMETER_TBL, ?, ?, ?);
```

**Expected Result**

RESULT:

| | TIMESTAMP | RAW_DATA | RESIDUAL | OUTLIER_SCORE | IS_OUTLIER |
|---|---|---|---|---|---|
| 1 | 1 | 2 | 0 | 0 | 0 |
| 2 | 2 | 2.5 | 0 | 0 | 0 |
| 3 | 3 | 3.2 | 0 | 0 | 0 |
| 4 | 4 | 3.8 | 0.39999... | 26,979,599.999... | 1 |
| 5 | 5 | 3.4 | 0 | 0 | 0 |
| 6 | 6 | 2.9 | -0.20000... | -13,489,800.00... | 1 |
| 7 | 7 | 3.1 | 0 | 0 | 0 |
| 8 | 8 | 11 | 7.5 | 505,867,500 | 1 |
| 9 | 9 | 3.5 | -0.39999... | -26,979,599.99... | 1 |
| 10 | 10 | 3.9 | 0 | 0 | 0 |
| 11 | 11 | 4 | 0 | 0 | 0 |
| 12 | 12 | 4.3 | 0 | 0 | 0 |
| 13 | 13 | 4.7 | 0 | 0 | 0 |
| 14 | 14 | 5.1 | 0 | 0 | 0 |
| 15 | 15 | 5.3 | 0 | 0 | 0 |
| 16 | 16 | 13 | 7.7 | 519,357,300.00... | 1 |
| 17 | 17 | 4.6 | 0 | 0 | 0 |
| 18 | 18 | 4.4 | 0 | 0 | 0 |
| 19 | 19 | 4.1 | 0 | 0 | 0 |
| 20 | 20 | 3.7 | 0 | 0 | 0 |

STATISTICS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | DETECT_SEASONALITY | 0 |
| 2 | OutlierNum | 5 |
| 3 | Median | 0 |
| 4 | Median Absolute De... | 1.000000e-... |
| 5 | HandleZero | 1 |

**Example 2: Series with seasonal decomposition**

In the example below, we take a time series with seasonality and do the seasonal decomposition. We find that the MAD score method works well.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_OUTLIER_DATA_TBL;
CREATE COLUMN TABLE PAL_OUTLIER_DATA_TBL (
    "ID" INT,
    "RAWDATA" DOUBLE
);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (1, 0.917022);
```

```
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (2, 1.22032449);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (3, 10.50011437);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (4, 0.80233257);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (5, 0.64675589);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (6, 10.59233859);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (7, 0.68626021);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (8, 0.84556073);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (9, 10.89676747);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (10, 5.03881673);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (11, 0.91919451);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (12, 11.1852195);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (13, 0.70445225);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (14, 1.37811744);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (15, 10.52738759);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (16, 1.17046751);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (17, 0.9173048);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (18, 11.05868983);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (19, 0.64038694);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (20, 0.69810149);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('AUTO', 0,NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('DETECT_SEASONALITY', 1,NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('OUTLIER_METHOD',3, NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('WINDOW_SIZE',1, NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES('THRESHOLD',NULL,3,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('EXTRAPOLATION',1, NULL,NULL);
CALL "_SYS_AFL"."PAL_OUTLIER_DETECTION_FOR_TIME_SERIES"(PAL_OUTLIER_DATA_TBL,
#PAL_PARAMETER_TBL, ?, ?, ?);
```

### Expected Result

RESULT:

| | TIMESTAMP | RAW_DATA | RESIDUAL | OUTLIER_SCORE | IS_OUTLIER |
|---|---|---|---|---|---|
| 1 | 1 | 0.917022 | -0.5377799990740781 | -0.7560851143232041 | 0 |
| 2 | 2 | 1.22032449 | 0.46934951148148185 | 0.7101520010038659 | 0 |
| 3 | 3 | 10.500114... | -0.033542661296295... | -0.02198737233998... | 0 |
| 4 | 4 | 0.80233257 | -0.2828471268518524 | -0.3849391641236445 | 0 |
| 5 | 5 | 0.64675589 | 0.09445884814814764 | 0.164364593319126... | 0 |
| 6 | 6 | 10.592338... | 0.2578204720370385 | 0.4021958490370364 | 0 |
| 7 | 7 | 0.68626021 | -0.4572383868518517 | -0.6388279989961343 | 0 |
| 8 | 8 | 0.84556073 | 0.1642099014814815 | 0.265912191873830... | 0 |
| 9 | 9 | 10.896767... | -1.0563473946296291 | -1.5110453793070164 | 0 |
| 10 | 10 | 5.03881673 | 2.3184450731481476 | 3.4021717600184163 | 1 |
| 11 | 11 | 0.91919451 | -1.3337037618518521 | -1.9148367443015517 | 0 |
| 12 | 12 | 11.1852195 | 0.5561975253703713 | 0.8365903388955701 | 0 |
| 13 | 13 | 0.70445225 | -0.8202562335185188 | -1.167330276737121 | 0 |
| 14 | 14 | 1.37811744 | 0.6363103214814814 | 0.9532231590439373 | 0 |
| 15 | 15 | 10.527387... | -0.190669811296295... | -0.2507421220925574 | 0 |
| 16 | 16 | 1.17046751 | -0.1366978768518523 | -0.1721666731169942 | 0 |
| 17 | 17 | 0.9173048 | -0.003337271851851... | 0.021987372339989... | 0 |
| 18 | 18 | 11.058689... | 0.49382941870370445 | 0.7457912590861617 | 0 |
| 19 | 19 | 0.64038694 | -0.5941179001851857 | -0.838105073765327 | 0 |
| 20 | 20 | 0.69810149 | 0.16937231703709532 | 0.273427933606199... | 0 |

STATISTICS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | DETECT_SEASONALITY | 1 |
| 2 | Type | additive |
| 3 | Period | 3 |
| 4 | ACF | 0.786964 |
| 5 | OutlierNum | 1 |
| 6 | Median | -0.0184399 |
| 7 | Median Absolute De... | 0.463293 |
| 8 | HandleZero | 0 |

## Isolation Forest Score

Isolation forest is a useful algorithm in outlier detection. We apply this algorithm to the residual, and generate the isolation forest score for each residual point. Because outliers are few and different, the corresponding residual values are also different from others. Those residual points corresponding to outliers are easier to isolate than normal residual points. The isolation forest score is between 0 and 1. Those outlier points tend to

have high scores. It also needs a threshold. Usually we set the threshold to 0.7 (default value for isolation forest score).

### Relevant Parameters

- OUTLIER_METHOD
  To use Isolation Forest score method, you need to set OUTLIER_METHOD = 4.
- THRESHOLD
  To use Isolation Forest score method, we usually set THRESHOLD = 0.7 (default value for Isolation Forest score). A larger value indicates stricter requirement for outlier, that is, fewer outlier points.
- SEED
  It is the seed for random number generator. The default value 0 means using the current time (in second) as seed. We suggest setting SEED to a non-zero value, in case to get the same result every time.
- N_ESTIMATORS
  It specifies the number of trees to grow. The default value is 100.
- MAX_SAMPLES
  It specifies the number of samples to draw from input to train each tree. If MAX_SAMPLES is larger than the number of samples provided, all samples will be used for all trees. The default value is 256.
- BOOTSTRAP
  It specifies sampling method. Its legal values are 0 and 1. The default value 0 means sampling without replacement. The value 1 means sampling with replacement.
- CONTAMINATION
  It is the proportion of outliers in the data set. When CONTAMINATION is specified, THRESHOLD is not valid.
- THREAD_RATIO
  THREAD_RATIO is valid when OUTLIER_METHOD = 4.

### Examples

#### Example 1: Detect outliers with threshold

In this example, we detect outlier with threshold. We set the THRESHOLD to 0.7 (default value for isolation forest score).

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_OUTLIER_DATA_TBL;
CREATE COLUMN TABLE PAL_OUTLIER_DATA_TBL (
    "ID" INT,
    "RAWDATA" DOUBLE
);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (1, 2.0);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (2, 2.5);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (3, 3.2);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (4, 3.8);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (5, 3.4);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (6, 2.9);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (7, 3.1);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (8, 11);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (9, 3.5);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (10, 3.9);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (11, 4.0);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (12, 4.3);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (13, 4.7);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (14, 5.1);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (15, 5.3);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (16, 13.0);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (17, 4.6);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (18, 4.4);
```

```
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (19, 4.1);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (20, 3.7);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('AUTO', 0,NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('DETECT_SEASONALITY', 0,NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('OUTLIER_METHOD',4, NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('WINDOW_SIZE',3, NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES('THRESHOLD',NULL,0.7,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES('SEED',1,NULL,NULL);
CALL "_SYS_AFL"."PAL_OUTLIER_DETECTION_FOR_TIME_SERIES"(PAL_OUTLIER_DATA_TBL,
#PAL_PARAMETER_TBL, ?, ?, ?);
```

**Expected Result**

RESULT:

| | TIMESTAMP | RAW_DATA | RESIDUAL | OUTLIER_SCORE | IS_OUTLIER |
|---|---|---|---|---|---|
| 1 | 1 | 2 | 0 | 0.33858859202157027 | 0 |
| 2 | 2 | 2.5 | 0 | 0.33858859202157027 | 0 |
| 3 | 3 | 3.2 | 0 | 0.33858859202157027 | 0 |
| 4 | 4 | 3.8 | 0.39999... | 0.708218277293074 | 1 |
| 5 | 5 | 3.4 | 0 | 0.33858859202157027 | 0 |
| 6 | 6 | 2.9 | -0.20000... | 0.61064080165046 | 0 |
| 7 | 7 | 3.1 | 0 | 0.33858859202157027 | 0 |
| 8 | 8 | 11 | 7.5 | 0.7575849925737201 | 1 |
| 9 | 9 | 3.5 | -0.39999... | 0.670147589984609 | 0 |
| 10 | 10 | 3.9 | 0 | 0.33858859202157027 | 0 |
| 11 | 11 | 4 | 0 | 0.33858859202157027 | 0 |
| 12 | 12 | 4.3 | 0 | 0.33858859202157027 | 0 |
| 13 | 13 | 4.7 | 0 | 0.33858859202157027 | 0 |
| 14 | 14 | 5.1 | 0 | 0.33858859202157027 | 0 |
| 15 | 15 | 5.3 | 0 | 0.33858859202157027 | 0 |
| 16 | 16 | 13 | 7.7 | 0.7575849925737201 | 1 |
| 17 | 17 | 4.6 | 0 | 0.33858859202157027 | 0 |
| 18 | 18 | 4.4 | 0 | 0.33858859202157027 | 0 |
| 19 | 19 | 4.1 | 0 | 0.33858859202157027 | 0 |
| 20 | 20 | 3.7 | 0 | 0.33858859202157027 | 0 |

STATISTICS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | DETECT_SEASONALITY | 0 |
| 2 | OutlierNum | 3 |
| 3 | HandleZero | 0 |

**Example 2: Detect outliers with contamination**

In this example, we detect outlier with contamination and set the THRESHOLD to 0.1. We use the same time series as example 1 and still keep THRESHOLD = 0.7. We find that the result is different from example 1. THRESHOLD is not valid when CONTAMINATION is specified.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_OUTLIER_DATA_TBL;
CREATE COLUMN TABLE PAL_OUTLIER_DATA_TBL (
    "ID" INT,
    "RAWDATA" DOUBLE
);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (1, 2.0);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (2, 2.5);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (3, 3.2);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (4, 3.8);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (5, 3.4);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (6, 2.9);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (7, 3.1);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (8, 11);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (9, 3.5);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (10, 3.9);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (11, 4.0);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (12, 4.3);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (13, 4.7);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (14, 5.1);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (15, 5.3);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (16, 13.0);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (17, 4.6);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (18, 4.4);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (19, 4.1);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (20, 3.7);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('AUTO', 0,NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('DETECT_SEASONALITY', 0,NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('OUTLIER_METHOD',4, NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('WINDOW_SIZE',3, NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES('THRESHOLD',NULL,0.7,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES('SEED',1,NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES('CONTAMINATION',NULL,0.1,NULL);
CALL "_SYS_AFL"."PAL_OUTLIER_DETECTION_FOR_TIME_SERIES"(PAL_OUTLIER_DATA_TBL,
#PAL_PARAMETER_TBL, ?, ?, ?);
```

**Expected Result**

RESULT:

| | TIMESTAMP | RAW_DATA | RESIDUAL | OUTLIER_SCORE | IS_OUTLIER |
|---|---|---|---|---|---|
| 1 | 1 | 2 | 0 | 0.33858859202157027 | 0 |
| 2 | 2 | 2.5 | 0 | 0.33858859202157027 | 0 |
| 3 | 3 | 3.2 | 0 | 0.33858859202157027 | 0 |
| 4 | 4 | 3.8 | 0.39999... | 0.708218277293074 | 0 |
| 5 | 5 | 3.4 | 0 | 0.33858859202157027 | 0 |
| 6 | 6 | 2.9 | -0.20000... | 0.61064080165046 | 0 |
| 7 | 7 | 3.1 | 0 | 0.33858859202157027 | 0 |
| 8 | 8 | 11 | 7.5 | 0.7575849925737201 | 1 |
| 9 | 9 | 3.5 | -0.39999... | 0.670147589984609 | 0 |
| 10 | 10 | 3.9 | 0 | 0.33858859202157027 | 0 |
| 11 | 11 | 4 | 0 | 0.33858859202157027 | 0 |
| 12 | 12 | 4.3 | 0 | 0.33858859202157027 | 0 |
| 13 | 13 | 4.7 | 0 | 0.33858859202157027 | 0 |
| 14 | 14 | 5.1 | 0 | 0.33858859202157027 | 0 |
| 15 | 15 | 5.3 | 0 | 0.33858859202157027 | 0 |
| 16 | 16 | 13 | 7.7 | 0.7575849925737201 | 1 |
| 17 | 17 | 4.6 | 0 | 0.33858859202157027 | 0 |
| 18 | 18 | 4.4 | 0 | 0.33858859202157027 | 0 |
| 19 | 19 | 4.1 | 0 | 0.33858859202157027 | 0 |
| 20 | 20 | 3.7 | 0 | 0.33858859202157027 | 0 |

STATISTICS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | DETECT_SEASONALITY | 0 |
| 2 | OutlierNum | 2 |
| 3 | HandleZero | 0 |

## DBSCAN

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is a density-based data clustering algorithm. We apply this algorithm to the normalized residual, and divide normalized residual data points into several clusters. Because normal points are the majority and gathered very close, and outlier points are few and away from normal points, we consider the points in the largest cluster as normal points. Other points are considered as outliers. BDSCAN method doesn't need a threshold.

### Relevant Parameters

- OUTLIER_METHOD
  To use DBSCAN method, you need to set OUTLIER_METHOD = 5.
- AUTO_PARAM
  It specifies whether the MINPTS and RADIUS parameters are determined automatically or by user. This parameter is case sensitive. The valid value is `true` or `false`. If `true`, the parameters are automatically determined. If `false`, the algorithm uses parameter values provided by you.
  The default value is `false`.
- MINPTS
  It specifies the minimum number of points required to form a cluster. The point itself is not included in MINPTS. Larger MINPTS tends to predict more outliers. The default value is 1.
- RADIUS
  It specifies the scan radius (eps). Larger MINPTS tends to predict fewer outliers. The default value is 0.5.
- DISTANCE_METHOD
  It specifies the method to compute the distance between two points.
  The default value is 2 (Euclidean distance).
- DBSCAN_NORMALIZATION
  It specifies whether to take normalization of data before using the DBSCAN method. The valid value is 0 or 1. 0 for not normalization and 1 for normalization.
  The default value is 1.
- DBSCAN_OUTLIER_FROM_CLUSTER
  It specifies how to take outliers from the DBSCAN result. The valid value is 0 or 1. If the value is 0, the algorithm takes the largest cluster as normal points and others as outliers. If the value is 1, the algorithm takes the points with CLUSTER_ID = -1 as outliers.
  The default value is 0.
- THREAD_RATIO
  THREAD_RATIO is valid when OUTLIER_METHOD = 5.

### Example

As there is no score in DBSCAN method, the column OUTLIER_SCORE in RESUDLT table is empty.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_OUTLIER_DATA_TBL;
CREATE COLUMN TABLE PAL_OUTLIER_DATA_TBL (
    "ID" INT,
    "RAWDATA" DOUBLE
);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (1, 2.0);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (2, 2.5);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (3, 3.2);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (4, 3.8);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (5, 3.4);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (6, 2.9);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (7, 3.1);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (8, 11);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (9, 3.5);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (10, 3.9);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (11, 4.0);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (12, 4.3);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (13, 4.7);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (14, 5.1);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (15, 5.3);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (16, 13.0);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (17, 4.6);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (18, 4.4);
```

```
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (19, 4.1);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (20, 3.7);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('AUTO', 0,NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('DETECT_SEASONALITY', 0,NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('OUTLIER_METHOD',5, NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('WINDOW_SIZE',3, NULL,NULL);
CALL "_SYS_AFL"."PAL_OUTLIER_DETECTION_FOR_TIME_SERIES"(PAL_OUTLIER_DATA_TBL,
#PAL_PARAMETER_TBL, ?, ?, ?);
```

**Expected Result**

RESULT:

| | TIMESTAMP | RAW_DATA | RESIDUAL | OUTLIER_SCORE | IS_OUTLIER |
|---|---|---|---|---|---|
| 1 | 1 | 2 | 0 | ? | 0 |
| 2 | 2 | 2.5 | 0 | ? | 0 |
| 3 | 3 | 3.2 | 0 | ? | 0 |
| 4 | 4 | 3.8 | 0.39999... | ? | 0 |
| 5 | 5 | 3.4 | 0 | ? | 0 |
| 6 | 6 | 2.9 | -0.20000... | ? | 0 |
| 7 | 7 | 3.1 | 0 | ? | 0 |
| 8 | 8 | 11 | 7.5 | ? | 1 |
| 9 | 9 | 3.5 | -0.39999... | ? | 0 |
| 10 | 10 | 3.9 | 0 | ? | 0 |
| 11 | 11 | 4 | 0 | ? | 0 |
| 12 | 12 | 4.3 | 0 | ? | 0 |
| 13 | 13 | 4.7 | 0 | ? | 0 |
| 14 | 14 | 5.1 | 0 | ? | 0 |
| 15 | 15 | 5.3 | 0 | ? | 0 |
| 16 | 16 | 13 | 7.7 | ? | 1 |
| 17 | 17 | 4.6 | 0 | ? | 0 |
| 18 | 18 | 4.4 | 0 | ? | 0 |
| 19 | 19 | 4.1 | 0 | ? | 0 |
| 20 | 20 | 3.7 | 0 | ? | 0 |

STATISTICS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | DETECT_SEASONALITY | 0 |
| 2 | OutlierNum | 2 |
| 3 | HandleZero | 0 |

## Voting

You can select some of the previous methods and detect the outliers by voting the selected methods.

### Relevant Parameters

- `VOTING_OUTLIER_METHOD`
  It specifies the outlier methods for voting. The valid values and their corresponding meaning are the same as OUTLIER_METHOD. This parameter can have multiple values.
  No default value for this parameter.
- `THRESHOLD_OUTLIER_METHOD_{num}`
  It specifies the threshold for each outlier method in voting.
  The default value is the same as the THRESHOLD parameter for each outlier method.
- `CONTAMINATION_OUTLIER_METHOD_{num}`
  It specifies the proportion of outliers in the data set for each outlier method when it is valid. Currently, it is only valid when num = 4 (Isolation Forest). CONTAMINATION_OUTLIER_METHOD_{num} has a higher priority than THRESHOLD_OUTLIER_METHOD_{num}.
- `VOTING_OUTLIER_METHOD_CRITERION`
  It specifies the criterion for outlier voting. Suppose the number of voters is N. If more than int(criterion * N) voters detect the point as an outlier, the point will be treated as an outlier.
  The default value is 0.5.
- Other parameters for outlier methods
  Except THRESHOLD and CONTAMINATION, other parameters for outlier methods are also valid in the voting logic.

### Example

In the voting logic, the column OUTLIER_SCORE in the RESULT table is empty.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_OUTLIER_DATA_TBL;
CREATE COLUMN TABLE PAL_OUTLIER_DATA_TBL (
    "ID" INT,
    "RAWDATA" DOUBLE
);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (1, 2.0);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (2, 2.5);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (3, 3.2);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (4, 3.8);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (5, 3.4);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (6, 2.9);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (7, 3.1);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (8, -5);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (9, 3.5);
```

```
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (10, 3.9);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (11, 4.0);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (12, 4.3);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (13, 4.7);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (14, 5.1);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (15, 5.3);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (16, 14.0);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (17, 14.6);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (18, 4.4);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (19, 4.1);
INSERT INTO PAL_OUTLIER_DATA_TBL VALUES (20, 3.7);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('AUTO', 1,NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('VOTING_OUTLIER_METHOD', 0,NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('VOTING_OUTLIER_METHOD', 1,NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('VOTING_OUTLIER_METHOD', 2,NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('VOTING_OUTLIER_METHOD', 3,NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('VOTING_OUTLIER_METHOD', 4,NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('VOTING_OUTLIER_METHOD', 5,NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('THRESHOLD_OUTLIER_METHOD_0',
NULL,2.0,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('THRESHOLD_OUTLIER_METHOD_1',
NULL,2.0,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('VOTING_OUTLIER_METHOD_CRITERION',
NULL,0.5,NULL);
CALL "_SYS_AFL"."PAL_OUTLIER_DETECTION_FOR_TIME_SERIES"(PAL_OUTLIER_DATA_TBL,
#PAL_PARAMETER_TBL, ?, ?, ?);
```

### Expected Result

RESULT:

| | TIMESTAMP | RAW_DATA | RESIDUAL | OUTLIER_SCORE | IS_OUTLIER |
|---|---|---|---|---|---|
| 1 | 1 | 2 | 0 | ? | 0 |
| 2 | 2 | 2.5 | 0 | ? | 0 |
| 3 | 3 | 3.2 | 0 | ? | 0 |
| 4 | 4 | 3.8 | 0.39999... | ? | 0 |
| 5 | 5 | 3.4 | 0 | ? | 0 |
| 6 | 6 | 2.9 | -0.20000... | ? | 0 |
| 7 | 7 | 3.1 | 0.20000... | ? | 0 |
| 8 | 8 | -5 | -8.1 | ? | 1 |
| 9 | 9 | 3.5 | 0 | ? | 0 |
| 10 | 10 | 3.9 | 0 | ? | 0 |
| 11 | 11 | 4 | 0 | ? | 0 |
| 12 | 12 | 4.3 | 0 | ? | 0 |
| 13 | 13 | 4.7 | 0 | ? | 0 |
| 14 | 14 | 5.1 | 0 | ? | 0 |
| 15 | 15 | 5.3 | 0 | ? | 0 |
| 16 | 16 | 14 | 0 | ? | 1 |
| 17 | 17 | 14.6 | 0.59999... | ? | 1 |
| 18 | 18 | 4.4 | 0 | ? | 0 |
| 19 | 19 | 4.1 | 0 | ? | 0 |
| 20 | 20 | 3.7 | 0 | ? | 0 |

STATISTICS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | Auto Outlier Method | median filter |
| 2 | DETECT_INTERMITTENT_TS | 0 |
| 3 | INTERMITTENT_FLAG | 0 |
| 4 | REMOVE_LEADING_AND_TRAILING_ZEROS | 0 |
| 5 | OutlierNum | 3 |
| 6 | HandleZero | 1 |
| 7 | ID of Z1 Outliers | 8, 16, 17 |
| 8 | ID of Z1 Supplemented Outliers | 16, 17 |
| 9 | ID of Z2 Outliers | 8, 16, 17 |
| 10 | ID of Z2 Supplemented Outliers | 16, 17 |
| 11 | ID of IQR Outliers | 4, 6, 7, 8, 15, 16, 17, 18 |
| 12 | ID of IQR Supplemented Outliers | 15, 16, 17, 18 |
| 13 | ID of MAD Outliers | 4, 6, 7, 8, 15, 16, 17, 18, 19 |
| 14 | ID of MAD Supplemented Outliers | 15, 16, 17, 18, 19 |
| 15 | ID of IsolationForest Outliers | 8 |
| 16 | ID of IsolationForest Supplemented Outliers | |
| 17 | ID of DBSCAN Outliers | 8, 16, 17 |
| 18 | ID of DBSCAN Supplemented Outliers | 16, 17 |

## 5.5.25 Periodogram

This function performs power spectral density estimate of the input signal using a periodogram.

The periodogram is a nonparametric estimate of the power spectral density (PSD) of a wide-sense stationary random process. For a signal Xn sampled at fs samples per unit time, the two-sided periodogram is defined as

$$\hat{P}(f) = \frac{1}{N \cdot f_s} \left| \sum_{n=0}^{N-1} x_n e^{-j2\Pi f_n / f_s} \right|^2, -f_s/2 < f \leq f_s/2$$

For a one-sided periodogram, the values at all frequencies except 0 and the Nyquist, fs/2, are multiplied by 2 so that the total power is conserved.

The modified periodogram is also provided where the input time series is multiplied by a window function. A suitable window function is non-negative and decays to zero at the beginning and end points. Multiplying the time series by the window function smoothes discontinuities at the edges of the sampled signal and helps to alleviate the leakage in the periodogram.

The periodogram gives a measure of the relative importance of possible periods (or frequencies). A relatively large value of Pxx(f) indicates relatively more importance for the frequency f (or near f) in explaining the oscillation pattern of the observed series. This is useful for identifying the dominant cyclical behavior in the series, particularly when the cycles are not related to the commonly encountered monthly or quarterly seasonality.

Prerequisites

- An ID column must be provided.
- The input data does not contain null value.

### _SYS_AFL.PAL_PERIODOGRAM

This function estimates periodogram power spectral density on input series.

**Overview of All Tables**

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<RESULT table>` | OUT |

**Input Table**

| Table | Column | Reference for Output Table | Data Type | Description |
|---|---|---|---|---|
| DATA | 1st Column | ID | INTEGER | The ID column that indicates the order of sequence. This must be the first column. |
| | 2nd Column | | INTEGER, DOUBLE, or DECIMAL (p, s) | Sampled sequence |

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| SAMPLING_RATE | DOUBLE | 1.0 | Sampling frequency of the sequence | |
| NUM_FFT | INTEGER | The length of sequence | Number of DFT points. If NUM_FFT is smaller than the length of the input, the input is cropped. If it is larger, the input is padded with zeros. | |
| FREQ_RANGE | INTEGER | 0 | Indicates result frequency range:<br>• 0: one side<br>• 1: two sides | |
| SPECTRUM_TYPE | INTEGER | 0 | Indicates power spectrum scaling type:<br>• 0: density<br>• 1: spectrum | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| WINDOW | VARCHAR, NVARCHAR | none | Available window types:<br>• none<br>• bartlett<br>• bartlett_hann<br>• blackman<br>• blackman_harris<br>• bohman<br>• chebwin<br>• cosine<br>• flattop<br>• gaussian<br>• hamming<br>• hann<br>• kaiser<br>• nuttall<br>• parzen<br>• tukey | |
| ALPHA | DOUBLE | • blackman: 0.16<br>• gaussian: 2.5 | Window parameter | Only valid for blackman and gaussian window. |
| BETA | DOUBLE | 8.6 | Kaiser window parameter | Only valid for kaiser window. |
| ATTENUATION | DOUBLE | 50.0 | Chewin window parameter | Only valid for chewin window. |
| MODE | VARCHAR, NVARCHAR | symmetric | Flattop window parameter<br>• symmetric<br>• periodic | Only valid for flattop window. |
| PRECISION | VARCHAR, NVARCHAR | none | Flattop window parameter<br>• none<br>• octave | Only valid for flattop window. |
| R | DOUBLE | 0.5 | Tukey window parameter | Only valid for tukey window. |

### Output Table

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| RESULT | 1st Column | ID (in input table) data type | ID (in input table) column name | ID column |
| | 2nd Column | DOUBLE | FREQ | Value of sample frequencies |

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| | 3rd Column | DOUBLE | PXX | Power spectral density or power spectrum of input data |

### Example

Assume that:

- DM_PAL is a schema belonging to USER1;
- USER1 has been assigned the AFL__SYS_AFL_AFLPAL_EXECUTE or AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION role.

### Example

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_PERIODOGRAM_DATA_TBL;
CREATE COLUMN TABLE PAL_PERIODOGRAM_DATA_TBL (
    "ID" INTEGER,
    "X" DOUBLE);
INSERT INTO PAL_PERIODOGRAM_DATA_TBL VALUES (1, -2.0);
INSERT INTO PAL_PERIODOGRAM_DATA_TBL VALUES (2, 8.0);
INSERT INTO PAL_PERIODOGRAM_DATA_TBL VALUES (3, 6.0);
INSERT INTO PAL_PERIODOGRAM_DATA_TBL VALUES (4, 4.0);
INSERT INTO PAL_PERIODOGRAM_DATA_TBL VALUES (5, 1.0);
INSERT INTO PAL_PERIODOGRAM_DATA_TBL VALUES (6, 0.0);
INSERT INTO PAL_PERIODOGRAM_DATA_TBL VALUES (7, 3.0);
INSERT INTO PAL_PERIODOGRAM_DATA_TBL VALUES (8, 5.0);
DROP TABLE PAL_PERIODOGRAM_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PERIODOGRAM_PARAMETER_TBL (
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR (1000)
);
INSERT INTO PAL_PERIODOGRAM_PARAMETER_TBL VALUES ('SAMPLING_RATE', NULL, 100,
NULL);
INSERT INTO PAL_PERIODOGRAM_PARAMETER_TBL VALUES ('FREQ_RANGE', 1, NULL, NULL);
INSERT INTO PAL_PERIODOGRAM_PARAMETER_TBL VALUES ('WINDOW', NULL, NULL,
'hamming');
DROP TABLE PAL_PERIODOGRAM_RESULT_TBL;
CREATE COLUMN TABLE PAL_PERIODOGRAM_RESULT_TBL (
    "ID" INTEGER,
    "FREQ" DOUBLE,
    "PXX" DOUBLE
);
DO BEGIN
  lt_data = SELECT * FROM PAL_PERIODOGRAM_DATA_TBL;
  lt_param = SELECT * FROM PAL_PERIODOGRAM_PARAMETER_TBL;
  CALL _SYS_AFL.PAL_PERIODOGRAM (:lt_data, :lt_param, lt_result);
  INSERT INTO PAL_PERIODOGRAM_RESULT_TBL SELECT * FROM :lt_result;
END;
SELECT * FROM PAL_PERIODOGRAM_RESULT_TBL;
```

### Expected Result

| | ID | FREQ | PXX |
|---|---|---|---|
| 1 | 1 | 0 | 0.44937128546306543 |
| 2 | 2 | 12.5 | 0.07273682191145418 |
| 3 | 3 | 25 | 0.07579024453401592 |
| 4 | 4 | 37.5 | 0.006659143304802... |
| 5 | 5 | -50 | 0.000960437733323... |
| 6 | 6 | -37.5 | 0.006659143304802... |
| 7 | 7 | -25 | 0.07579024453401592 |
| 8 | 8 | -12.5 | 0.07273682191145418 |

## 5.5.26 ROCKET

RandOm Convolutional KErnel Transform (ROCKET) is an exceptionally efficient algorithm for time series classification. Unlike other proposed time series classification algorithms which attain excellent accuracy, ROCKET maintains its performance with a fraction of the computational expense by transforming input time series using random convolutional kernels.

Afterwards, a simple classifier is enough to train the transformed features and render the state-of-the-art accuracy. Specifically, in our implementation, the multi-class logistic regression is employed.

Instead of the original ROKCET algorithm, we provide two variants of it to transform the input time series, that is, MiniRocket and MultiRocket. MiniRokcet is even faster than ROCKET while maintaining essentially the same accuracy. MultiRocket takes the first order difference of the input time series and improves the diversity of generated features to boost the accuracy with the extra cost of time and space.

Particularly, this function supports both univariate and multivariate time series.

### Prerequisites

- No null data in the inputs.
- The length of time series must be at least 9 for MiniRocket and 10 for MultiRocket.

### _SYS_AFL.PAL_ROCKET_TRAIN

This function trains the time series upon their labels.

**Overview of All Tables**

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |

| Position | Table Name | Parameter Type |
|---|---|---|
| 2 | `<LABEL table>` | IN |
| 3 | `<PARAMETER table>` | IN |
| 4 | `<MODEL table>` | OUT |
| 5 | `<STATS table>` | OUT |

### Input Table

| Table | Column | Data Type | Description |
|---|---|---|---|
| DATA | 1st Column | INTEGER | Record ID |
| | 2nd Column | INTEGER, DOUBLE | Time series value |

For univariate time series, each row represents one time series. However, for multivariate time series, a fixed number of consecutive rows form one time series, and that number is designated by the parameter DATA_DIM.

| Table | Column | Data Type | Description |
|---|---|---|---|
| LABEL | 1st Column | INTEGER | Data ID |
| | 2nd Column | INTEGER, NVARCHAR | Labels |

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| METHOD | INTEGER | 0 | • 0: MiniRocket<br>• 1: MultiRocket | |
| NUM_FEATURES | INTEGER | • MiniRocket: 9996<br>• Multi-Rocket:49728 | Number of trans-formed features for each time series | |
| DATA_DIM | INTEGER | 1 | • 1: for univariate time series<br>• Other values: for multivariate | Cannot be smaller than 1. |
| RANDOM_SEED | INTEGER | 0 | 0 indicates using the machine time as the seed. | |
| MAX_ITER | INTEGER | 100 | Maximum number of iterations for the MCLR classifier. | |

### Output Table

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| MODEL | 1st Column | INTEGER | ROW_INDEX | Indicates the ID of the current row. |
| | 2nd Column | NVARCHAR | MODEL_CONTENT | First row represents the method name and others are JSON strings. |
| STATS | 1st Column | NVARCHAR | STAT_NAME | Name of training stats |
| | 2nd Column | NVARCHAR | STAT_VALUE | Corresponding stat values |

### Examples

Assume that:

- `DM_PAL` is a schema belonging to `USER1`;
- `USER1` has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

**Example 1: Univariate time series trained by MiniRocket**

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_ROCKET_DATA_TBL;
CREATE COLUMN TABLE PAL_ROCKET_DATA_TBL (
    "RECORD_ID" INTEGER,
    "VAL_01"     DOUBLE,
    "VAL_02"     DOUBLE,
    "VAL_03"     DOUBLE,
    "VAL_04"     DOUBLE,
    "VAL_05"     DOUBLE,
    "VAL_06"     DOUBLE,
    "VAL_07"     DOUBLE,
    "VAL_08"     DOUBLE,
    "VAL_09"     DOUBLE,
    "VAL_10"     DOUBLE,
    "VAL_11"     DOUBLE,
    "VAL_12"     DOUBLE,
    "VAL_13"     DOUBLE,
    "VAL_14"     DOUBLE,
    "VAL_15"     DOUBLE,
    "VAL_16"     DOUBLE
);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(0,1.598,1.599,1.571,1.550,1.507,1.434,1.369,1.305,1.210,1.117,1.024,0.926,0.828,
0.739,0.643,0.556);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(1,1.701,1.671,1.619,1.547,1.475,1.391,1.306,1.237,1.153,1.070,0.985,0.899,0.816,
0.733,0.658,0.581);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(2,1.722,1.695,1.657,1.606,1.512,1.414,1.314,1.213,1.113,1.015,0.920,0.828,0.740,
0.658,0.586,0.501);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(3,1.726,1.660,1.573,1.496,1.409,1.332,1.246,1.159,1.073,0.987,0.901,0.815,0.730,
0.644,0.558,0.484);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(4,1.779,1.761,1.703,1.611,1.492,1.369,1.245,1.121,1.011,0.900,0.786,0.679,0.580,
0.502,0.415,0.333);
```

```
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(5,1.800,1.743,1.686,1.633,1.532,1.423,1.311,1.201,1.090,0.979,0.872,0.767,0.664,
0.561,0.453,0.355);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(6,1.749,1.727,1.659,1.560,1.457,1.355,1.252,1.162,1.062,0.961,0.864,0.771,0.682,
0.595,0.513,0.427);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(7,1.348,1.237,1.129,1.022,0.939,0.847,0.750,0.659,0.566,0.474,0.388,0.306,0.218,
0.133,0.061,0.009);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(8,1.696,1.634,1.596,1.507,1.414,1.323,1.242,1.186,1.122,1.048,0.966,0.890,0.805,
0.719,0.632,0.553);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(9,1.723,1.713,1.665,1.587,1.495,1.404,1.312,1.220,1.130,1.041,0.955,0.870,0.787,
0.706,0.622,0.547);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(10,1.614,1.574,1.557,1.521,1.460,1.406,1.316,1.227,1.137,1.045,0.957,0.862,0.771
,0.681,0.587,0.497);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(11,1.652,1.665,1.656,1.623,1.571,1.499,1.412,1.339,1.253,1.155,1.058,0.973,0.877
,0.797,0.704,0.609);
DROP TABLE PAL_ROCKET_LABEL_TBL;
CREATE COLUMN TABLE PAL_ROCKET_LABEL_TBL (
    "DATA_ID"  INTEGER,
    "LABEL"    NVARCHAR(5000)
);
INSERT INTO PAL_ROCKET_LABEL_TBL VALUES (0,  'A');
INSERT INTO PAL_ROCKET_LABEL_TBL VALUES (1,  'B');
INSERT INTO PAL_ROCKET_LABEL_TBL VALUES (2,  'C');
INSERT INTO PAL_ROCKET_LABEL_TBL VALUES (3,  'A');
INSERT INTO PAL_ROCKET_LABEL_TBL VALUES (4,  'B');
INSERT INTO PAL_ROCKET_LABEL_TBL VALUES (5,  'C');
INSERT INTO PAL_ROCKET_LABEL_TBL VALUES (6,  'A');
INSERT INTO PAL_ROCKET_LABEL_TBL VALUES (7,  'B');
INSERT INTO PAL_ROCKET_LABEL_TBL VALUES (8,  'C');
INSERT INTO PAL_ROCKET_LABEL_TBL VALUES (9,  'B');
INSERT INTO PAL_ROCKET_LABEL_TBL VALUES (10, 'C');
INSERT INTO PAL_ROCKET_LABEL_TBL VALUES (11, 'A');
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL (
    PARAM_NAME VARCHAR(100),
    INT_VALUE INTEGER,
    DOUBLE_VALUE DOUBLE,
    STRING_VALUE VARCHAR(100)
);
INSERT INTO PAL_PARAMETER_TBL VALUES ('METHOD', 0, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('RANDOM_SEED', 1, NULL, NULL);
DROP TABLE PAL_ROCKET_MODEL_TBL; -- for predict followed
CREATE COLUMN TABLE PAL_ROCKET_MODEL_TBL (
    ROW_INDEX     INTEGER,
    MODEL_CONTENT NVARCHAR(5000)
);
DROP TABLE PAL_ROCKET_STAT_TBL;
CREATE COLUMN TABLE PAL_ROCKET_STAT_TBL (
    STAT_NAME  NVARCHAR(5000),
    STAT_VALUE NVARCHAR(5000)
);
DO BEGIN
    lt_data  = SELECT * FROM PAL_ROCKET_DATA_TBL;
    lt_label = SELECT * FROM PAL_ROCKET_LABEL_TBL;
    lt_param = SELECT * FROM PAL_PARAMETER_TBL;
    CALL _SYS_AFL.PAL_ROCKET_TRAIN (:lt_data, :lt_label, :lt_param, lt_model,
lt_stat);
    INSERT INTO PAL_ROCKET_MODEL_TBL SELECT * FROM :lt_model;
    INSERT INTO PAL_ROCKET_STAT_TBL  SELECT * FROM :lt_stat;
END;
SELECT * FROM PAL_ROCKET_MODEL_TBL;
SELECT * FROM PAL_ROCKET_STAT_TBL;
```

MODEL table:

| | ROW_INDEX | MODEL_CONTENT |
|---|---|---|
| 1 | -1 | MiniRocket |
| 2 | 0 | {"SERIES_LENGTH":16,"NUM_CHANNELS":1,"BIAS_SIZE":9996,"DILATIONS":[{"DILATION_SIZ |
| 3 | 1 | 5121315,1.685720499622002,2.8191069172360176,1.8498903116950654,-2.67195875409352 |
| 4 | 2 | 10192183,1.4931236298379538,-4.462113103585729,1.6273520864075167,1.2275774717150 |
| 5 | 3 | 4860881,-6.2434692203217339,0.6595998500205768,-0.9305545047450137,0.818679318210 |
| 6 | 4 | 69559065,1.9060829312562862,1.545352086407516,-3.891791599197902,1.60358381798334 |
| 7 | 5 | 74254174964419,0.7807669820501145,-0.9404343044089281,0.8390216814708396,0.739839 |
| 8 | 6 | 81533021405,1.4075106120727759,1.2378664004701428,-0.5964396623017748,1.340824166 |
| 9 | 7 | .83201472074542,0.5910975533534777,-2.907152491462228,0.8018479869696528,0.540090 |
| 10 | 8 | 311754219297887,0.35028137687727026,0.26115163808221117,-2.457315595076049,0.3212 |
| 11 | 9 | .3706551767099899,0.6838884428175558,0.5483745183608766,-2.2840642681305116,0.648 |
| 12 | 10 | 085586,-0.17951904283598225,0.6509367151145861,-0.030211107816317616,-0.7079800052 |
| 13 | 11 | 176979,-0.8446306428177884,-0.21020776670309317,-0.42104800992405808,-2.6271982615 |
| 14 | 12 | 9486,-0.6999096078918885,-2.165277648064085,-0.5491074049031134,-0.804202154647029 |
| 15 | 13 | 8,0.5726893606046842,1.6531783686003617,1.4366720611027709,-2.4937537939283676,1.6 |
| 16 | 14 | 921852,1.2683646272351895,2.73706420920466,1.3037572643270477,0.9823329555522138, |
| 17 | 15 | 535774557819811,0.871204507966845,0.5841400709763462,-1.0110881667513896,0.655198 |
| 18 | 16 | 86642209044933,1.2494887680461456,1.630779192343078,1.304772619024411,-0.58177762 |
| 19 | 17 | 83325529,-0.054321200823858269,0.4034681264041353,0.3135593117957363,-3.453942261 |
| 20 | 18 | 41664007713,1.6205068045466014,0.5622363823690758,0.3956126022157074,1.017838106 |

STATS table:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | MINIROCKET_TRANSFORM_TIME | 0.043s |
| 2 | TRAINING_TIME | 0.039s |
| 3 | TRAINING_ACCURACY | 1 |
| 4 | TRAINING_OBJ | 6.45594e-14 |
| 5 | TRAINING_ITER | 56 |

### Example 2: Multivariate time series (with dimensionality 8) trained by MultiRocket

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_ROCKET_DATA_TBL;
CREATE COLUMN TABLE PAL_ROCKET_DATA_TBL (
    "RECORD_ID" INTEGER,
    "VAL_01"     DOUBLE,
    "VAL_02"     DOUBLE,
    "VAL_03"     DOUBLE,
    "VAL_04"     DOUBLE,
    "VAL_05"     DOUBLE,
    "VAL_06"     DOUBLE,
    "VAL_07"     DOUBLE,
    "VAL_08"     DOUBLE,
    "VAL_09"     DOUBLE,
    "VAL_10"     DOUBLE,
    "VAL_11"     DOUBLE,
    "VAL_12"     DOUBLE,
    "VAL_13"     DOUBLE,
    "VAL_14"     DOUBLE,
    "VAL_15"     DOUBLE,
    "VAL_16"     DOUBLE
);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(0,1.645,1.646,1.621,1.585,1.540,1.470,1.408,1.318,1.225,1.161,1.070,0.980,0.893,
0.798,0.705,0.620);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(1,1.704,1.705,1.706,1.680,1.632,1.560,1.467,1.376,1.282,1.186,1.090,0.994,0.895,
0.799,0.702,0.605);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(2,1.699,1.666,1.621,1.538,1.454,1.357,1.270,1.173,1.076,0.979,0.885,0.793,0.706,
0.623,0.541,0.460);
```

```
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(3,1.709,1.663,1.580,1.497,1.413,1.330,1.246,1.163,1.080,0.997,0.913,0.831,0.748,
0.665,0.582,0.509);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(4,1.687,1.688,1.674,1.619,1.531,1.439,1.345,1.253,1.160,1.069,0.977,0.900,0.810,
0.722,0.644,0.557);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(5,1.665,1.623,1.584,1.504,1.420,1.336,1.251,1.167,1.083,0.998,0.913,0.828,0.742,
0.659,0.586,0.507);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(6,1.715,1.680,1.601,1.512,1.422,1.332,1.243,1.162,1.073,0.986,0.899,0.813,0.729,
0.644,0.559,0.475);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(7,1.594,1.586,1.568,1.531,1.464,1.384,1.301,1.221,1.139,1.053,0.968,0.882,0.798,
0.714,0.630,0.547);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(8,1.754,1.757,1.736,1.693,1.618,1.527,1.412,1.322,1.202,1.083,0.963,0.874,0.764,
0.663,0.567,0.495);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(9,1.484,1.486,1.513,1.509,1.469,1.455,1.398,1.314,1.232,1.144,1.057,0.963,0.862,
0.767,0.665,0.570);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(10,1.610,1.611,1.577,1.534,1.493,1.436,1.359,1.271,1.193,1.096,1.000,0.903,0.805
,0.712,0.619,0.527);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(11,1.579,1.614,1.623,1.610,1.581,1.535,1.460,1.381,1.311,1.226,1.142,1.058,0.976
,0.892,0.817,0.737);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(12,1.636,1.547,1.458,1.369,1.280,1.199,1.111,1.023,0.935,0.848,0.768,0.681,0.595
,0.509,0.423,0.337);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(13,1.150,1.243,1.328,1.415,1.501,1.588,1.676,1.741,1.728,1.655,1.568,1.490,1.403
,1.317,1.231,1.152);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(14,1.324,1.340,1.190,1.140,1.109,0.995,1.078,0.960,0.853,0.763,0.660,0.535,0.486
,0.457,0.446,0.451);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(15,1.659,1.650,1.603,1.516,1.414,1.322,1.221,1.121,1.028,0.960,0.916,0.884,0.830
,0.776,0.699,0.627);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(16,1.793,1.765,1.717,1.624,1.517,1.409,1.302,1.197,1.091,0.989,0.888,0.802,0.702
,0.608,0.513,0.420);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(17,1.637,1.614,1.575,1.506,1.424,1.342,1.268,1.182,1.097,1.010,0.923,0.838,0.751
,0.665,0.592,0.508);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(18,1.707,1.677,1.632,1.570,1.478,1.384,1.288,1.192,1.115,1.022,0.928,0.833,0.741
,0.653,0.560,0.472);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(19,1.821,1.794,1.798,1.775,1.720,1.669,1.557,1.443,1.332,1.204,1.081,0.953,0.836
,0.717,0.609,0.508);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(20,1.609,1.610,1.597,1.586,1.539,1.498,1.424,1.339,1.242,1.140,1.035,0.932,0.826
,0.722,0.617,0.515);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(21,1.409,1.507,1.597,1.689,1.747,1.758,1.723,1.637,1.549,1.458,1.369,1.280,1.189
,1.099,1.010,0.920);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(22,1.676,1.650,1.582,1.495,1.400,1.317,1.231,1.133,1.039,0.943,0.848,0.756,0.665
,0.572,0.481,0.396);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(23,1.575,1.551,1.530,1.504,1.458,1.392,1.313,1.216,1.121,1.022,0.923,0.825,0.728
,0.626,0.530,0.440);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(24,1.722,1.723,1.696,1.651,1.566,1.471,1.377,1.282,1.187,1.091,0.993,0.896,0.798
,0.700,0.604,0.510);
```

```
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(25,1.627,1.609,1.592,1.572,1.532,1.451,1.361,1.271,1.175,1.078,0.982,0.882,0.786
,0.686,0.590,0.494);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(26,1.684,1.656,1.601,1.517,1.432,1.346,1.261,1.176,1.106,1.022,0.939,0.856,0.772
,0.688,0.604,0.522);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(27,1.697,1.665,1.590,1.508,1.424,1.341,1.258,1.175,1.092,1.009,0.926,0.844,0.760
,0.678,0.595,0.513);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(28,1.406,1.320,1.234,1.148,1.063,0.978,0.893,0.809,0.725,0.642,0.558,0.477,0.396
,0.314,0.234,0.153);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(29,1.592,1.593,1.571,1.551,1.527,1.475,1.410,1.346,1.252,1.160,1.058,0.956,0.859
,0.763,0.668,0.574);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(30,1.688,1.648,1.570,1.490,1.408,1.327,1.246,1.164,1.092,1.011,0.930,0.849,0.768
,0.687,0.606,0.524);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(31,1.708,1.663,1.595,1.504,1.411,1.318,1.225,1.134,1.042,0.951,0.861,0.794,0.704
,0.614,0.529,0.446);
DROP TABLE PAL_ROCKET_LABEL_TBL;
CREATE COLUMN TABLE PAL_ROCKET_LABEL_TBL (
    "DATA_ID"  INTEGER,
    "LABEL"    NVARCHAR(5000)
);
INSERT INTO PAL_ROCKET_LABEL_TBL VALUES (0,  'A');
INSERT INTO PAL_ROCKET_LABEL_TBL VALUES (1,  'B');
INSERT INTO PAL_ROCKET_LABEL_TBL VALUES (2,  'C');
INSERT INTO PAL_ROCKET_LABEL_TBL VALUES (3,  'A');
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL (
    PARAM_NAME VARCHAR(100),
    INT_VALUE INTEGER,
    DOUBLE_VALUE DOUBLE,
    STRING_VALUE VARCHAR(100)
);
-- PAL_ROCKET_FIT part
INSERT INTO PAL_PARAMETER_TBL VALUES ('METHOD',      1, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('DATA_DIM',    8, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('RANDOM_SEED', 1, NULL, NULL);
DROP TABLE PAL_ROCKET_MODEL_TBL; -- for predict followed
CREATE COLUMN TABLE PAL_ROCKET_MODEL_TBL (
    ROW_INDEX     INTEGER,
    MODEL_CONTENT NVARCHAR(5000)
);
DROP TABLE PAL_ROCKET_STAT_TBL;
CREATE COLUMN TABLE PAL_ROCKET_STAT_TBL (
    STAT_NAME  NVARCHAR(5000),
    STAT_VALUE NVARCHAR(5000)
);
DO BEGIN
    lt_data  = SELECT * FROM PAL_ROCKET_DATA_TBL;
    lt_label = SELECT * FROM PAL_ROCKET_LABEL_TBL;
    lt_param = SELECT * FROM PAL_PARAMETER_TBL;
    CALL _SYS_AFL.PAL_ROCKET_TRAIN (:lt_data, :lt_label, :lt_param, lt_model,
lt_stat);
    INSERT INTO PAL_ROCKET_MODEL_TBL SELECT * FROM :lt_model;
    INSERT INTO PAL_ROCKET_STAT_TBL  SELECT * FROM :lt_stat;
END;
SELECT * FROM PAL_ROCKET_MODEL_TBL;
SELECT * FROM PAL_ROCKET_STAT_TBL;
```

MODEL table:

| | ROW_INDEX | |
|---|---|---|
| 1 | -1 | MultiRocket |
| 2 | 0 | {"SERIES_LENGTH":16,"NUM_CHANNELS":8,"BIAS_SIZE":6216,"BIAS_DIFF_SIZE":6216,"DILATIONS":[{"DILATION_SIZE":1,"NUM_FEATURES":74}],"D |
| 3 | 1 | HANNELS":[6]},{"ID":77,"CHANNELS":[1,4,7,6,5]},{"ID":78,"CHANNELS":[7,0,5,6]},{"ID":79,"CHANNELS":[1,4]},{"ID":80,"CHANNELS":[1,4,6,2]},{"ID":81,"C |
| 4 | 2 | 504,-22.340878522815215,7.959895076819708,5.81470488591989,11.8578032022155,7.627485955788726,-13.98688419511595,8.1233062650490 |
| 5 | 3 | 48,9.013306944001223,-18.05915327183857,9.197905923784694,8.537362238736196,13.965540234001758,9.113025371001515,-5.4316397891810 |
| 6 | 4 | 8578,-5.9781939194202799,2.23437952460736,1.3156543834261514,3.434864760842622,1.9573681596337263,-3.037385353745023,2.369223250 |
| 7 | 5 | 07,9.199049371639408,7.1111111151007428,16.259771061331127,8.668296377734663,2.860724791448105,10.50366180880851,7.9032098073707 |
| 8 | 6 | .1919862979765044,0.27121051666530307,3.4304422071978797,0.9158462448828436,-1.38406311309445,1.345609162454411,0.7418658591550 |
| 9 | 7 | 530387977505125,-2.0780372353665874,-0.04089799798518077,-0.18266004693153657,0.5731650334644877,-0.12163461524651172,-0.7409659 |
| 10 | 8 | 81529759048407,-0.4224908796580212,0.3050867965370993,-0.36966210167075566,-0.9311206723207743,-0.3142002624633562,-0.3891387076 |
| 11 | 9 | 0869926636,8.950068054111778,21.702397663867726,12.027393495455799,-1.3422272814195898,17.231804326322768,10.819741185385425,-53 |
| 12 | 10 | ,2.6450713593122048,10.231613347274037,4.319696925529633,-1.0621511766606747,5.884599388593907,3.0121746460888225,-21.6541002870 |
| 13 | 11 | 621435368,2.409440699051263,2.213212553521503,6.010489226247179,2.35451730675841,2.0081136830140466,3.8239394250623,2.319601952 |
| 14 | 12 | 8979174,0.00015035332044800227,-2.813201738058908,0.0016913729829520068,2.390383211499055,8.073118777056227,2.758017414860195,1. |
| 15 | 13 | 1585639064478836,-0.05497382428246635,-0.445856425354717,2.747894240735274,-0.42178556352960719,-0.6521995643147284,0.1553947728 |
| 16 | 14 | .999937238989194,-2.5241789509560519,-0.978445003039326,-1.0457997212660993,2.4052241196023949,-0.9925299868357713,-1.77511646844 |
| 17 | 15 | 2060744629,14.773187672699806,6.18556909943719,-21.440110881626237,8.012145211769744,5.506329908183801,16.51463919949335,6.74798 |
| 18 | 16 | 29545709900083907,0.011287594156603989,-0.34922107956267497,0.10739853849382485,-0.006330951674932782,1.8079908408810182,-0.7270 |
| 19 | 17 | 5270445,0.05339909989038331,-0.26186436361349627,1.7113432481556519,-0.250402174376644,-0.27085955026450805,0.13376350872295884, |
| 20 | 18 | 1416770384739,0.03841058726298097,-0.35992059571897619,-1.3170372381551772,-0.08256168079938416,-0.5139750202168117,1.4643453760 |

STATS table:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | MULTIROCKET_TRANSFORM_TIME | 0.035s |
| 2 | TRAINING_TIME | 0.185s |
| 3 | TRAINING_ACCURACY | 1 |
| 4 | TRAINING_OBJ | 2.08e-13 |
| 5 | TRAINING_ITER | 48 |

# _SYS_AFL.PAL_ROCKET_PREDICT

This function predicts the classes of input time series based on the trained model.

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<MODEL table>` | IN |
| 3 | `<PARAMETER table>` | IN |
| 4 | `<RESULT table>` | OUT |

### Input Table

| Table | Column | Data Type | Column Name |
|---|---|---|---|
| DATA | 1st Column | INTEGER | Record ID |
| | 2nd Column | INTEGER, DOUBLE | Time series value |
| MODEL | 1st Column | INTEGER | ROW_INDEX |

| Table | Column | Data Type | Column Name |
|---|---|---|---|
| | 2nd Column | NVARCHAR | MODEL_CONTENT |

The DATA table should be in the exact same format as that in PAL_ROCKET_TRAIN, especially the length and dimensionality of time series. The MODEL table should come from PAL_ROCKET_TRAIN as well.

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION |
|---|---|---|---|
| THREAD_RATIO | DOUBLE | 1.0 | The ratio of available threads.<br>• 0: single thread<br>• 0~1: percentage<br>• Others: heuristically determined |

### Output Table

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| RESULTS | 1st Column | INTEGER | ID | Data id |
| | 2nd Column | NVARCHA | CLASS | Predicted classes |
| | 3rd Column | DOUBLE | PROBABILITY | Confidence of the predictions |

### Examples

Assume that:

- `DM_PAL` is a schema belonging to `USER1`;
- `USER1` has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

#### Example 1: Univariate time series predicted by MiniRocket

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_ROCKET_DATA_TBL;
CREATE COLUMN TABLE PAL_ROCKET_DATA_TBL (
    "RECORD_ID" INTEGER,
    "VAL_01"     DOUBLE,
    "VAL_02"     DOUBLE,
    "VAL_03"     DOUBLE,
    "VAL_04"     DOUBLE,
    "VAL_05"     DOUBLE,
    "VAL_06"     DOUBLE,
    "VAL_07"     DOUBLE,
    "VAL_08"     DOUBLE,
    "VAL_09"     DOUBLE,
```

```
    "VAL_10"     DOUBLE,
    "VAL_11"     DOUBLE,
    "VAL_12"     DOUBLE,
    "VAL_13"     DOUBLE,
    "VAL_14"     DOUBLE,
    "VAL_15"     DOUBLE,
    "VAL_16"     DOUBLE
);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(0,1.598,1.599,1.571,1.550,1.507,1.434,1.369,1.305,1.210,1.117,1.024,0.926,0.828,
0.739,0.643,0.556);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(1,1.701,1.671,1.619,1.547,1.475,1.391,1.306,1.237,1.153,1.070,0.985,0.899,0.816,
0.733,0.658,0.581);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(2,1.722,1.695,1.657,1.606,1.512,1.414,1.314,1.213,1.113,1.015,0.920,0.828,0.740,
0.658,0.586,0.501);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(3,1.726,1.660,1.573,1.496,1.409,1.332,1.246,1.159,1.073,0.987,0.901,0.815,0.730,
0.644,0.558,0.484);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(4,1.779,1.761,1.703,1.611,1.492,1.369,1.245,1.121,1.011,0.900,0.786,0.679,0.580,
0.502,0.415,0.333);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(5,1.800,1.743,1.686,1.633,1.532,1.423,1.311,1.201,1.090,0.979,0.872,0.767,0.664,
0.561,0.453,0.355);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(6,1.749,1.727,1.659,1.560,1.457,1.355,1.252,1.162,1.062,0.961,0.864,0.771,0.682,
0.595,0.513,0.427);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(7,1.348,1.237,1.129,1.022,0.939,0.847,0.750,0.659,0.566,0.474,0.388,0.306,0.218,
0.133,0.061,0.009);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(8,1.696,1.634,1.596,1.507,1.414,1.323,1.242,1.186,1.122,1.048,0.966,0.890,0.805,
0.719,0.632,0.553);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(9,1.723,1.713,1.665,1.587,1.495,1.404,1.312,1.220,1.130,1.041,0.955,0.870,0.787,
0.706,0.622,0.547);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(10,1.614,1.574,1.557,1.521,1.460,1.406,1.316,1.227,1.137,1.045,0.957,0.862,0.771
,0.681,0.587,0.497);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(11,1.652,1.665,1.656,1.623,1.571,1.499,1.412,1.339,1.253,1.155,1.058,0.973,0.877
,0.797,0.704,0.609);
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL (
    PARAM_NAME VARCHAR(100),
    INT_VALUE INTEGER,
    DOUBLE_VALUE DOUBLE,
    STRING_VALUE VARCHAR(100)
);
INSERT INTO PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 1.0, NULL);
CALL _SYS_AFL.PAL_ROCKET_PREDICT (PAL_ROCKET_DATA_TBL, PAL_ROCKET_MODEL_TBL,
PAL_PARAMETER_TBL, ?);
```

RESULT table:

| | ID | CLASS | PROBABILITY |
|---|---|---|---|
| 1 | 0 | A | 0.9999999999996552 |
| 2 | 1 | B | 0.9999999999999996 |
| 3 | 2 | C | 0.9999999999998403 |
| 4 | 3 | A | 1 |
| 5 | 4 | B | 1 |
| 6 | 5 | C | 1 |
| 7 | 6 | A | 1 |
| 8 | 7 | B | 1 |
| 9 | 8 | C | 1 |
| 10 | 9 | B | 1 |
| 11 | 10 | C | 0.9999999999997302 |
| 12 | 11 | A | 1 |

**Example 2: Multivariate time series (with dimensionality 8) predicted by MultiRocket**

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_ROCKET_DATA_TBL;
CREATE COLUMN TABLE PAL_ROCKET_DATA_TBL (
    "RECORD_ID" INTEGER,
    "VAL_01"     DOUBLE,
    "VAL_02"     DOUBLE,
    "VAL_03"     DOUBLE,
    "VAL_04"     DOUBLE,
    "VAL_05"     DOUBLE,
    "VAL_06"     DOUBLE,
    "VAL_07"     DOUBLE,
    "VAL_08"     DOUBLE,
    "VAL_09"     DOUBLE,
    "VAL_10"     DOUBLE,
    "VAL_11"     DOUBLE,
    "VAL_12"     DOUBLE,
    "VAL_13"     DOUBLE,
    "VAL_14"     DOUBLE,
    "VAL_15"     DOUBLE,
    "VAL_16"     DOUBLE
);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(0,1.645,1.646,1.621,1.585,1.540,1.470,1.408,1.318,1.225,1.161,1.070,0.980,0.893,
0.798,0.705,0.620);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(1,1.704,1.705,1.706,1.680,1.632,1.560,1.467,1.376,1.282,1.186,1.090,0.994,0.895,
0.799,0.702,0.605);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(2,1.699,1.666,1.621,1.538,1.454,1.357,1.270,1.173,1.076,0.979,0.885,0.793,0.706,
0.623,0.541,0.460);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(3,1.709,1.663,1.580,1.497,1.413,1.330,1.246,1.163,1.080,0.997,0.913,0.831,0.748,
0.665,0.582,0.509);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(4,1.687,1.688,1.674,1.619,1.531,1.439,1.345,1.253,1.160,1.069,0.977,0.900,0.810,
0.722,0.644,0.557);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(5,1.665,1.623,1.584,1.504,1.420,1.336,1.251,1.167,1.083,0.998,0.913,0.828,0.742,
0.659,0.586,0.507);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(6,1.715,1.680,1.601,1.512,1.422,1.332,1.243,1.162,1.073,0.986,0.899,0.813,0.729,
0.644,0.559,0.475);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(7,1.594,1.586,1.568,1.531,1.464,1.384,1.301,1.221,1.139,1.053,0.968,0.882,0.798,
0.714,0.630,0.547);
```

```
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(8,1.754,1.757,1.736,1.693,1.618,1.527,1.412,1.322,1.202,1.083,0.963,0.874,0.764,
0.663,0.567,0.495);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(9,1.484,1.486,1.513,1.509,1.469,1.455,1.398,1.314,1.232,1.144,1.057,0.963,0.862,
0.767,0.665,0.570);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(10,1.610,1.611,1.577,1.534,1.493,1.436,1.359,1.271,1.193,1.096,1.000,0.903,0.805
,0.712,0.619,0.527);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(11,1.579,1.614,1.623,1.610,1.581,1.535,1.460,1.381,1.311,1.226,1.142,1.058,0.976
,0.892,0.817,0.737);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(12,1.636,1.547,1.458,1.369,1.280,1.199,1.111,1.023,0.935,0.848,0.768,0.681,0.595
,0.509,0.423,0.337);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(13,1.150,1.243,1.328,1.415,1.501,1.588,1.676,1.741,1.728,1.655,1.568,1.490,1.403
,1.317,1.231,1.152);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(14,1.324,1.340,1.190,1.140,1.109,0.995,1.078,0.960,0.853,0.763,0.660,0.535,0.486
,0.457,0.446,0.451);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(15,1.659,1.650,1.603,1.516,1.414,1.322,1.221,1.121,1.028,0.960,0.916,0.884,0.830
,0.776,0.699,0.627);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(16,1.793,1.765,1.717,1.624,1.517,1.409,1.302,1.197,1.091,0.989,0.888,0.802,0.702
,0.608,0.513,0.420);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(17,1.637,1.614,1.575,1.506,1.424,1.342,1.268,1.182,1.097,1.010,0.923,0.838,0.751
,0.665,0.592,0.508);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(18,1.707,1.677,1.632,1.570,1.478,1.384,1.288,1.192,1.115,1.022,0.928,0.833,0.741
,0.653,0.560,0.472);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(19,1.821,1.794,1.798,1.775,1.720,1.669,1.557,1.443,1.332,1.204,1.081,0.953,0.836
,0.717,0.609,0.508);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(20,1.609,1.610,1.597,1.586,1.539,1.498,1.424,1.339,1.242,1.140,1.035,0.932,0.826
,0.722,0.617,0.515);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(21,1.409,1.507,1.597,1.689,1.747,1.758,1.723,1.637,1.549,1.458,1.369,1.280,1.189
,1.099,1.010,0.920);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(22,1.676,1.650,1.582,1.495,1.400,1.317,1.231,1.133,1.039,0.943,0.848,0.756,0.665
,0.572,0.481,0.396);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(23,1.575,1.551,1.530,1.504,1.458,1.392,1.313,1.216,1.121,1.022,0.923,0.825,0.728
,0.626,0.530,0.440);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(24,1.722,1.723,1.696,1.651,1.566,1.471,1.377,1.282,1.187,1.091,0.993,0.896,0.798
,0.700,0.604,0.510);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(25,1.627,1.609,1.592,1.572,1.532,1.451,1.361,1.271,1.175,1.078,0.982,0.882,0.786
,0.686,0.590,0.494);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(26,1.684,1.656,1.601,1.517,1.432,1.346,1.261,1.176,1.106,1.022,0.939,0.856,0.772
,0.688,0.604,0.522);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(27,1.697,1.665,1.590,1.508,1.424,1.341,1.258,1.175,1.092,1.009,0.926,0.844,0.760
,0.678,0.595,0.513);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(28,1.406,1.320,1.234,1.148,1.063,0.978,0.893,0.809,0.725,0.642,0.558,0.477,0.396
,0.314,0.234,0.153);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(29,1.592,1.593,1.571,1.551,1.527,1.475,1.410,1.346,1.252,1.160,1.058,0.956,0.859
,0.763,0.668,0.574);
```

```
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(30,1.688,1.648,1.570,1.490,1.408,1.327,1.246,1.164,1.092,1.011,0.930,0.849,0.768
,0.687,0.606,0.524);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(31,1.708,1.663,1.595,1.504,1.411,1.318,1.225,1.134,1.042,0.951,0.861,0.794,0.704
,0.614,0.529,0.446);
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL (
    PARAM_NAME VARCHAR(100),
    INT_VALUE INTEGER,
    DOUBLE_VALUE DOUBLE,
    STRING_VALUE VARCHAR(100)
);
INSERT INTO PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 1.0, NULL);
CALL _SYS_AFL.PAL_ROCKET_PREDICT (PAL_ROCKET_DATA_TBL, PAL_ROCKET_MODEL_TBL,
PAL_PARAMETER_TBL, ?);
```

RESULT table:

| | ID | CLASS | PROBABILITY |
|---|---|---|---|
| 1 | 0 | A | 0.999999999999547 |
| 2 | 1 | B | 0.9999999999999991 |
| 3 | 2 | C | 0.9999999999998839 |
| 4 | 3 | A | 0.9999999999997486 |

## _SYS_AFL.PAL_ROCKET_FIT

This function fits the time series and generates the model for further transformation.

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | <DATA table> | IN |
| 2 | <PARAMETER table> | IN |
| 3 | <MODEL table> | OUT |

### Input Table

| Table | Column | Data Type | Description |
|---|---|---|---|
| DATA | 1st Column | INTEGER | Record ID |
| | 2nd Column | INTEGER, DOUBLE | Time series value |

For univariate time series, each row represents one time series, while for multivariate time series, a fixed number of consecutive rows forms one time series, and that number is designated by the parameter DATA_DIM.

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| METHOD | INTEGER | 0 | • 0: MiniRocket<br>• 1: MultiRocket | |
| NUM_FEATURES | INTEGER | • MiniRocket: 9996<br>• Multi-Rocket:49728 | Number of transformed features for each time series | |
| DATA_DIM | INTEGER | 1 | 1 for univariate time series and others for multivariate | Cannot be smaller than 1 |
| RANDOM_SEED | INTEGER | 0 | 0 indicates using machine time as seed. | |

### Output Table

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| MODEL | 1st Column | INTEGER | ROW_INDEX | Indicates the ID of the current row. |
| | 2nd Column | NVARCHAR | MODEL_CONTENT | The first row represents the method name and others are JSON strings. |

## _SYS_AFL.PAL_ROCKET_TRANSFORM

This function transforms the time series by the fitted model table.

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | <DATA table> | IN |
| 2 | <MODEL table> | IN |
| 3 | <PARAMETER table> | IN |
| 4 | <FEATURES table> | OUT |

### Input Table

| Table | Column | Data Type | Column Name |
|---|---|---|---|
| DATA | 1st Column | INTEGER | Record ID |
| | 2nd Column | INTEGER, DOUBLE | Time series value |
| MODEL | 1st Column | INTEGER | ROW_INDEX |

| Table | Column | Data Type | Column Name |
|---|---|---|---|
| | 2nd Column | NVARCHAR | MODEL_CONTENT |

The DATA table should be in the exact same format as that in PAL_ROCKET_FIT, especially the length and dimensionality of time series. The MODEL table should come from PAL_ROCKET_FIT as well.

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION |
|---|---|---|---|
| THREAD_RATIO | DOUBLE | 1.0 | The ratio of available threads.<br>• 0: single thread<br>• 0~1: percentage<br>• Others: heuristically determined |

### Output Table

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| FEATURES | 1st Column | INTEGER | ROW_INDEX | Indicates the ID of current row. |
| | 2nd Column | NVARCHAR | MODEL_CONTENT | Transformed features written in JSON strings |

### Examples

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

#### Example 1: Univariate time series fitted and transformed by MiniRocket

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_ROCKET_DATA_TBL;
CREATE COLUMN TABLE PAL_ROCKET_DATA_TBL (
    "RECORD_ID" INTEGER,
    "VAL_01"     DOUBLE,
    "VAL_02"     DOUBLE,
    "VAL_03"     DOUBLE,
    "VAL_04"     DOUBLE,
    "VAL_05"     DOUBLE,
    "VAL_06"     DOUBLE,
    "VAL_07"     DOUBLE,
    "VAL_08"     DOUBLE,
    "VAL_09"     DOUBLE,
    "VAL_10"     DOUBLE,
```

```
    "VAL_11"     DOUBLE,
    "VAL_12"     DOUBLE,
    "VAL_13"     DOUBLE,
    "VAL_14"     DOUBLE,
    "VAL_15"     DOUBLE,
    "VAL_16"     DOUBLE
);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(0,1.598,1.599,1.571,1.550,1.507,1.434,1.369,1.305,1.210,1.117,1.024,0.926,0.828,
0.739,0.643,0.556);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(1,1.701,1.671,1.619,1.547,1.475,1.391,1.306,1.237,1.153,1.070,0.985,0.899,0.816,
0.733,0.658,0.581);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(2,1.722,1.695,1.657,1.606,1.512,1.414,1.314,1.213,1.113,1.015,0.920,0.828,0.740,
0.658,0.586,0.501);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(3,1.726,1.660,1.573,1.496,1.409,1.332,1.246,1.159,1.073,0.987,0.901,0.815,0.730,
0.644,0.558,0.484);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(4,1.779,1.761,1.703,1.611,1.492,1.369,1.245,1.121,1.011,0.900,0.786,0.679,0.580,
0.502,0.415,0.333);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(5,1.800,1.743,1.686,1.633,1.532,1.423,1.311,1.201,1.090,0.979,0.872,0.767,0.664,
0.561,0.453,0.355);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(6,1.749,1.727,1.659,1.560,1.457,1.355,1.252,1.162,1.062,0.961,0.864,0.771,0.682,
0.595,0.513,0.427);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(7,1.348,1.237,1.129,1.022,0.939,0.847,0.750,0.659,0.566,0.474,0.388,0.306,0.218,
0.133,0.061,0.009);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(8,1.696,1.634,1.596,1.507,1.414,1.323,1.242,1.186,1.122,1.048,0.966,0.890,0.805,
0.719,0.632,0.553);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(9,1.723,1.713,1.665,1.587,1.495,1.404,1.312,1.220,1.130,1.041,0.955,0.870,0.787,
0.706,0.622,0.547);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(10,1.614,1.574,1.557,1.521,1.460,1.406,1.316,1.227,1.137,1.045,0.957,0.862,0.771
,0.681,0.587,0.497);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(11,1.652,1.665,1.656,1.623,1.571,1.499,1.412,1.339,1.253,1.155,1.058,0.973,0.877
,0.797,0.704,0.609);
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL (
    PARAM_NAME VARCHAR(100),
    INT_VALUE INTEGER,
    DOUBLE_VALUE DOUBLE,
    STRING_VALUE VARCHAR(100)
);
-- PAL_ROCKET_FIT part
INSERT INTO PAL_PARAMETER_TBL VALUES ('METHOD', 0, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('RANDOM_SEED', 1, NULL, NULL);
DROP TABLE PAL_ROCKET_MODEL_TBL; -- for predict followed
CREATE COLUMN TABLE PAL_ROCKET_MODEL_TBL (
    ROW_INDEX INTEGER,
    MODEL_CONTENT NVARCHAR(5000)
);
DO BEGIN
    lt_data  = SELECT * FROM PAL_ROCKET_DATA_TBL;
    lt_param = SELECT * FROM PAL_PARAMETER_TBL;
    CALL _SYS_AFL.PAL_ROCKET_FIT (:lt_data, :lt_param, lt_model);
    INSERT INTO PAL_ROCKET_MODEL_TBL  SELECT * FROM :lt_model;
END;
SELECT * FROM PAL_ROCKET_MODEL_TBL;
-- PAL_ROCKET_TRANSFORM part
TRUNCATE TABLE PAL_PARAMETER_TBL;
INSERT INTO PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 1.0, NULL);
```

```
DROP TABLE PAL_ROCKET_FEATURES_TBL; -- for predict followed
CREATE COLUMN TABLE PAL_ROCKET_FEATURES_TBL (
    ROW_INDEX INTEGER,
    FEATURE_CONTENT NVARCHAR(5000)
);
DO BEGIN
    lt_data  = SELECT * FROM PAL_ROCKET_DATA_TBL;
    lt_model = SELECT * FROM PAL_ROCKET_MODEL_TBL;
    lt_param = SELECT * FROM PAL_PARAMETER_TBL;
    CALL _SYS_AFL.PAL_ROCKET_TRANSFORM (:lt_data, :lt_model, :lt_param, lt_feat);
    INSERT INTO PAL_ROCKET_FEATURES_TBL SELECT * FROM :lt_feat;
END;
SELECT * FROM PAL_ROCKET_FEATURES_TBL;
```

### Expected Result

MODEL table:

| | ROW_INDEX | MODEL_CONTENT |
|---|---|---|
| 1 | -1 | MiniRocket |
| 2 | 0 | {"SERIES_LENGTH":16,"NUM_CHANNELS":1,"BIAS_SIZE":9996,"DILATIONS":[{"DILATION_SIZE":1,"NUM_FEATURES":119}],"BIASES":[2.385419060381160 |
| 3 | 1 | 66098464,1.0396523357230486,2.005001093476571,1.7872128802451375,-1.2951776858372828,1.9089999999999994,1.3850813175719348,-3.6421418 |
| 4 | 2 | 73,1.3764943031483486,1.1499780774072535,-3.119415995567719,1.3618215349508707,0.9461612896983989,1.4575832409033394,1.28443389706986 |
| 5 | 3 | 485649487700363,-4.853621348269184,0.8749419076651069,0.02471668726256818,0.9647594886575343,0.8512189317437576,-3.4034453898605405,0 |
| 6 | 4 | 2532,1.5195149626729508,-0.39726233250070988,1.6307027845688616,1.1747634106619607,-2.353311078837883,1.5846520355300456,0.5844323663 |
| 7 | 5 | 40499432960379,-2.3215022315852509,0.827924709008419,0.701975985290604,0.8674254174964419,0.7807669820501145,-0.9404343044089281,0.83 |
| 8 | 6 | ,1.3041661512718297,0.5998381533021405,1.4075106120727759,1.2378664004701428,-0.5964396623017748,1.3408241667701664,0.935707396371716 |
| 9 | 7 | 849631812,0.08118405775750475,0.8163095758206369,0.603630112937346,-2.1313585876086797,0.7841220611534585,0.4539821511206032,0.855972 |
| 10 | 8 | 736,0.23902987389749559,0.4489183930819394,0.3044115545293888,-1.798748403065718,0.3508187281167353,0.2974580548400582,0.472003064226 |
| 11 | 9 | 5341669654548,0.5134032017462823,0.22743196208537504,0.5598286495843399,0.40147312400225046,-1.7145372144801372,0.5276035162454504,0. |
| 12 | 10 | 1679220880409,0.23200000000000035,0.05223968771721317,-0.1627015599043082,0.21072808946331038,0.02662114188565101,-2.643993024906784, |
| 13 | 11 | 4327921,0.02794822197938287 3,-0.2385910594072971 4,-2.262874942639753,-0.1549546803798016,-0.2517505891361507,0.1394439117609464,-0.2180 |
| 14 | 12 | 55190862230119,0.00072099329773878 63,-0.5046685622702566,-1.8366080144506399,-0.3348163531038023,-0.5150808618588123,0.22382138060407 |
| 15 | 13 | 050256,2.659055764384081,1.5803473857776378,0.5726893606046842,1.6531783686003617,1.4366720611027709,-2.4937537939283676,1.6279051359 |
| 16 | 14 | 484979562,-0.1341261168395503,1.0819657706657505,0.768113294432584,2.4441035120642859,1.0522265270086218,0.3907275075467551,1.0989233 |
| 17 | 15 | 386552,0.49164073827549178,-2.6849589374564677,0.5328108380823792,0.4574717140626201,0.5761955672570505,0.5191146035261021,-0.4000742 |
| 18 | 16 | 6848106686,0.13474235566841334,1.0940530020988475,0.906787565208205,2.305092159557022,1.0700469202584262,0.4689054334779451,1.112394 |
| 19 | 17 | .2829276713625599,0.23814967979168109,-1.2324231124041798,0.26346287361256739,0.21892471443191356,0.3228922987149089,0.2543802696665 |
| 20 | 18 | 7462970147057,2.0147542147589414,0.5164688714633022,0.12305691384230194,0.572139175820652,0.4977477340776675,-0.37389607355510109,0.5 |

FEATURES table:

| | ROW_INDEX | |
|---|---|---|
| 1 | 0 | {"NUM_FEATURES_PER_DATA":9996,"FEATURES":[{"DATA_ID":0,"FEATURE_VALS":[0.375,0.75,0.3125,0.5,0.8125,0.375,0.625,0.25,0.4375,0.75,0.375,0.5625 |
| 2 | 1 | ,0.375,0.875,0.125,0.5,1.0,0.25,0.75,1.0,0.375,1.0,0.125,0.5,1.0,0.25,0.875,0.125,0.5,1.0,0.125,0.625,1.0,0.375,1.0,0.125,0.5,1.0,0.25,0.75,1.0,0.375,1.0,0.125,0. |
| 3 | 2 | .625,1.0,0.375,0.875,0.125,0.5,1.0,0.125,0.875,0.0,0.375,1.0,0.125,0.625,1.0,0.125,0.875,0.0,0.5,1.0,0.125,0.75,1.0,0.375,0.875,0.125,0.625,1.0,0.125,0.875,0.0 |
| 4 | 3 | 0.5625,0.0625,0.25,0.75,0.1875,0.3125,0.875,0.1875,0.625,0.1875,0.25,0.875,0.0,0.0,1.0,0.0,0.125,0.0,0.0,1.0,0.0,0.0,0.0,0.0,0.125,0.0,0.0,1.0,0.0,0.0,0.0,0.0,1.0 |
| 5 | 4 | 25,0.4375,0.6875,0.875,0.4375,0.8125,0.375,0.5,0.8125,0.4375,0.75,0.0,0.4375,0.8125,0.4375,0.6875,0.875,0.4375,0.8125,0.375,0.375,1.0,0.375,0.875,1.0,0.3 |
| 6 | 5 | ,0.1875,0.25,0.9375,0.1875,0.3125,0.1875,0.25,0.75,0.1875,0.3125,0.0625,0.1875,0.375,0.1875,0.25,0.875,0.1875,0.3125,0.1875,0.1875,0.375,0.1875,0.25,0.9 |
| 7 | 6 | 5,0.5,0.75,0.375,0.5625,0.125,0.5,0.5625,0.25,0.5,0.8125,0.5,0.5625,0.125,0.5,0.75,0.3125,0.5,0.125,0.5,0.5625,0.25,0.5,0.75,0.4375,0.5625,0.125,0.5,0.6875,0 |
| 8 | 7 | 75,1.0,0.8125,0.9375,0.0625,0.9375,1.0,0.375,0.9375,1.0,0.875,1.0,0.1875,0.9375,1.0,0.8125,0.9375,0.0625,0.9375,1.0,0.3125,0.9375,1.0,0.8125,0.9375,0.125 |
| 9 | 8 | 0.0,0.5,0.875,0.0,0.75,1.0,0.375,0.75,0.0,0.75,1.0,0.25,0.75,0.0,0.375,0.75,0.0,0.75,1.0,0.375,0.75,0.0,0.75,0.875,0.0,0.75,1.0,0.375,0.75,0.0,0.4375,0.75,0.25,0.5 |
| 10 | 9 | 5,0.0,0.625,0.875,0.0,0.75,0.875,0.375,0.875,0.0,0.75,0.875,0.0,0.875,1.0,0.625,0.875,0.0,0.75,0.875,0.25,0.875,0.0,0.625,0.875,0.0,0.75,0.875,0.5,0.875,0.0,0.7 |
| 11 | 10 | 25,0.4375,0.6875,0.25,0.625,0.8125,0.25,0.75,0.0,0.625,0.875,0.0,0.75,0.0,0.5,0.75,0.0,0.625,0.875,0.25,0.75,0.0,0.625,0.875,0.0,0.625,1.0,0.375,0.75,0.0,0.625 |
| 12 | 11 | 5,0.8125,0.25,0.75,0.25,0.25,0.75,0.25,0.6875,0.875,0.25,0.75,0.25,0.3125,0.8125,0.25,0.75,0.25,0.25,0.75,0.25,0.625,0.8125,0.25,0.75,0.25,0.25,1.0,0.0,1.0,1.0 |
| 13 | 12 | 0.0,1.0,0.0,0.875,1.0,0.0,1.0,0.0,0.0,1.0,0.0,1.0,1.0,0.0,1.0,0.0,0.75,1.0,0.0,1.0,0.0,0.0,1.0,0.0,1.0,1.0,0.0,1.0,0.0,0.375,1.0,0.0,1.0,0.0,0.0,1.0,0.0,0.75,1.0,0.0,1.0,0 |
| 14 | 13 | ,0.0,0.0,0.0,1.0,0.0,0.0,1.0,0.0,0.125,0.0,0.0,1.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,1.0,0.0,0.625,0.0,0.0,1.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,1.0,0.0 |
| 15 | 14 | ,1.0,0.0,0.5,0.0,0.0,1.0,0.0,0.125,0.0,0.0,0.75,0.0,0.125,1.0,0.0,0.5,0.0,0.0,1.0,0.0,0.125,1.0,0.0,0.5,0.0,0.0,1.0,0.0,0.125,0.0,0.0,0.875,0.0,0.125,1.0,0.0,0.5,0.0,0.0 |
| 16 | 15 | 0.0,0.25,0.0,0.0,1.0,0.0,0.0,0.0,0.0,0.25,0.0,0.0,1.0,0.0,0.25,0.0,0.0,1.0,0.0,0.0,0.0,0.0,0.25,0.0,0.0,1.0,0.0,0.125,0.0,0.0,1.0,0.0,0.0,1.0,0.0,0.25,0.0,0.0,1.0,0.0,0.0, |
| 17 | 16 | 5,0.375,0.8125,1.0,0.5,1.0,0.375,1.0,1.0,0.5,1.0,0.0,0.75,1.0,0.375,1.0,1.0,0.5,1.0,0.0,1.0,1.0,0.5,1.0,0.0,0.5,1.0,0.375,1.0,1.0,0.5,1.0,0.0,0.75,1.0,0.5,1.0,1.0,0.5,1 |
| 18 | 17 | ,0.25,1.0,0.0,0.625,1.0,0.125,1.0,0.0,0.25,1.0,0.0,0.875,1.0,0.125,1.0,0.0,0.5,1.0,0.125,1.0,0.0,0.25,1.0,0.0,0.625,1.0,0.125,1.0,0.0,0.25,1.0,0.0,1.0,1.0,0.25,1.0,0.0 |
| 19 | 18 | ,0.9375,0.9375,0.9375,0.125,0.9375,0.9375,0.375,0.9375,0.9375,0.9375,0.9375,0.1875,0.9375,0.9375,0.0,1.0,0.0,0.0,1.0,0.0,0.25,1.0,0.0,1.0,0.0,0.0,1.0,0.0,1.0, |

### Example 2: Multivariate time series (with dimensionality 8) fitted and transformed by MultiRocket

```
SET SCHEMA DM_PAL;
```

```
DROP TABLE PAL_ROCKET_DATA_TBL;
CREATE COLUMN TABLE PAL_ROCKET_DATA_TBL (
    "RECORD_ID" INTEGER,
    "VAL_01"     DOUBLE,
    "VAL_02"     DOUBLE,
    "VAL_03"     DOUBLE,
    "VAL_04"     DOUBLE,
    "VAL_05"     DOUBLE,
    "VAL_06"     DOUBLE,
    "VAL_07"     DOUBLE,
    "VAL_08"     DOUBLE,
    "VAL_09"     DOUBLE,
    "VAL_10"     DOUBLE,
    "VAL_11"     DOUBLE,
    "VAL_12"     DOUBLE,
    "VAL_13"     DOUBLE,
    "VAL_14"     DOUBLE,
    "VAL_15"     DOUBLE,
    "VAL_16"     DOUBLE
);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(0,1.645,1.646,1.621,1.585,1.540,1.470,1.408,1.318,1.225,1.161,1.070,0.980,0.893,
0.798,0.705,0.620);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(1,1.704,1.705,1.706,1.680,1.632,1.560,1.467,1.376,1.282,1.186,1.090,0.994,0.895,
0.799,0.702,0.605);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(2,1.699,1.666,1.621,1.538,1.454,1.357,1.270,1.173,1.076,0.979,0.885,0.793,0.706,
0.623,0.541,0.460);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(3,1.709,1.663,1.580,1.497,1.413,1.330,1.246,1.163,1.080,0.997,0.913,0.831,0.748,
0.665,0.582,0.509);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(4,1.687,1.688,1.674,1.619,1.531,1.439,1.345,1.253,1.160,1.069,0.977,0.900,0.810,
0.722,0.644,0.557);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(5,1.665,1.623,1.584,1.504,1.420,1.336,1.251,1.167,1.083,0.998,0.913,0.828,0.742,
0.659,0.586,0.507);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(6,1.715,1.680,1.601,1.512,1.422,1.332,1.243,1.162,1.073,0.986,0.899,0.813,0.729,
0.644,0.559,0.475);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(7,1.594,1.586,1.568,1.531,1.464,1.384,1.301,1.221,1.139,1.053,0.968,0.882,0.798,
0.714,0.630,0.547);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(8,1.754,1.757,1.736,1.693,1.618,1.527,1.412,1.322,1.202,1.083,0.963,0.874,0.764,
0.663,0.567,0.495);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(9,1.484,1.486,1.513,1.509,1.469,1.455,1.398,1.314,1.232,1.144,1.057,0.963,0.862,
0.767,0.665,0.570);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(10,1.610,1.611,1.577,1.534,1.493,1.436,1.359,1.271,1.193,1.096,1.000,0.903,0.805
,0.712,0.619,0.527);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(11,1.579,1.614,1.623,1.610,1.581,1.535,1.460,1.381,1.311,1.226,1.142,1.058,0.976
,0.892,0.817,0.737);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(12,1.636,1.547,1.458,1.369,1.280,1.199,1.111,1.023,0.935,0.848,0.768,0.681,0.595
,0.509,0.423,0.337);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(13,1.150,1.243,1.328,1.415,1.501,1.588,1.676,1.741,1.728,1.655,1.568,1.490,1.403
,1.317,1.231,1.152);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(14,1.324,1.340,1.190,1.140,1.109,0.995,1.078,0.960,0.853,0.763,0.660,0.535,0.486
,0.457,0.446,0.451);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(15,1.659,1.650,1.603,1.516,1.414,1.322,1.221,1.121,1.028,0.960,0.916,0.884,0.830
,0.776,0.699,0.627);
```

```
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(16,1.793,1.765,1.717,1.624,1.517,1.409,1.302,1.197,1.091,0.989,0.888,0.802,0.702
,0.608,0.513,0.420);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(17,1.637,1.614,1.575,1.506,1.424,1.342,1.268,1.182,1.097,1.010,0.923,0.838,0.751
,0.665,0.592,0.508);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(18,1.707,1.677,1.632,1.570,1.478,1.384,1.288,1.192,1.115,1.022,0.928,0.833,0.741
,0.653,0.560,0.472);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(19,1.821,1.794,1.798,1.775,1.720,1.669,1.557,1.443,1.332,1.204,1.081,0.953,0.836
,0.717,0.609,0.508);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(20,1.609,1.610,1.597,1.586,1.539,1.498,1.424,1.339,1.242,1.140,1.035,0.932,0.826
,0.722,0.617,0.515);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(21,1.409,1.507,1.597,1.689,1.747,1.758,1.723,1.637,1.549,1.458,1.369,1.280,1.189
,1.099,1.010,0.920);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(22,1.676,1.650,1.582,1.495,1.400,1.317,1.231,1.133,1.039,0.943,0.848,0.756,0.665
,0.572,0.481,0.396);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(23,1.575,1.551,1.530,1.504,1.458,1.392,1.313,1.216,1.121,1.022,0.923,0.825,0.728
,0.626,0.530,0.440);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(24,1.722,1.723,1.696,1.651,1.566,1.471,1.377,1.282,1.187,1.091,0.993,0.896,0.798
,0.700,0.604,0.510);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(25,1.627,1.609,1.592,1.572,1.532,1.451,1.361,1.271,1.175,1.078,0.982,0.882,0.786
,0.686,0.590,0.494);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(26,1.684,1.656,1.601,1.517,1.432,1.346,1.261,1.176,1.106,1.022,0.939,0.856,0.772
,0.688,0.604,0.522);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(27,1.697,1.665,1.590,1.508,1.424,1.341,1.258,1.175,1.092,1.009,0.926,0.844,0.760
,0.678,0.595,0.513);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(28,1.406,1.320,1.234,1.148,1.063,0.978,0.893,0.809,0.725,0.642,0.558,0.477,0.396
,0.314,0.234,0.153);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(29,1.592,1.593,1.571,1.551,1.527,1.475,1.410,1.346,1.252,1.160,1.058,0.956,0.859
,0.763,0.668,0.574);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(30,1.688,1.648,1.570,1.490,1.408,1.327,1.246,1.164,1.092,1.011,0.930,0.849,0.768
,0.687,0.606,0.524);
INSERT INTO PAL_ROCKET_DATA_TBL VALUES
(31,1.708,1.663,1.595,1.504,1.411,1.318,1.225,1.134,1.042,0.951,0.861,0.794,0.704
,0.614,0.529,0.446);
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL (
    PARAM_NAME VARCHAR(100),
    INT_VALUE INTEGER,
    DOUBLE_VALUE DOUBLE,
    STRING_VALUE VARCHAR(100)
);
-- PAL_ROCKET_FIT part
INSERT INTO PAL_PARAMETER_TBL VALUES ('METHOD',      1, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('DATA_DIM',    8, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('RANDOM_SEED', 1, NULL, NULL);
DROP TABLE PAL_ROCKET_MODEL_TBL; -- for predict followed
CREATE COLUMN TABLE PAL_ROCKET_MODEL_TBL (
    ROW_INDEX INTEGER,
    MODEL_CONTENT NVARCHAR(5000)
);
DO BEGIN
    lt_data  = SELECT * FROM PAL_ROCKET_DATA_TBL;
    lt_param = SELECT * FROM PAL_PARAMETER_TBL;
    CALL _SYS_AFL.PAL_ROCKET_FIT (:lt_data, :lt_param, lt_model);
```

```
    INSERT INTO PAL_ROCKET_MODEL_TBL  SELECT * FROM :lt_model;
END;
SELECT * FROM PAL_ROCKET_MODEL_TBL;
-- PAL_ROCKET_TRANSFORM part
TRUNCATE TABLE PAL_PARAMETER_TBL;
INSERT INTO PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 1.0, NULL);
DROP TABLE PAL_ROCKET_FEATURES_TBL; -- for predict followed
CREATE COLUMN TABLE PAL_ROCKET_FEATURES_TBL (
    ROW_INDEX INTEGER,
    FEATURE_CONTENT NVARCHAR(5000)
);
DO BEGIN
    lt_data  = SELECT * FROM PAL_ROCKET_DATA_TBL;
    lt_model = SELECT * FROM PAL_ROCKET_MODEL_TBL;
    lt_param = SELECT * FROM PAL_PARAMETER_TBL;
    CALL _SYS_AFL.PAL_ROCKET_TRANSFORM (:lt_data, :lt_model, :lt_param, lt_feat);
    INSERT INTO PAL_ROCKET_FEATURES_TBL SELECT * FROM :lt_feat;
END;
SELECT * FROM PAL_ROCKET_FEATURES_TBL;
```

### Expected Result

MODEL table:

| | ROW_INDEX | |
|---|---|---|
| 1 | -1 | MultiRocket |
| 2 | 0 | {"SERIES_LENGTH":16,"NUM_CHANNELS":8,"BIAS_SIZE":6216,"BIAS_DIFF_SIZE":6216,"DILATIONS":[{"DILATION_SIZE":1,"NUM_FEATURES":74}],"DILATIC |
| 3 | 1 | HANNELS":[6]},{"ID":77,"CHANNELS":[1,4,7,6,5]},{"ID":78,"CHANNELS":[7,0,5,6]},{"ID":79,"CHANNELS":[1,4]},{"ID":80,"CHANNELS":[1,4,6,2]},{"ID":81,"CHANN |
| 4 | 2 | 504,-22.340878522815215,7.959895076819708,5.81470488591989,11.8578032022155,7.627485955788726,-13.98688419511595,8.123306265049058,6.81 |
| 5 | 3 | 48,9.013306944001223,-18.05915327183857,9.197905923784694,8.537362238736196,13.965540234001758,9.113025371001515,-5.43163978918106,10.8 |
| 6 | 4 | 8578,-5.9781939194202799,2.23437952460736,1.3156543834261514,3.434864760842622,1.9573681596337263,-3.037385353745023,2.36922325038982( |
| 7 | 5 | 07,9.199049371639408,7.1111111151007428,16.259771061331127,8.668296377734663,2.860724791448105,10.50366180880851,7.903209807370763,-22 |
| 8 | 6 | .1919862979765044,0.27121051666530307,3.4304422071978797,0.9158462448828436,-1.38406311309445,1.345609162454411,0.7418658591550984,-1. |
| 9 | 7 | 530387977505125,-2.0780372353665874,-0.04089799798518077,-0.18266004693153657,0.5731650334644877,-0.12163461524651172,-0.740965960776 |
| 10 | 8 | 81529759048407,-0.4224908796580212,0.3050867965370993,-0.36966210167075566,-0.9311206723207743,-0.3142002624633562,-0.389138707648494 |
| 11 | 9 | 0869926636,8.950068054111778,21.702397663867726,12.027393495455799,-1.3422272814195898,17.231804326322768,10.819741185385425,-53.10750 |
| 12 | 10 | ,2.6450713593122048,10.231613347274037,4.319696925529633,-1.0621511766606747,5.884599388593907,3.0121746460888225,-21.654100287036856, |
| 13 | 11 | 621435368,2.409440699051263,2.213212553521503,6.010489226247179,2.35451730675841,2.0081136830140466,3.8239394250623,2.319601952984093 |
| 14 | 12 | 8979174,0.00015035332044800227,-2.813201738058908,0.0016913729829520068,2.390383211499055,8.073118777056227,2.758017414860195,1.32571 |
| 15 | 13 | 1585639064478836,-0.05497382428246635,-0.445856425354717,2.747894240735274,-0.42178556352960719,-0.6521995643147284,0.155394772894950 |
| 16 | 14 | .999937238989194,-2.5241789509560519,-0.978445003039326,-1.0457997212660993,2.4052241196023949,-0.9925299868357713,-1.775116468448252,- |
| 17 | 15 | 2060744629,14.773187672699806,6.18556909943719,-21.440110881626237,8.012145211769744,5.506329908183801,16.51463919949335,6.7479863429 |
| 18 | 16 | 29545709900083907,0.011287594156603989,-0.34922107956267497,0.10739853849382485,-0.006330951674932782,1.8079908408810182,-0.727010617 |
| 19 | 17 | 5270445,0.05339909989038331,-0.26186436361349627,1.7113432481556519,-0.250402174376644,-0.27085955026450805,0.13376350872295884,-0.259 |

FEATURES table:

| | ROW_INDEX | |
|---|---|---|
| 1 | 0 | {"NUM_FEATURES_PER_DATA":49728,"FEATURES":[{"DATA_ID":0,"FEATURE_VALS":[0.375,0.75,0.1875,0.5,0.875,0.3125,0.6875,0.0625,0.4375,0.8125,0.25,0 |
| 2 | 1 | 0625,0.5625,0.8125,0.125,0.6875,0.9375,0.5,0.6875,0.0625,0.625,0.875,0.125,0.6875,0.0,0.5,0.75,0.125,0.6875,0.9375,0.4375,0.6875,0.0625,0.5625,0.8125,0. |
| 3 | 2 | ,0.0625,0.5,0.8125,0.4375,0.5625,0.9375,0.5,0.75,0.125,0.5,0.875,0.5,0.625,0.0,0.625,1.0,0.125,0.75,1.0,0.375,1.0,0.0,0.625,1.0,0.25,0.875,1.0,0.5,1.0,0.0,0.625 |
| 4 | 3 | 0,1.0,0.0,1.0,0.0,1.0,1.0,0.0,1.0,1.0,0.75,1.0,0.0,1.0,1.0,0.0,1.0,1.0,1.0,1.0,0.0,1.0,1.0,0.375,1.0,0.0,1.0,1.0,0.0,1.0,1.0,0.875,1.0,0.0,0.375,0.8125,0.25,0.6875,0.93 |
| 5 | 4 | 5,0.3125,0.0625,0.3125,0.375,0.1875,0.3125,0.9375,0.3125,0.3125,0.0625,0.3125,0.5625,0.25,0.3125,0.9375,0.3125,0.3125,0.125,0.3125,0.9375,0.25,0.3125,( |
| 6 | 5 | ,0.3125,0.75,0.125,0.75,0.875,0.25,0.75,0.0625,0.4375,0.8125,0.1875,0.75,0.9375,0.3125,0.75,0.0625,0.75,0.875,0.25,0.75,1.0,0.375,0.75,0.1875,0.75,0.875,0.2 |
| 7 | 6 | 0.25,1.0,0.0,0.375,0.0,0.125,0.75,0.0,0.25,1.0,0.125,0.75,0.0,0.25,1.0,0.0,0.375,0.0,0.125,0.75,0.0,0.25,1.0,0.0,0.5,0.0,0.125,1.0,0.0,0.375,0.0,0.125,0.75,0.0,0.25 |
| 8 | 7 | 2.0,3.0,12.0,3.0,8.0,13.0,3.0,7.0,1.0,6.0,7.0,6.0,7.0,1.0,6.0,7.0,6.0,6.0,7.0,6.0,7.0,1.0,6.0,7.0,6.0,6.0,7.0,6.0,7.0,2.0,6.0,7.0,6.0,7.0,1.0,6.0,7.0,6.0,6.0,7.0,6.0,7.0,1.0 |
| 9 | 8 | .0,1.0,7.0,7.0,6.0,7.0,1.0,6.0,7.0,5.0,7.0,7.0,6.0,7.0,1.0,7.0,7.0,6.0,7.0,1.0,6.0,7.0,1.0,7.0,7.0,6.0,7.0,1.0,6.0,7.0,5.0,7.0,7.0,6.0,7.0,1.0,7.0,7.0,6.0,7.0,1.0,6.0,7.0,5.0, |
| 10 | 9 | .0,7.0,1.0,7.0,7.0,1.0,7.0,7.0,7.0,7.0,1.0,7.0,7.0,5.0,7.0,1.0,7.0,7.0,1.0,7.0,7.0,7.0,7.0,1.0,7.0,7.0,1.0,7.0,1.0,7.0,7.0,1.0,7.0,7.0,7.0,7.0,1.0,7.0,7.0,1.0,7.0,7.0,7.0,7.0, |
| 11 | 10 | ,1.0,5.0,1.0,3.0,7.0,1.0,5.0,1.0,1.0,5.0,1.0,4.0,7.0,1.0,5.0,1.0,2.0,7.0,1.0,5.0,7.0,1.0,5.0,1.0,3.0,7.0,1.0,5.0,1.0,2.0,7.0,1.0,4.0,7.0,1.0,7.0,3.0,5.0,8.0,3.0,7.0,2.0,4.0,7. |
| 12 | 11 | 15.0,4.0,12.0,3.0,5.0,12.0,4.0,12.0,2.0,4.0,12.0,3.0,6.0,15.0,4.0,12.0,2.0,5.0,12.0,4.0,12.0,15.0,4.0,12.0,3.0,5.0,12.0,4.0,12.0,2.0,4.0,12.0,3.0,12.0,15.0,4.0,12.0,3.( |
| 13 | 12 | 34622,24.090257,40.909071,31.131432,-28.296288,34.176641,8.058914,0.0,8.389983,0.788176,0.0,8.303165,-7.820291,8.433181,7.664992,0.0,8.33698,-3.4 |
| 14 | 13 | 749,4.295691,3.164129,6.018792,3.9925,-0.408471,4.850911,3.518775,-3.481628,4.146976,2.386714,5.759656,3.845729,-1.577107,4.302408,3.35577,6.258 |
| 15 | 14 | 6,-12.571347,5.326943,2.739519,6.524393,4.807137,-5.438506,5.343608,4.02686,7.597405,5.008982,2.248025,5.958085,4.531538,-10.877847,5.332962,3.2 |
| 16 | 15 | 57086,-0.39394,-1.039489,-0.215672,-0.408308,-2.414914,-0.354923,-0.63202,0.643764,-0.397973,-1.333897,-0.341891,-0.465972,-3.929167,-0.381948,-0.8 |
| 17 | 16 | 5,6.255359,3.373521,-7.811375,5.79383,3.145967,9.383015,3.451143,-1.090889,5.936029,3.224008,-12.312942,5.682737,2.701204,6.543121,3.427993,-5.32 |
| 18 | 17 | ,9.604905,3.066248,2.42892,5.639599,2.872272,0.0,4.562129,2.722381,7.472968,3.027355,1.827893,4.7095,2.859335,10.802448,4.022083,2.720166,5.9929 |
| 19 | 18 | ,-1.971491,1.285285,-1.845827,-2.482186,-1.629228,-1.877349,0.0,-1.764094,-2.146122,-0.152967,-1.860037,-3.242683,-1.729567,-1.966324,1.86692,-1.811 |

## 5.5.27 Single Exponential Smoothing

Single Exponential Smoothing model is suitable to model the time series without trend and seasonality. In the model, the smoothed value is the weighted sum of previous smoothed value and previous observed value.

PAL provides two simple exponential smoothing algorithms: single exponential smoothing and adaptive-response-rate simple exponential smoothing. The adaptive-response-rate single exponential smoothing algorithm may have an advantage over single exponential smoothing in that it allows the value of alpha to be modified.

For single exponential smoothing, let $S_t$ be the smoothed value for the t-th time period. Mathematically:

$$S_1 = x_0$$

$$S_t = \alpha x_{t-1} + (1-\alpha)S_{t-1}$$

Where $\alpha \in (0,1)$ is a user specified parameter. Forecast is made by:

$$S_{T+m} = \alpha x_T + (1-\alpha)S_T$$

For adaptive-response-rate single exponential smoothing, let $S_t$ be the smoothed value for the t-th time period. Initialize for adaptive-response-rate single exponential smoothing as follows:

$$S_1 = x_0$$

$$\alpha 1 = \alpha 2 = \alpha 3 = \delta = 0.2$$

$$A_0 = M_0 = 0$$

Update the parameter of α as follows:

$$E_t = X_t - S_t$$

$$A_t = \delta E_t + (1 - \delta)A_{t-1}$$

$$M_t = \delta|E_t| + (1 - \delta)M_{t-1}$$

$$\alpha_t = \left|\frac{A_{t-1}}{M_{t-1}}\right| \quad (t \geq 4)$$

The calculation of the smoothed value as follows:

$$S_{t+1} = \alpha_t x_t + (1 - \alpha_t) S_t$$

Where $\alpha, \delta \in (0,1)$ is a user specified parameter, and $|\ \ |$ denotes absolute values.

It is worth nothing that when $t \geq T+2$, the smoothed value $S_t$, that is, the forecast value, is always $S_{T+1}$ ($x_{t-1}$ is not available and $S_{t-1}$ is used instead).

PAL calculates the prediction interval to get the idea of likely variation. Assume that the forecast data is normally distributed. The mean value is $S_t$ and the variance is $\sigma^2$. Let $U_t$ be the upper bound of prediction interval for $S_t$ and $L_t$ be the lower bound. Then they are calculated as follows:

$$U_t = S_t + z\sigma$$

$$L_t = S_t - z\sigma$$

Here z is the one-tailed value of a standard normal distribution. It is derived from the input parameters PREDICTION_CONFIDENCE_1 and PREDICTION_CONFIDENCE_2.

> **Note**
>
> The algorithm is backward compatible. You can still work in the SAP HANA Platform 1.0 SPS 11 or older versions where the prediction interval feature is not available. In that case, only point forecasts are calculated.

## Prerequisites

- No missing or null data in the inputs.
- The data is numeric, not categorical.

## _SYS_AFL.PAL_SINGLE_EXPSMOOTH

This is a single exponential smoothing function.

### Overview of all Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<FORECAST table>` | OUT |

| Position | Table Name | Parameter Type |
|---|---|---|
| 4 | `<STATISTICS table>` | OUT |

### Input Table

| Table | Column | Column Data Type | Description |
|---|---|---|---|
| DATA | 1st column | INTEGER | ID |
| | 2nd column | INTEGER, DOUBLE, or DECIMAL(p, s) | Raw data |

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| ALPHA | DOUBLE | 0.1 (Single exponential smoothing)<br>0.2 (Adaptive-response-rate single exponential smoothing) | The smoothing constant alpha for single exponential smoothing or the initialization value for adaptive-response-rate single exponential smoothing ($0 < \alpha < 1$). | |
| DELTA | DOUBLE | 0.2 | Value of weighted for $A_t$ and $M_t$. | Only valid when ADAPTIVE_METHOD is 1. |
| FORECAST_NUM | INTEGER | 0 | Number of values to be forecast. When it is set to 1, the algorithm only forecasts one value. | |
| ADAPTIVE_METHOD | INTEGER | 0 | • 0: Single exponential smoothing<br>• 1: Adaptive-response-rate single exponential smoothing | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| MEASURE_NAME | VARCHAR | No default value | Measure name. Supported measures are MPE, MSE, RMSE, ET, MAD, MASE, WMAPE, SMAPE, and MAPE.<br>For detailed information on measures, see *Forecast Accuracy Measures*. | |
| IGNORE_ZERO | INTEGER | 0 | • 0: Uses zero values in the input dataset when calculating MPE or MAPE.<br>• 1: Ignores zero values in the input dataset when calculating MPE or MAPE. | Only valid when MEASURE_NAME is MPE or MAPE. |
| EXPOST_FLAG | INTEGER | 1 | • 0: Does not output the expost forecast, and just outputs the forecast values.<br>• 1: Outputs the expost forecast and the forecast values. | |
| PREDICTION_CONFIDENCE_1 | DOUBLE | 0.8 | Prediction confidence for interval 1 | Only valid when the upper and lower columns are provided in the result table. |
| PREDICTION_CONFIDENCE_2 | DOUBLE | 0.95 | Prediction confidence for interval 2 | Only valid when the upper and lower columns are provided in the result table. |

## Output Tables

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| FORECAST | 1st column | INTEGER | TIMESTAMP | ID |

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| | 2nd column | DOUBLE | VALUE | Output result |
| | 3rd column | DOUBLE | PI1_LOWER | Lower bound of prediction interval 1 |
| | 4th column | DOUBLE | PI1_UPPER | Upper bound of prediction interval 1 |
| | 5th column | DOUBLE | PI2_LOWER | Lower bound of prediction interval 2 |
| | 6th column | DOUBLE | PI2_UPPER | Upper bound of prediction interval 2 |
| STATISTICS | 1st column | NVARCHAR(100) | STAT_NAME | Name of statistics |
| | 2nd column | DOUBLE | STAT_VALUE | Value of statistics |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA DM_PAL;
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME " VARCHAR(100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (100)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('ADAPTIVE_METHOD',0, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MEASURE_NAME', NULL, NULL, 'MSE');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('ALPHA', NULL,0.1, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FORECAST_NUM',12, NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('EXPOST_FLAG',1, NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PREDICTION_CONFIDENCE_1', NULL, 0.8,
NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PREDICTION_CONFIDENCE_2', NULL, 0.95,
NULL);
DROP TABLE PAL_SINGLESMOOTH_DATA_TBL;
CREATE COLUMN TABLE PAL_SINGLESMOOTH_DATA_TBL ("ID" INT, "RAWDATA" DOUBLE);
INSERT INTO PAL_SINGLESMOOTH_DATA_TBL VALUES (1,200.0);
INSERT INTO PAL_SINGLESMOOTH_DATA_TBL VALUES (2,135.0);
INSERT INTO PAL_SINGLESMOOTH_DATA_TBL VALUES (3,195.0);
INSERT INTO PAL_SINGLESMOOTH_DATA_TBL VALUES (4,197.5);
INSERT INTO PAL_SINGLESMOOTH_DATA_TBL VALUES (5,310.0);
INSERT INTO PAL_SINGLESMOOTH_DATA_TBL VALUES (6,175.0);
INSERT INTO PAL_SINGLESMOOTH_DATA_TBL VALUES (7,155.0);
INSERT INTO PAL_SINGLESMOOTH_DATA_TBL VALUES (8,130.0);
INSERT INTO PAL_SINGLESMOOTH_DATA_TBL VALUES (9,220.0);
INSERT INTO PAL_SINGLESMOOTH_DATA_TBL VALUES (10,277.5);
INSERT INTO PAL_SINGLESMOOTH_DATA_TBL VALUES (11,235.0);
CALL _SYS_AFL.PAL_SINGLE_EXPSMOOTH(PAL_SINGLESMOOTH_DATA_TBL,
#PAL_PARAMETER_TBL, ?,?);
```

**Expected Result**

FORECAST:

| | TIMESTAMP | VALUE | PI1_LOWER | PI1_UPPER | PI2_LOWER | PI2_UPPER |
|---|---|---|---|---|---|---|
| 1 | 2 | 200 | ? | ? | ? | ? |
| 2 | 3 | 193.5 | ? | ? | ? | ? |
| 3 | 4 | 193.65 | ? | ? | ? | ? |
| 4 | 5 | 194.035 | ? | ? | ? | ? |
| 5 | 6 | 205.6315 | ? | ? | ? | ? |
| 6 | 7 | 202.56834999999998 | ? | ? | ? | ? |
| 7 | 8 | 197.81151499999999 | ? | ? | ? | ? |
| 8 | 9 | 191.0303635 | ? | ? | ? | ? |
| 9 | 10 | 193.92732715 | ? | ? | ? | ? |
| 10 | 11 | 202.284594435 | ? | ? | ? | ? |
| 11 | 12 | 205.55613499150002 | 126.70094020329802 | 284.411329779702 | 84.95752161218711 | 326.1547483708129 |
| 12 | 13 | 205.55613499150002 | 126.30764502143099 | 284.80462496156906 | 84.35602853732597 | 326.75624144567405 |
| 13 | 14 | 205.55613499150002 | 125.91629207360505 | 285.19597790939497 | 83.75750585310105 | 327.35476412989897 |
| 14 | 15 | 205.55613499150002 | 125.5268528665313 | 285.58541711646876 | 83.16190998278738 | 327.95036000021264 |
| 15 | 16 | 205.55613499150002 | 125.13929959685962 | 285.97297038614045 | 82.56919840483035 | 328.5430715781697 |
| 16 | 17 | 205.55613499150002 | 124.75360512801404 | 286.358664854986 | 81.97932961741843 | 329.1329403655816 |
| 17 | 18 | 205.55613499150002 | 124.36974296801867 | 286.74252701498136 | 81.3922631045708 | 329.7200068784292 |
| 18 | 19 | 205.55613499150002 | 123.98768724826263 | 287.1245827347374 | 80.8079593036612 | 330.3043106793388 |
| 19 | 20 | 205.55613499150002 | 123.60741270315619 | 287.50485727984386 | 80.22637957430459 | 330.88589040869545 |
| 20 | 21 | 205.55613499150002 | 123.22889465063257 | 287.88337533236745 | 79.64748616853728 | 331.46478381446275 |
| 21 | 22 | 205.55613499150002 | 122.85210897345294 | 288.2601610095471 | 79.0712422022254 | 332.04102778077464 |
| 22 | 23 | 205.55613499150002 | 122.47703210127462 | 288.6352378817254 | 78.49761162764057 | 332.61465835535944 |

STATISTICS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | MSE | 3,438.3321253085405 |

> ⓘ Note
>
> The wrapper procedure generation approach to call PAL functions is still supported. For more information, refer to the older versions (prior to SAP HANA Platform 2.0 SPS 02) of this document.

## Related Information

Forecast Accuracy Measures [page 762]

# 5.5.28 Double Exponential Smoothing

Double Exponential Smoothing model is suitable to model the time series with trend but without seasonality. In the model there are two kinds of smoothed quantities: smoothed signal and smoothed trend.

PAL provides two methods of double exponential smoothing: Holt's linear exponential smoothing and additive damped trend Holt's linear exponential smoothing. The Holt's linear exponential smoothing displays a constant

trend indefinitely into the future. Empirical evidence shows that the Holt's linear method tends to over-forecast. A parameter that is used to damp the trend could improve the situation.

**Holt's Linear Exponential Smoothing**

Let $S_t$ and $b_t$ be the smoothed value and smoothed trend for the (t+1)-th time period, respectively. The following rules are satisfied:

$$S_0 = x_0$$

$$b_0 = x_1 - x_0$$

$$S_t = \alpha x_t + (1 - \alpha)(S_{t-1} + b_{t-1})$$

$$b_t = \beta (S_t - S_{t-1}) + (1 - \beta) b_{t-1}$$

Where $\alpha, \beta \in (0,1)$ are two user specified parameters. The model can be understood as two coupled Single Exponential Smoothing models, and forecast can be made by the following equation:

$$F_{T+m} = S_T + mb_T$$

**Additive Damped Trend Holt's Linear Exponential Smoothing**

Let $S_t$ and $b_t$ be the smoothed value and smoothed trend for the (t+1)-th time period, respectively. The following rules are satisfied:

$$S_0 = x_0$$

$$b_0 = x_1 - x_0$$

$$S_t = \alpha x_t + (1 - \alpha)(S_{t-1} + \Phi b_{t-1})$$

$$b_t = \beta (S_t - S_{t-1}) + (1 - \beta) \Phi b_{t-1}$$

Where $\alpha, \beta, \Phi \in (0,1)$ are three user specified parameters. Forecast can be made by the following equation:

$$F_{T+m} = S_T + (\Phi + \Phi^2 + \ldots + \Phi^m) b_T$$

> **Note**
>
> $F_0$ is not defined because there is no estimation for time 0. According to the definition, you can get $F_1 = S_0 + b_0$ and so on.

PAL calculates the prediction interval to get the idea of likely variation. Assume that the forecast data is normally distributed. The mean value is $S_t$ and the variance is $\sigma^2$. Let $U_t$ be the upper bound of prediction interval for $S_t$ and $L_t$ be the lower bound. Then they are calculated as follows:

$$U_t = S_t + z\sigma$$

$$L_t = S_t - z\sigma$$

Here z is the one-tailed value of a standard normal distribution. It is derived from the input parameters PREDICTION_CONFIDENCE_1 and PREDICTION_CONFIDENCE_2.

> **Note**
>
> The algorithm is backward compatible. You can still work in the SAP HANA Platform 1.0 SPS 11 or older versions where the prediction interval feature is not available. In that case, only point forecasts are calculated.

## Prerequisites

- No missing or null data in the inputs.
- The data is numeric, not categorical.

## _SYS_AFL.PAL_DOUBLE_EXPSMOOTH

This is a double exponential smoothing function.

### Overview of all Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<FORECAST table>` | OUT |
| 4 | `<STATISTICS table>` | OUT |

### Input Table

| Table | Column | Data Type | Description |
|---|---|---|---|
| DATA | 1st column | INTEGER | ID |
| | 2nd column | INTEGER, DOUBLE, or DECIMAL(p, s) | Raw data |

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| ALPHA | DOUBLE | 0.1 | Value of the smoothing constant alpha ($0 < \alpha < 1$). | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| BETA | DOUBLE | 0.1 | Value of the smoothing constant beta ($0 < \beta < 1$). | |
| FORECAST_NUM | INTEGER | 0 | Number of values to be forecast. | |
| PHI | DOUBLE | 0.1 | Value of the damped smoothing constant $\Phi$ ($0 < \Phi < 1$). | |
| DAMPED | INTEGER | 0 | • 0: Uses the Holt's linear method.<br>• 1: Uses the additive damped trend Holt's linear method. | |
| MEASURE_NAME | VARCHAR | No default value | Measure name. Supported measures are MPE, MSE, RMSE, ET, MAD, MASE, WMAPE, SMAPE, and MAPE.<br>For detailed information on measures, see *Forecast Accuracy Measures*. | |
| IGNORE_ZERO | INTEGER | 0 | • 0: Uses zero values in the input dataset when calculating MPE or MAPE.<br>• 1: Ignores zero values in the input dataset when calculating MPE or MAPE. | Only valid when MEASURE_NAME is MPE or MAPE. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| EXPOST_FLAG | INTEGER | 1 | • 0: Does not output the expost forecast, and just outputs the forecast values.<br>• 1: Outputs the expost forecast and the forecast values. | |
| PREDICTION_CONFIDENCE_1 | DOUBLE | 0.8 | Prediction confidence for interval 1 | Only valid when the upper and lower columns are provided in the result table. |
| PREDICTION_CONFIDENCE_2 | DOUBLE | 0.95 | Prediction confidence for interval 2 | Only valid when the upper and lower columns are provided in the result table. |

### Output Tables

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| FORECAST | 1st column | INTEGER | TIMESTAMP | ID |
| | 2nd column | DOUBLE | VALUE | Output result |
| | 3rd column | DOUBLE | PI1_LOWER | Lower bound of prediction interval 1 |
| | 4th column | DOUBLE | PI1_UPPER | Upper bound of prediction interval 1 |
| | 5th column | DOUBLE | PI2_LOWER | Lower bound of prediction interval 2 |
| | 6th column | DOUBLE | PI2_UPPER | Upper bound of prediction interval 2 |
| STATISTICS | 1st column | NVARCHAR(100) | STAT_NAME | Name of statistics |
| | 2nd column | DOUBLE | STAT_VALUE | Value of statistics |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;

- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA DM_PAL;
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME " VARCHAR(100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (100)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('ALPHA', NULL,0.501, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('BETA', NULL,0.072, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FORECAST_NUM',6, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('EXPOST_FLAG',1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MEASURE_NAME', NULL, NULL, 'MSE');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PREDICTION_CONFIDENCE_1', NULL, 0.8,
NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PREDICTION_CONFIDENCE_2', NULL, 0.95,
NULL);
DROP TABLE PAL_DOUBLESMOOTH_DATA_TBL;
CREATE COLUMN TABLE PAL_DOUBLESMOOTH_DATA_TBL ("ID" INT, "RAWDATA" DOUBLE);
INSERT INTO PAL_DOUBLESMOOTH_DATA_TBL VALUES (1,143.0);
INSERT INTO PAL_DOUBLESMOOTH_DATA_TBL VALUES (2,152.0);
INSERT INTO PAL_DOUBLESMOOTH_DATA_TBL VALUES (3,161.0);
INSERT INTO PAL_DOUBLESMOOTH_DATA_TBL VALUES (4,139.0);
INSERT INTO PAL_DOUBLESMOOTH_DATA_TBL VALUES (5,137.0);
INSERT INTO PAL_DOUBLESMOOTH_DATA_TBL VALUES (6,174.0);
INSERT INTO PAL_DOUBLESMOOTH_DATA_TBL VALUES (7,142.0);
INSERT INTO PAL_DOUBLESMOOTH_DATA_TBL VALUES (8,141.0);
INSERT INTO PAL_DOUBLESMOOTH_DATA_TBL VALUES (9,162.0);
INSERT INTO PAL_DOUBLESMOOTH_DATA_TBL VALUES (10,180.0);
INSERT INTO PAL_DOUBLESMOOTH_DATA_TBL VALUES (11,164.0);
INSERT INTO PAL_DOUBLESMOOTH_DATA_TBL VALUES (12,171.0);
INSERT INTO PAL_DOUBLESMOOTH_DATA_TBL VALUES (13,206.0);
INSERT INTO PAL_DOUBLESMOOTH_DATA_TBL VALUES (14,193.0);
INSERT INTO PAL_DOUBLESMOOTH_DATA_TBL VALUES (15,207.0);
INSERT INTO PAL_DOUBLESMOOTH_DATA_TBL VALUES (16,218.0);
INSERT INTO PAL_DOUBLESMOOTH_DATA_TBL VALUES (17,229.0);
INSERT INTO PAL_DOUBLESMOOTH_DATA_TBL VALUES (18,225.0);
INSERT INTO PAL_DOUBLESMOOTH_DATA_TBL VALUES (19,204.0);
INSERT INTO PAL_DOUBLESMOOTH_DATA_TBL VALUES (20,227.0);
INSERT INTO PAL_DOUBLESMOOTH_DATA_TBL VALUES (21,223.0);
INSERT INTO PAL_DOUBLESMOOTH_DATA_TBL VALUES (22,242.0);
INSERT INTO PAL_DOUBLESMOOTH_DATA_TBL VALUES (23,239.0);
INSERT INTO PAL_DOUBLESMOOTH_DATA_TBL VALUES (24,266.0);
CALL _SYS_AFL.PAL_DOUBLE_EXPSMOOTH(PAL_DOUBLESMOOTH_DATA_TBL,
#PAL_PARAMETER_TBL, ?,?);
```

### Expected Result

FORECAST:

| | TIMESTAMP | VALUE | PI1_LOWER | PI1_UPPER | PI2_LOWER | PI2_UPPER |
|---|---|---|---|---|---|---|
| 1 | 2 | 152 | ? | ? | ? | ? |
| 2 | 3 | 161 | ? | ? | ? | ? |
| 3 | 4 | 170 | ? | ? | ? | ? |
| 4 | 5 | 162.350768 | ? | ? | ? | ? |
| 5 | 6 | 156.617348328704 | ? | ? | ? | ? |
| 6 | 7 | 172.9203989238143 | ? | ? | ? | ? |
| 7 | 8 | 163.90826054079446 | ? | ? | ? | ? |
| 8 | 9 | 158.08385671344007 | ? | ? | ? | ? |
| 9 | 10 | 165.83974232422298 | ? | ? | ? | ? |
| 10 | 11 | 179.23871805888433 | ? | ? | ? | ? |
| 11 | 12 | 177.35911591266023 | ? | ? | ? | ? |
| 12 | 13 | 179.69880841249295 | ? | ? | ? | ? |
| 13 | 14 | 199.35005155285404 | ? | ? | ? | ? |
| 14 | 15 | 202.41396282027966 | ? | ? | ? | ? |
| 15 | 16 | 211.12228207587194 | ? | ? | ? | ? |
| 16 | 17 | 221.2268264253716 | ? | ? | ? | ? |
| 17 | 18 | 232.06038797295594 | ? | ? | ? | ? |
| 18 | 19 | 235.20765287024005 | ? | ? | ? | ? |
| 19 | 20 | 225.13141559964953 | ? | ? | ? | ? |
| 20 | 21 | 231.69377677811428 | ? | ? | ? | ? |
| 21 | 22 | 232.6507930902281 | ? | ? | ? | ? |
| 22 | 23 | 242.98458882162217 | ? | ? | ? | ? |
| 23 | 24 | 246.49442080361422 | ? | ? | ? | ? |
| 24 | 25 | 262.4764322154003 | 240.7230516016306 | 284.22981282917 | 229.20750750626294 | 295.7453569245376 |
| 25 | 26 | 268.6861484497971 | 243.9939405112287 | 293.3783563883655 | 230.92267520414418 | 306.44962169545 |
| 26 | 27 | 274.8958646841939 | 247.23448984628158 | 302.5572395221062 | 232.59144252687585 | 317.20028684151197 |
| 27 | 28 | 281.1055809185907 | 250.43343372285102 | 311.7777281143304 | 234.19657979180866 | 328.0145820453728 |
| 28 | 29 | 287.3152971529875 | 253.58365814380176 | 321.04693616217327 | 235.7272070792061 | 338.9033872267689 |
| 29 | 30 | 293.5250133873843 | 256.68058778168626 | 330.3694389930824 | 237.17632702957397 | 349.8736997451947 |

STATISTICS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | MSE | 274.8960228191495 |

> **Note**
>
> The wrapper procedure generation approach to call PAL functions is still supported. For more information, refer to the older versions (prior to SAP HANA Platform 2.0 SPS 02) of this document.

## Related Information

Forecast Accuracy Measures [page 762]

## 5.5.29 Triple Exponential Smoothing

Triple exponential smoothing is used to handle the time series data containing a seasonal component.

This method is based on three smoothing equations: stationary component, trend, and seasonal. Both seasonal and trend can be additive or multiplicative. PAL supports multiplicative triple exponential smoothing and additive triple exponential smoothing. For additive triple exponential smoothing, an additive damped method is also supported.

Multiplicative triple exponential smoothing is given by the below formula:

$$S_t = \alpha \times \frac{X_t}{C_{t-L}} + (1 - \alpha) \times (S_{t-1} + B_{t-1})$$

$$B_t = \beta \times (S_t - S_{t-1}) + (1 - \beta) \times B_{t-1}$$

$$C_t = \gamma \times \frac{X_t}{S_t} + (1 - \gamma) \times C_{t-L}$$

$$F_{t+m} = (S_t + m \times B_t) \times C_{t-L+1+((m-1) \bmod L)}$$

Additive triple exponential smoothing is given by the below formula:

$$S_t = \alpha \times (X_t - C_{t-L}) + (1 - \alpha) \times (S_{t-1} + B_{t-1})$$

$$B_t = \beta \times (S_t - S_{t-1}) + (1 - \beta) \times B_{t-1}$$

$$C_t = \gamma \times (X_t - S_t) + (1 - \gamma) \times C_{t-L}$$

$$F_{t+m} = S_t + m \times B_t + C_{t-L+1+((m-1) \bmod L)}$$

The additive damped method of additive triple exponential smoothing is given by the below formula:

$$S_t = \alpha \times (X_t - C_{t-L}) + (1 - \alpha) \times (S_{t-1} + \Phi \times B_{t-1})$$

$$B_t = \beta \times (S_t - S_{t-1}) + (1 - \beta) \times \Phi \times B_{t-1}$$

$$C_t = \gamma \times (X_t - S_t) + (1 - \gamma) \times C_{t-L}$$

$$F_{t+m} = S_t + (\Phi + \Phi^2 + \ldots + \Phi^m) \times B_t + C_{t-L+1+((m-1) \bmod L)}$$

Where:

| | |
|---|---|
| $\alpha$ | Data smoothing factor. The range is $0 < \alpha < 1$. |
| $\beta$ | Trend smoothing factor. The range is $0 \le \beta < 1$. |
| $\gamma$ | Seasonal change smoothing factor. The range is $0 < \gamma < 1$. |
| $\Phi$ | Damped smoothing factor. The range is $0 < \Phi < 1$. |
| X | Observation |
| S | Smoothed observation |

| | |
|---|---|
| B | Trend factor |
| C | Seasonal index |
| F | The forecast at m periods ahead |
| t | The index that denotes a time period |

> **Note**
>
> α, β, and γ are the constants given by the user and they usually should be tuned based on historic data. Please refer to Auto Exponential Smoothing function in PAL to optimize the exponential smoothing parameters.

PAL uses two methods for initialization. The first is the following formula:

To initialize trend estimate $B_{L-1}$:

$$B_{L-1} = \frac{1}{L} \times \left(\sum_{i=0}^{L-1} \frac{X_{L+i} - X_i}{L}\right)$$

To initialize the seasonal indices $C_i$ for $i = 0,1,...,L-1$ for multiplicative triple exponential smoothing:

$$C_i = \frac{X_i}{S_{L-1}} \quad 0 \le i \le L-1$$

To initialize the seasonal indices $C_i$ for $i = 0,1,...,L-1$ for additive triple exponential smoothing:

$c_i = x_i - S_{L-1}$ $0 \le i \le L-1$

Where

$$S_{L-1} = \frac{1}{L} \times \left(\sum_{i=0}^{L-1} X_i\right)$$

> **Note**
>
> $S_{L-1}$ is the average value of x in the L cycle of your data.

The second initialization method is as follows:

1. Get the trend component by using moving averages with the first two CYCLE observations.
2. The seasonal component is computed by removing the trend component from the observations. For additive, use the observations minus the trend component. For multiplicative, use the observations divide the trend component.
3. The start values of $C_t$ are initialized by using the seasonal component calculated in Step 2. The start values of $S_t$ and $B_t$ are initialized by using a simple linear regressing on the trend component, $S_t$ is initialized by intercept and $B_t$ is initialized by slope.

PAL calculates the prediction interval to get the idea of likely variation. Assume that the forecast data is normally distributed. The mean value is $S_t$ and the variance is $\sigma^2$. Let $U_t$ be the upper bound of prediction interval for $S_t$ and $L_t$ be the lower bound. Then they are calculated as follows:

$$U_t = S_t + z\sigma$$

$$L_t = S_t - z\sigma$$

Here z is the one-tailed value of a standard normal distribution. It is derived from the input parameters PREDICTION_CONFIDENCE_1 and PREDICTION_CONFIDENCE_2.

This function cannot be used on a multiplicative model when the data contains negative values. If the data contains positive and negative data, the multiplicative seasonal indices for the negative data are the inverse of what they are for the positive data. This causes the model to fit the data incorrectly.

> **Note**
>
> The algorithm is backward compatible. You can still work in the SAP HANA Platform 1.0 SPS 11 or older versions where the prediction interval feature is not available. In that case, only point forecasts are calculated.

## Prerequisites

- No missing or null data in the inputs.
- The data is numeric, not categorical.
- The number of valid historical data should be at least twice the value of the CYCLE parameter.

## _SYS_AFL.PAL_TRIPLE_EXPSMOOTH

This is a triple exponential smoothing function.

### Overview of all Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<FORECAST table>` | OUT |
| 4 | `<STATISTICS table>` | OUT |

### Input Table

| Table | Column | Data Type | Description |
|---|---|---|---|
| DATA | 1st column | INTEGER | ID |
| | 2nd column | INTEGER, DOUBLE, or DECIMAL(p, s) | Raw data |

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| ALPHA | DOUBLE | 0.1 | Value of the smoothing constant alpha ($0 < \alpha < 1$). | |
| BETA | DOUBLE | 0.1 | Value of the smoothing constant beta ($0 \leq \beta < 1$). | |
| GAMMA | DOUBLE | 0.1 | Value of the smoothing constant gamma ( $0 < \gamma < 1$). | |
| CYCLE | INTEGER | 2 | A cycle of length L ($L > 1$). For example, quarterly data cycle is 4, and monthly data cycle is 12. | |
| FORECAST_NUM | INTEGER | 0 | Number of values to be forecast. | |
| SEASONAL | INTEGER | 0 | • 0: Multiplicative triple exponential smoothing<br>• 1: Additive triple exponential smoothing | When SEASONAL is set to 1, the default value of INITIAL_METHOD is 1; When SEASONAL is set to 0, the default value of INITIAL_METHOD is 0. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| INITIAL_METHOD | INTEGER | 0 or 1 | Initialization method for trend and seasonal<br>• 0: The first initialization method as the description part.<br>• 1: The second initialization method as the description part. | When SEASONAL is set to 1, the default value of INITIAL_METHOD is 1; When SEASONAL is set to 0, the default value of INITIAL_METHOD is 0. |
| PHI | DOUBLE | 0.1 | Value of the damped smoothing constant Φ ($0 < \Phi < 1$). | |
| DAMPED | INTEGER | 0 | • 0: Uses the Holt's linear method.<br>• 1: Uses the additive damped trend Holt's linear method. | |
| MEASURE_NAME | VARCHAR | No default value | Measure name. Supported measures are MPE, MSE, RMSE, ET, MAD, MASE, WMAPE, SMAPE, and MAPE.<br>For detailed information on measures, see *Forecast Accuracy Measures*. | |
| IGNORE_ZERO | INTEGER | 0 | • 0: Uses zero values in the input dataset when calculating MPE or MAPE.<br>• 1: Ignores zero values in the input dataset when calculating MPE or MAPE. | Only valid when MEASURE_NAME is MPE or MAPE. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| EXPOST_FLAG | INTEGER | 1 | • 0: Does not output the expost forecast, and just outputs the forecast values.<br>• 1: Outputs the expost forecast and the forecast values. | |
| LEVEL_START | DOUBLE | No default value | The initial value for level component S. If this value is not provided, it will be calculated in the way as described earlier.<br>**Note**: LEVEL_START cannot be zero. If it is set to zero, 0.0000000001 will be used instead. | |
| TREND_START | DOUBLE | No default value | The initial value for trend component B. If this value is not provided, it will be calculated in the way as described earlier. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| SEASON_START | INTEGER or DOUBLE | No default value | A list of initial values for seasonal component **C**. If this parameter is not used, the initial values of **C** will be calculated in the way as described earlier.<br><br>Two values must be provided for each cycle:<br><br>• Cycle ID: An INTEGER which represents which cycle the initial value is used for.<br>• Initial value: A DOUBLE precision number which represents the initial value for the corresponding cycle.<br><br>For example: To give the initial value 0.5 to the 3rd cycle, insert tuple `('SEASON_START', 3, 0.5, NULL)` into the parameter table.<br><br>**Note**: The initial values of all cycles must be provided if this parameter is used. | |
| PREDICTION_CONFIDENCE_1 | DOUBLE | 0.8 | Prediction confidence for interval 1 | Only valid when the upper and lower columns are provided in the result table. |
| PREDICTION_CONFIDENCE_2 | DOUBLE | 0.95 | Prediction confidence for interval 2 | Only valid when the upper and lower columns are provided in the result table. |

> **Note**
>
> Cycle determines the seasonality within the time series data by considering the seasonal factor of a data point$_{t\text{-}CYCLE+1}$ in the forecast calculation of data point$_{t+1}$. Additionally, the algorithm of TESM takes an entire CYCLE as the base to calculate the first forecasted value for data point$_{CYCLE+1}$. The value for CYCLE should be within the range of 2 ≤ CYCLE ≤ entire number of data point/2.
>
> For example, there is one year of weekly data (52 data points) as input time series. The value for CYCLE should range within 2 ≤ CYCLE ≤ 26. If CYCLE is 4, we get the first forecast value for data point 5 (e.g. week 201205) which considers the seasonal factor of data point 1 (e.g. week 201201). The second forecast value for data point 6 (e.g. week 201206) considers the seasonal factor of data point 2 (e.g. week 201202), etc. If CYCLE is 2, then we get the first forecast value for data point 3 (e.g. week 201203) which considers the seasonal factor of data point 1 (e.g. week 201201). The second forecast value for data point 4 (e.g. week 201204) considers the seasonal factor of data point 2 (e.g. week 201202), etc.

### Output Tables

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| FORECAST | 1st column | INTEGER | TIMESTAMP | ID |
| | 2nd column | DOUBLE | VALUE | Output result |
| | 3rd column | DOUBLE | PI1_LOWER | Lower bound of prediction interval 1 |
| | 4th column | DOUBLE | PI1_UPPER | Upper bound of prediction interval 1 |
| | 5th column | DOUBLE | PI2_LOWER | Lower bound of prediction interval 2 |
| | 6th column | DOUBLE | PI2_UPPER | Upper bound of prediction interval 2 |
| STATISTICS | 1st column | NVARCHAR(100) | STAT_NAME | Name of statistics |
| | 2nd column | DOUBLE | STAT_VALUE | Value of statistics |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA DM_PAL;
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME " VARCHAR(100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
```

```
    "STRING_VALUE" VARCHAR (100)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('ALPHA', NULL, 0.822, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('BETA', NULL, 0.055, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('GAMMA', NULL, 0.055, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('CYCLE', 4, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FORECAST_NUM', 6, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SEASONAL', 0, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('INITIAL_METHOD', 0, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MEASURE_NAME', NULL, NULL, 'MSE');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('EXPOST_FLAG', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PREDICTION_CONFIDENCE_1', NULL, 0.8,
NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PREDICTION_CONFIDENCE_2', NULL, 0.95,
NULL);
DROP TABLE PAL_TRIPLESMOOTH_DATA_TBL;
CREATE COLUMN TABLE PAL_TRIPLESMOOTH_DATA_TBL ("ID" INT, "RAWDATA" DOUBLE);
INSERT INTO PAL_TRIPLESMOOTH_DATA_TBL VALUES (1,362.0);
INSERT INTO PAL_TRIPLESMOOTH_DATA_TBL VALUES (2,385.0);
INSERT INTO PAL_TRIPLESMOOTH_DATA_TBL VALUES (3,432.0);
INSERT INTO PAL_TRIPLESMOOTH_DATA_TBL VALUES (4,341.0);
INSERT INTO PAL_TRIPLESMOOTH_DATA_TBL VALUES (5,382.0);
INSERT INTO PAL_TRIPLESMOOTH_DATA_TBL VALUES (6,409.0);
INSERT INTO PAL_TRIPLESMOOTH_DATA_TBL VALUES (7,498.0);
INSERT INTO PAL_TRIPLESMOOTH_DATA_TBL VALUES (8,387.0);
INSERT INTO PAL_TRIPLESMOOTH_DATA_TBL VALUES (9,473.0);
INSERT INTO PAL_TRIPLESMOOTH_DATA_TBL VALUES (10,513.0);
INSERT INTO PAL_TRIPLESMOOTH_DATA_TBL VALUES (11,582.0);
INSERT INTO PAL_TRIPLESMOOTH_DATA_TBL VALUES (12,474.0);
INSERT INTO PAL_TRIPLESMOOTH_DATA_TBL VALUES (13,544.0);
INSERT INTO PAL_TRIPLESMOOTH_DATA_TBL VALUES (14,582.0);
INSERT INTO PAL_TRIPLESMOOTH_DATA_TBL VALUES (15,681.0);
INSERT INTO PAL_TRIPLESMOOTH_DATA_TBL VALUES (16,557.0);
INSERT INTO PAL_TRIPLESMOOTH_DATA_TBL VALUES (17,628.0);
INSERT INTO PAL_TRIPLESMOOTH_DATA_TBL VALUES (18,707.0);
INSERT INTO PAL_TRIPLESMOOTH_DATA_TBL VALUES (19,773.0);
INSERT INTO PAL_TRIPLESMOOTH_DATA_TBL VALUES (20,592.0);
INSERT INTO PAL_TRIPLESMOOTH_DATA_TBL VALUES (21,627.0);
INSERT INTO PAL_TRIPLESMOOTH_DATA_TBL VALUES (22,725.0);
INSERT INTO PAL_TRIPLESMOOTH_DATA_TBL VALUES (23,854.0);
INSERT INTO PAL_TRIPLESMOOTH_DATA_TBL VALUES (24,661.0);
CALL _SYS_AFL.PAL_TRIPLE_EXPSMOOTH(PAL_TRIPLESMOOTH_DATA_TBL,
#PAL_PARAMETER_TBL, ?,?);
```

### Expected Result

FORECAST:

| | TIMESTAMP | VALUE | PI1_LOWER | PI1_UPPER | PI2_LOWER | PI2_UPPER |
|---|---|---|---|---|---|---|
| 1 | 5 | 371.28815789473686 | ? | ? | ? | ? |
| 2 | 6 | 414.63620709090577 | ? | ? | ? | ? |
| 3 | 7 | 471.4318077577633 | ? | ? | ? | ? |
| 4 | 8 | 399.2922358985797 | ? | ? | ? | ? |
| 5 | 9 | 423.22175605383677 | ? | ? | ? | ? |
| 6 | 10 | 506.39903682061964 | ? | ? | ? | ? |
| 7 | 11 | 589.5887382682944 | ? | ? | ? | ? |
| 8 | 12 | 471.56575268616933 | ? | ? | ? | ? |
| 9 | 13 | 515.8787903699542 | ? | ? | ? | ? |
| 10 | 14 | 586.9176998062271 | ? | ? | ? | ? |
| 11 | 15 | 670.2603541868965 | ? | ? | ? | ? |
| 12 | 16 | 548.6608417948656 | ? | ? | ? | ? |
| 13 | 17 | 605.1661021997458 | ? | ? | ? | ? |
| 14 | 18 | 678.4727594613225 | ? | ? | ? | ? |
| 15 | 19 | 807.7263193081459 | ? | ? | ? | ? |
| 16 | 20 | 628.86590524558 | ? | ? | ? | ? |
| 17 | 21 | 650.2559751722578 | ? | ? | ? | ? |
| 18 | 22 | 683.0454322452091 | ? | ? | ? | ? |
| 19 | 23 | 821.7673777439587 | ? | ? | ? | ? |
| 20 | 24 | 683.4132407295845 | ? | ? | ? | ? |
| 21 | 25 | 721.983642459322 | 695.6478636204438 | 748.3194212982003 | 681.7065444347268 | 762.2607404839173 |
| 22 | 26 | 782.2537878644209 | 742.9512468990112 | 821.5563288298307 | 722.1457384779116 | 842.3618372509303 |
| 23 | 27 | 894.3794445625507 | 841.2064621236985 | 947.5524270014029 | 813.058385517655 | 975.7005036074464 |
| 24 | 28 | 718.2726402707948 | 667.0367904396924 | 769.5084901018972 | 639.9141698930622 | 796.6311106485274 |
| 25 | 29 | 778.9447542551759 | 715.8740302169056 | 842.0154782934462 | 682.4864057053512 | 875.4031028050006 |
| 26 | 30 | 842.7761971859832 | 767.847136474087 | 917.7052578978793 | 728.1820877181183 | 957.370306653848 |

STATISTICS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | MSE | 616.5415419997985 |

> **Note**
>
> The wrapper procedure generation approach to call PAL functions is still supported. For more information, refer to the older versions (prior to SAP HANA Platform 2.0 SPS 02) of this document.

## Related Information

Forecast Accuracy Measures [page 762]
Auto Exponential Smoothing [page 651]

## 5.5.30 Seasonality Test

The algorithm tests whether a time series has a seasonality or not. If it does, the corresponding additive or multiplicative seasonality model is identified, and the series is decomposed into three components: seasonal, trend, and random.

Typically, there are two decomposition models for time series $x_t$, $t=1,2,...,n$:

1. Additive: $x_t = m_t + s_t + e_t$
2. Multiplicative: $x_t = m_t \times s_t \times e_t$

Where $m_t$, $s_t$, and $e_t$ are trend, seasonality, and random components, respectively. They satisfy properties,

$$s_{t+d} = s_t, \sum_{j=1}^{d} s_j = 0, E(e_t) = 0$$

Where d is the length of seasonality cycle, that is, the period. It is believed that the additive model is useful when the seasonal variation is relatively constant over time, whereas the multiplicative model is useful when the seasonal variation increases over time.

Autocorrelation is employed to identify the seasonality here. The autocorrelation coefficient at lag h is given by

$$\gamma_h = c_h/c_0$$

Where $c_h$ is the autocovariance function

$$c_h = \frac{1}{n}\sum_{t=1}^{n-h}(x_t - \bar{x})(x_{t+h} - \bar{x})$$

The resulting $\gamma_h$ has a value in the range of -1 to 1, and larger value indicates more relevance.

For an n-elements time series, the probable period would be from 2 to n/2. The main procedure to determine the seasonality, therefore, is to calculate the autocorrelation coefficients of all possible lags ($d=2,3,...,n/2$) in the case of both additive and multiplicative models. A user-defined criterion for the coefficient is given, for example, 0.5, indicating that only if the autocorrelation coefficient is greater than the criterion is the tested seasonality considered. If there is no lag satisfying such a requirement, the time series is regarded to be of no seasonality. Otherwise, the one possessing the largest autocorrelation is the optimal seasonality.

Practically, the autocorrelation coefficient at lag of d is calculated as follows:

1. Estimate the trend $\hat{m}_t$ ($t = q+1, ..., n-q$). Here moving average is applied to estimate the trend:

$$\hat{m}_t = \begin{cases} \frac{1}{2q+1}\sum_{j=-q}^{q} x_{t-j}, & d = 2q+1 \\ \frac{1}{2q}(0.5x_{t-q} + \sum_{j=-q+1}^{q-1} x_{t-j} + 0.5x_{t+q}), & d = 2q \end{cases}$$

2. De-trend the time series. For an additive model, this is done by subtracting the estimated trend from the series. For a multiplicative decomposition, likewise, this is done by dividing the series by the trend. And then two sets (additive and multiplicative) of autocorrelation coefficients are calculated with the de-trended series.

The seasonal component, correspondingly, can be estimated as:

For an additive model,

$$\hat{s}_k = w_k - \frac{1}{d}\sum_{i=1}^{d} w_i,\ k = 1, 2, \ldots, d$$

Where $w_k = \frac{1}{(n-2q)/d}\sum_j (x_{k+jd} - \hat{m}_{k+jd}),\ q < k + jd \le n - q$.

Similarly, for a multiplicative model,

$$\hat{s}_k = w_k / (\frac{1}{d}\sum_{i=1}^{d} w_i),\ k = 1, 2, \ldots, d$$

Where $w_k = \frac{1}{(n-2q)/d}\sum_j (x_{k+jd}/\hat{m}_{k+jd}),\ q < k + jd \le n - q$.

Consequently, the random component can be estimated straightforwardly.

Note that during the course of calculating the moving average, the element at `t` is determined by series from $x_{t-q}$ to $x_{t+q}$. As a result, the trend and random series are valid only within the time range between q and n−q. Should there be no seasonality, a linear regression is carried out to decompose the series into only two components, trend and random, both of which have n values.

## Prerequisites

- No null data in the inputs. The time periods should be unique (no duplication) and equal sampling.
- The length of time series must be at least 1.

## _SYS_AFL.PAL_SEASONALITY_TEST

This function identifies the seasonality of a time series and decomposes it into three components: seasonal, trend, and random.

### Overview of all Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |

| Position | Table Name | Parameter Type |
|---|---|---|
| 2 | `<PARAMETER table>` | IN |
| 3 | `<STATISTICS table>` | OUT |
| 4 | `<DECOMPOSE table>` | OUT |

### Input Table

| Table | Column | Reference for Output Table | Data Type | Description |
|---|---|---|---|---|
| DATA | 1st column | ID | INTEGER | Time stamp.<br>Time stamp does not need to be in order, but must be unique and equal sampling. |
| | 2nd column | | INTEGER, DOUBLE, or DECIMA(p, s) | Time series. |

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| ALPHA | DOUBLE | 0.2 | The criterion for the autocorrelation coefficient. The value range is (0, 1). A larger value indicates stricter requirement for seasonality. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| PERIODS | INTEGER | No default value | Length of the periods. When this parameter is not specified, the algorithm will search the seasonality cycle. When this parameter is specified between 2 and half of the series length, autocorrelation value is calculated for this number of periods and the result is compared to ALPHA parameter. If correlation value is equal to or higher than ALPHA, decomposition is executed with the value of PERIODS. Otherwise, decomposition is executed with no seasonality. For other value of PERIODS, decomposition is also executed with no seasonality. | |
| THREAD_RATIO | DOUBLE | -1 | The ratio of available threads.<br>• 0: single thread<br>• 0~1: percentage<br>• Others: heuristically determined | |
| EXTRAPOLATION | INTEGER | 0 | Specifies whether to extrapolate the end-points.<br>• 0: No<br>• 1: Yes | Set to 1 when there's an end-point issue. |
| SMOOTH_WIDTH | INTEGER | 0 | Specifies the width of the moving average applied to non-seasonal data.<br>0 indicates linear fitting to extract trends. | Can not be larger than half of the data length . |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| AUXILIARY_NORMALITYTEST | AUXILIARY_NORMALITYTEST | 0 | Specifies whether to use normality test to identify model types.<br>• 0: No<br>• 1: Yes | |

### Output Tables

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| STATISTICS | 1st column | NVARCHAR(100) | STAT_NAME | Name |
| | 2nd column | NVARCHAR(100) | STAT_VALUE | Value |
| DECOMPOSED | 1st column | ID (in input table) data type | ID (in input table) column name | Time stamp that is monotonically increasing sorted. |
| | 2nd column | DOUBLE | SEASONAL | The seasonal component.<br>NULLs if no seasonal; otherwise repeated periodically. |
| | 3rd column | DOUBLE | TREND | The trend component. |
| | 4th column | DOUBLE | RANDOM | The random component. |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_SEASONALITY_DATA_TBL;
CREATE COLUMN TABLE PAL_SEASONALITY_DATA_TBL (
    "TIME_STAMP" INTEGER,
    "SERIES" DOUBLE
);
INSERT INTO PAL_SEASONALITY_DATA_TBL VALUES (1, 10);
INSERT INTO PAL_SEASONALITY_DATA_TBL VALUES (2, 7);
INSERT INTO PAL_SEASONALITY_DATA_TBL VALUES (3, 17);
INSERT INTO PAL_SEASONALITY_DATA_TBL VALUES (4, 34);
INSERT INTO PAL_SEASONALITY_DATA_TBL VALUES (5, 9);
INSERT INTO PAL_SEASONALITY_DATA_TBL VALUES (6, 7);
```

```
INSERT INTO PAL_SEASONALITY_DATA_TBL VALUES (7, 18);
INSERT INTO PAL_SEASONALITY_DATA_TBL VALUES (8, 40);
INSERT INTO PAL_SEASONALITY_DATA_TBL VALUES (9, 27);
INSERT INTO PAL_SEASONALITY_DATA_TBL VALUES (10, 7);
INSERT INTO PAL_SEASONALITY_DATA_TBL VALUES (11, 27);
INSERT INTO PAL_SEASONALITY_DATA_TBL VALUES (12, 100);
INSERT INTO PAL_SEASONALITY_DATA_TBL VALUES (13, 93);
INSERT INTO PAL_SEASONALITY_DATA_TBL VALUES (14, 29);
INSERT INTO PAL_SEASONALITY_DATA_TBL VALUES (15, 159);
INSERT INTO PAL_SEASONALITY_DATA_TBL VALUES (16, 614);
INSERT INTO PAL_SEASONALITY_DATA_TBL VALUES (17, 548);
INSERT INTO PAL_SEASONALITY_DATA_TBL VALUES (18, 102);
INSERT INTO PAL_SEASONALITY_DATA_TBL VALUES (19, 21);
INSERT INTO PAL_SEASONALITY_DATA_TBL VALUES (20, 238);
INSERT INTO PAL_SEASONALITY_DATA_TBL VALUES (21, 89);
INSERT INTO PAL_SEASONALITY_DATA_TBL VALUES (22, 292);
INSERT INTO PAL_SEASONALITY_DATA_TBL VALUES (23, 446);
INSERT INTO PAL_SEASONALITY_DATA_TBL VALUES (24, 689);
INSERT INTO PAL_SEASONALITY_DATA_TBL VALUES (25, 521);
INSERT INTO PAL_SEASONALITY_DATA_TBL VALUES (26, 155);
INSERT INTO PAL_SEASONALITY_DATA_TBL VALUES (27, 968);
INSERT INTO PAL_SEASONALITY_DATA_TBL VALUES (28, 1456);
INSERT INTO PAL_SEASONALITY_DATA_TBL VALUES (29, 936);
INSERT INTO PAL_SEASONALITY_DATA_TBL VALUES (30, 10);
INSERT INTO PAL_SEASONALITY_DATA_TBL VALUES (31, 83);
INSERT INTO PAL_SEASONALITY_DATA_TBL VALUES (32, 55);
INSERT INTO PAL_SEASONALITY_DATA_TBL VALUES (33, 207);
INSERT INTO PAL_SEASONALITY_DATA_TBL VALUES (34, 25);
INSERT INTO PAL_SEASONALITY_DATA_TBL VALUES (35, 0);
INSERT INTO PAL_SEASONALITY_DATA_TBL VALUES (36, 0);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR(100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR(100)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 0.5, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('ALPHA', NULL, 0.2, NULL);
CALL _SYS_AFL.PAL_SEASONALITY_TEST(PAL_SEASONALITY_DATA_TBL,
"#PAL_PARAMETER_TBL", ?, ?);
```

### Expected Result

STATISTICS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | type | multiplicative |
| 2 | period | 4 |
| 3 | acf | 0.501947 |

DECOMPOSED:

| | TIME_STAMP | SEASONAL | TREND | RANDOM |
|---|---|---|---|---|
| 1 | 1 | 1.2526604705702862 | 10.125 | 0.7884453482738342 |
| 2 | 2 | 0.34995164693440334 | 14 | 1.428768815292138 |
| 3 | 3 | 0.7488508742402852 | 16.875 | 1.3452710573776516 |
| 4 | 4 | 1.6485370082550252 | 16.75 | 1.2313043238363521 |
| 5 | 5 | 1.2526604705702862 | 16.875 | 0.4257604880678705 |
| 6 | 6 | 0.34995164693440334 | 17.75 | 1.1269162486811228 |
| 7 | 7 | 0.7488508742402852 | 20.75 | 1.1584013711650227 |
| 8 | 8 | 1.6485370082550252 | 23 | 1.0549538324429104 |
| 9 | 9 | 1.2526604705702862 | 24.125 | 0.8934352210750649 |
| 10 | 10 | 0.34995164693440334 | 32.75 | 0.6107714019569445 |
| 11 | 11 | 0.7488508742402852 | 48.5 | 0.7434070655156976 |
| 12 | 12 | 1.6485370082550252 | 59.5 | 1.0194931994196192 |
| 13 | 13 | 1.2526604705702862 | 78.75 | 0.9427553664359989 |
| 14 | 14 | 0.34995164693440334 | 159.5 | 0.5195522964698683 |
| 15 | 15 | 0.7488508742402852 | 280.6... | 0.7566158324803169 |
| 16 | 16 | 1.6485370082550252 | 346.6... | 1.0745083318974957 |
| 17 | 17 | 1.2526604705702862 | 338.5 | 1.2923748936181296 |
| 18 | 18 | 0.34995164693440334 | 274.25 | 1.0627851898617908 |
| 19 | 19 | 0.7488508742402852 | 169.8... | 0.16508000898868... |
| 20 | 20 | 1.6485370082550252 | 136.25 | 1.05959950069587 |
| 21 | 21 | 1.2526604705702862 | 213.1... | 0.33336671646369... |
| 22 | 22 | 0.34995164693440334 | 322.6... | 2.586287448680693 |
| 23 | 23 | 0.7488508742402852 | 433 | 1.3754715793490764 |
| 24 | 24 | 1.6485370082550252 | 469.8... | 0.8894840852738921 |
| 25 | 25 | 1.2526604705702862 | 518 | 0.8029242795005811 |
| 26 | 26 | 0.34995164693440334 | 679.1... | 0.6521897040170259 |
| 27 | 27 | 0.7488508742402852 | 826.8... | 1.5632921771207287 |
| 28 | 28 | 1.6485370082550252 | 860.6... | 1.0262394754059023 |
| 29 | 29 | 1.2526604705702862 | 731.8... | 1.0209525623352522 |
| 30 | 30 | 0.34995164693440334 | 446.1... | 0.06405239855610... |
| 31 | 31 | 0.7488508742402852 | 179.8... | 0.6161861667790781 |
| 32 | 32 | 1.6485370082550252 | 90.625 | 0.36814250980421 |

ⓘ Note

The wrapper procedure generation approach to call PAL functions is still supported. For more information, refer to the older versions (prior to SAP HANA Platform 2.0 SPS 02) of this document.

## 5.5.31 Stationarity Test

Stationarity means that a time series has a constant mean and constant variance over time. For many time series models, the input data has to be stationary for reasonable analysis.

This fuction is used to identify whether a time series is stationary by two optional statistical tests: Kwiatkowski-Phillips-Schmidt-Shin (KPSS) test and Augmented Dickey Fuller (ADF) test. Other than the level-stationarity, this function can also detect the trend-stationarity which denotes the time series fluctuates around a deterministic trend.

### Prerequisites

- No null data in the inputs. The time periods should be unique (no duplication) and equal sampling.
- The length of time series must be at least 1.

### _SYS_AFL.PAL_STATIONARITY_TEST

#### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<STATS table>` | OUT |

#### Input Table

| Table | Column | Data Type | Description | Dependency |
|---|---|---|---|---|
| DATA | 1st column | INTEGER | Time stamp. Time stamp does not need to be in order, but must be unique and equal sampling | |
| | 2nd column | INTEGER, DOUBLE, or DECIMAL(p,s) | Time series value | |

#### Parameter Table

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| METHOD | INTEGER | 0 | Statistic test that used to determine stationarity.<br>• 0: KPSS<br>• 1: ADF | |
| MODE | INTEGER | 0 | Type of stationarity to determine.<br>• 0: LEVEL<br>• 1: TREND<br>• 2: NONE | The last option (2: NONE) is not applicable to KPSS. |
| LAG | INTEGER | • KPSS: int(12*(data_length / 100)^0.25<br>• ADF: int(4*(data_length / 100)^(2/9)) | The lag order to calculate the test statistic. | |
| PROBABILITY | DOUBLE | 0.9 | The confidence level for confirming stationarity. | |

### Output Table

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| STATS | 1st column | NCLOB | STATS_NAME | Indicates the name of stats. |
| | 2nd column | NCLOB | STATS_VALUE | Indicates the value of corresponding stats. |

### Examples

Assume that:

- DM_PAL is a schema belonging to USER1;
- USER1 has been assigned the AFL__SYS_AFL_AFLPAL_EXECUTE or AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION role.

#### Example 1: KPSS test

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_TEST_DATA_TBL;
CREATE COLUMN TABLE PAL_TEST_DATA_TBL (
    "TIME_STAMP" INTEGER,
    "SERIES" DOUBLE
);
INSERT INTO PAL_TEST_DATA_TBL VALUES (0, 0.0);
INSERT INTO PAL_TEST_DATA_TBL VALUES (1, 3.83);
INSERT INTO PAL_TEST_DATA_TBL VALUES (2, -5.33);
INSERT INTO PAL_TEST_DATA_TBL VALUES (3, -1.5);
INSERT INTO PAL_TEST_DATA_TBL VALUES (4, 2.33);
INSERT INTO PAL_TEST_DATA_TBL VALUES (5, -6.83);
INSERT INTO PAL_TEST_DATA_TBL VALUES (6, -3.0);
```

```
INSERT INTO PAL_TEST_DATA_TBL VALUES (7, 0.83);
INSERT INTO PAL_TEST_DATA_TBL VALUES (8, -8.33);
INSERT INTO PAL_TEST_DATA_TBL VALUES (9, -4.5);
INSERT INTO PAL_TEST_DATA_TBL VALUES (10, -0.67);
INSERT INTO PAL_TEST_DATA_TBL VALUES (11, -9.83);
INSERT INTO PAL_TEST_DATA_TBL VALUES (12, -6.0);
INSERT INTO PAL_TEST_DATA_TBL VALUES (13, -2.17);
INSERT INTO PAL_TEST_DATA_TBL VALUES (14, -11.33);
INSERT INTO PAL_TEST_DATA_TBL VALUES (15, -7.5);
INSERT INTO PAL_TEST_DATA_TBL VALUES (16, -3.67);
INSERT INTO PAL_TEST_DATA_TBL VALUES (17, -12.83);
INSERT INTO PAL_TEST_DATA_TBL VALUES (18, -9.0);
INSERT INTO PAL_TEST_DATA_TBL VALUES (19, -5.17);
INSERT INTO PAL_TEST_DATA_TBL VALUES (20, -14.33);
INSERT INTO PAL_TEST_DATA_TBL VALUES (21, -10.5);
INSERT INTO PAL_TEST_DATA_TBL VALUES (22, -6.67);
INSERT INTO PAL_TEST_DATA_TBL VALUES (23, -15.83);
INSERT INTO PAL_TEST_DATA_TBL VALUES (24, -15.83);
INSERT INTO PAL_TEST_DATA_TBL VALUES (25, -11.075);
INSERT INTO PAL_TEST_DATA_TBL VALUES (26, -12.891);
INSERT INTO PAL_TEST_DATA_TBL VALUES (27, -18.769);
INSERT INTO PAL_TEST_DATA_TBL VALUES (28, -20.585);
INSERT INTO PAL_TEST_DATA_TBL VALUES (29, -15.83);
INSERT INTO PAL_TEST_DATA_TBL VALUES (30, -11.075);
INSERT INTO PAL_TEST_DATA_TBL VALUES (31, -12.891);
INSERT INTO PAL_TEST_DATA_TBL VALUES (32, -18.769);
INSERT INTO PAL_TEST_DATA_TBL VALUES (33, -20.585);
INSERT INTO PAL_TEST_DATA_TBL VALUES (34, -15.83);
INSERT INTO PAL_TEST_DATA_TBL VALUES (35, -11.075);
INSERT INTO PAL_TEST_DATA_TBL VALUES (36, -12.891);
INSERT INTO PAL_TEST_DATA_TBL VALUES (37, -18.769);
INSERT INTO PAL_TEST_DATA_TBL VALUES (38, -20.585);
INSERT INTO PAL_TEST_DATA_TBL VALUES (39, -15.83);
INSERT INTO PAL_TEST_DATA_TBL VALUES (40, -11.075);
INSERT INTO PAL_TEST_DATA_TBL VALUES (41, -12.891);
INSERT INTO PAL_TEST_DATA_TBL VALUES (42, -18.769);
INSERT INTO PAL_TEST_DATA_TBL VALUES (43, -20.585);
INSERT INTO PAL_TEST_DATA_TBL VALUES (44, -15.83);
INSERT INTO PAL_TEST_DATA_TBL VALUES (45, -11.075);
INSERT INTO PAL_TEST_DATA_TBL VALUES (46, -12.891);
INSERT INTO PAL_TEST_DATA_TBL VALUES (47, -18.769);
INSERT INTO PAL_TEST_DATA_TBL VALUES (48, -20.585);
INSERT INTO PAL_TEST_DATA_TBL VALUES (49, -15.83);
INSERT INTO PAL_TEST_DATA_TBL VALUES (50, -11.075);
INSERT INTO PAL_TEST_DATA_TBL VALUES (51, -12.891);
INSERT INTO PAL_TEST_DATA_TBL VALUES (52, -18.769);
INSERT INTO PAL_TEST_DATA_TBL VALUES (53, -20.585);
INSERT INTO PAL_TEST_DATA_TBL VALUES (54, 20.585);
INSERT INTO PAL_TEST_DATA_TBL VALUES (55, 24.525);
INSERT INTO PAL_TEST_DATA_TBL VALUES (56, 25.52);
INSERT INTO PAL_TEST_DATA_TBL VALUES (57, 22.845);
INSERT INTO PAL_TEST_DATA_TBL VALUES (58, 18.536);
INSERT INTO PAL_TEST_DATA_TBL VALUES (59, 15.861);
INSERT INTO PAL_TEST_DATA_TBL VALUES (60, 16.856);
INSERT INTO PAL_TEST_DATA_TBL VALUES (61, 20.795);
INSERT INTO PAL_TEST_DATA_TBL VALUES (62, 24.735);
INSERT INTO PAL_TEST_DATA_TBL VALUES (63, 25.73);
INSERT INTO PAL_TEST_DATA_TBL VALUES (64, 23.055);
INSERT INTO PAL_TEST_DATA_TBL VALUES (65, 18.746);
INSERT INTO PAL_TEST_DATA_TBL VALUES (66, 16.071);
INSERT INTO PAL_TEST_DATA_TBL VALUES (67, 17.066);
INSERT INTO PAL_TEST_DATA_TBL VALUES (68, 21.005);
INSERT INTO PAL_TEST_DATA_TBL VALUES (69, 24.945);
INSERT INTO PAL_TEST_DATA_TBL VALUES (70, 25.94);
INSERT INTO PAL_TEST_DATA_TBL VALUES (71, 23.265);
INSERT INTO PAL_TEST_DATA_TBL VALUES (72, 18.956);
INSERT INTO PAL_TEST_DATA_TBL VALUES (73, 16.281);
INSERT INTO PAL_TEST_DATA_TBL VALUES (74, 17.276);
```

```
INSERT INTO PAL_TEST_DATA_TBL VALUES (75, 21.215);
INSERT INTO PAL_TEST_DATA_TBL VALUES (76, 25.155);
INSERT INTO PAL_TEST_DATA_TBL VALUES (77, 26.15);
INSERT INTO PAL_TEST_DATA_TBL VALUES (78, 23.475);
INSERT INTO PAL_TEST_DATA_TBL VALUES (79, 19.166);
INSERT INTO PAL_TEST_DATA_TBL VALUES (80, 16.491);
INSERT INTO PAL_TEST_DATA_TBL VALUES (81, 17.486);
INSERT INTO PAL_TEST_DATA_TBL VALUES (82, 21.425);
INSERT INTO PAL_TEST_DATA_TBL VALUES (83, 25.365);
INSERT INTO PAL_TEST_DATA_TBL VALUES (84, 26.36);
INSERT INTO PAL_TEST_DATA_TBL VALUES (85, 23.685);
INSERT INTO PAL_TEST_DATA_TBL VALUES (86, 19.376);
INSERT INTO PAL_TEST_DATA_TBL VALUES (87, 16.701);
INSERT INTO PAL_TEST_DATA_TBL VALUES (88, 17.696);
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL ( "NAME" VARCHAR (50),"INT_VALUE"
INTEGER,"DOUBLE_VALUE" DOUBLE,"STRING_VALUE" VARCHAR (100));
INSERT INTO PAL_PARAMETER_TBL VALUES ('METHOD',         0, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('MODE',          1, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('LAG',          5, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('PROBABILITY', NULL, 0.95, NULL);
CALL _SYS_AFL.PAL_STATIONARITY_TEST(PAL_TEST_DATA_TBL, PAL_PARAMETER_TBL, ?);
```

**Expected Result**

| | STATS_NAME | STATS_VALUE |
|---|---|---|
| 1 | stationary | 0 |
| 2 | kpss_stat | 0.26801 |
| 3 | p-value | 0.01 |
| 4 | lags | 5 |
| 5 | number of observations | 89 |
| 6 | critical values | {'10%': 0.119, '5%': 0.146, '2.5%': 0.176, '1%': 0.216} |

**Example 2: ADF test**

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_TEST_DATA_TBL;
CREATE COLUMN TABLE PAL_TEST_DATA_TBL (
    "TIME_STAMP" INTEGER,
    "SERIES" DOUBLE
);
INSERT INTO PAL_TEST_DATA_TBL VALUES (0, 0.0);
INSERT INTO PAL_TEST_DATA_TBL VALUES (1, 3.83);
INSERT INTO PAL_TEST_DATA_TBL VALUES (2, -5.33);
INSERT INTO PAL_TEST_DATA_TBL VALUES (3, -1.5);
INSERT INTO PAL_TEST_DATA_TBL VALUES (4, 2.33);
INSERT INTO PAL_TEST_DATA_TBL VALUES (5, -6.83);
INSERT INTO PAL_TEST_DATA_TBL VALUES (6, -3.0);
INSERT INTO PAL_TEST_DATA_TBL VALUES (7, 0.83);
INSERT INTO PAL_TEST_DATA_TBL VALUES (8, -8.33);
INSERT INTO PAL_TEST_DATA_TBL VALUES (9, -4.5);
INSERT INTO PAL_TEST_DATA_TBL VALUES (10, -0.67);
INSERT INTO PAL_TEST_DATA_TBL VALUES (11, -9.83);
INSERT INTO PAL_TEST_DATA_TBL VALUES (12, -6.0);
INSERT INTO PAL_TEST_DATA_TBL VALUES (13, -2.17);
INSERT INTO PAL_TEST_DATA_TBL VALUES (14, -11.33);
INSERT INTO PAL_TEST_DATA_TBL VALUES (15, -7.5);
INSERT INTO PAL_TEST_DATA_TBL VALUES (16, -3.67);
INSERT INTO PAL_TEST_DATA_TBL VALUES (17, -12.83);
INSERT INTO PAL_TEST_DATA_TBL VALUES (18, -9.0);
```

```
INSERT INTO PAL_TEST_DATA_TBL VALUES (19, -5.17);
INSERT INTO PAL_TEST_DATA_TBL VALUES (20, -14.33);
INSERT INTO PAL_TEST_DATA_TBL VALUES (21, -10.5);
INSERT INTO PAL_TEST_DATA_TBL VALUES (22, -6.67);
INSERT INTO PAL_TEST_DATA_TBL VALUES (23, -15.83);
INSERT INTO PAL_TEST_DATA_TBL VALUES (24, -15.83);
INSERT INTO PAL_TEST_DATA_TBL VALUES (25, -11.075);
INSERT INTO PAL_TEST_DATA_TBL VALUES (26, -12.891);
INSERT INTO PAL_TEST_DATA_TBL VALUES (27, -18.769);
INSERT INTO PAL_TEST_DATA_TBL VALUES (28, -20.585);
INSERT INTO PAL_TEST_DATA_TBL VALUES (29, -15.83);
INSERT INTO PAL_TEST_DATA_TBL VALUES (30, -11.075);
INSERT INTO PAL_TEST_DATA_TBL VALUES (31, -12.891);
INSERT INTO PAL_TEST_DATA_TBL VALUES (32, -18.769);
INSERT INTO PAL_TEST_DATA_TBL VALUES (33, -20.585);
INSERT INTO PAL_TEST_DATA_TBL VALUES (34, -15.83);
INSERT INTO PAL_TEST_DATA_TBL VALUES (35, -11.075);
INSERT INTO PAL_TEST_DATA_TBL VALUES (36, -12.891);
INSERT INTO PAL_TEST_DATA_TBL VALUES (37, -18.769);
INSERT INTO PAL_TEST_DATA_TBL VALUES (38, -20.585);
INSERT INTO PAL_TEST_DATA_TBL VALUES (39, -15.83);
INSERT INTO PAL_TEST_DATA_TBL VALUES (40, -11.075);
INSERT INTO PAL_TEST_DATA_TBL VALUES (41, -12.891);
INSERT INTO PAL_TEST_DATA_TBL VALUES (42, -18.769);
INSERT INTO PAL_TEST_DATA_TBL VALUES (43, -20.585);
INSERT INTO PAL_TEST_DATA_TBL VALUES (44, -15.83);
INSERT INTO PAL_TEST_DATA_TBL VALUES (45, -11.075);
INSERT INTO PAL_TEST_DATA_TBL VALUES (46, -12.891);
INSERT INTO PAL_TEST_DATA_TBL VALUES (47, -18.769);
INSERT INTO PAL_TEST_DATA_TBL VALUES (48, -20.585);
INSERT INTO PAL_TEST_DATA_TBL VALUES (49, -15.83);
INSERT INTO PAL_TEST_DATA_TBL VALUES (50, -11.075);
INSERT INTO PAL_TEST_DATA_TBL VALUES (51, -12.891);
INSERT INTO PAL_TEST_DATA_TBL VALUES (52, -18.769);
INSERT INTO PAL_TEST_DATA_TBL VALUES (53, -20.585);
INSERT INTO PAL_TEST_DATA_TBL VALUES (54, 20.585);
INSERT INTO PAL_TEST_DATA_TBL VALUES (55, 24.525);
INSERT INTO PAL_TEST_DATA_TBL VALUES (56, 25.52);
INSERT INTO PAL_TEST_DATA_TBL VALUES (57, 22.845);
INSERT INTO PAL_TEST_DATA_TBL VALUES (58, 18.536);
INSERT INTO PAL_TEST_DATA_TBL VALUES (59, 15.861);
INSERT INTO PAL_TEST_DATA_TBL VALUES (60, 16.856);
INSERT INTO PAL_TEST_DATA_TBL VALUES (61, 20.795);
INSERT INTO PAL_TEST_DATA_TBL VALUES (62, 24.735);
INSERT INTO PAL_TEST_DATA_TBL VALUES (63, 25.73);
INSERT INTO PAL_TEST_DATA_TBL VALUES (64, 23.055);
INSERT INTO PAL_TEST_DATA_TBL VALUES (65, 18.746);
INSERT INTO PAL_TEST_DATA_TBL VALUES (66, 16.071);
INSERT INTO PAL_TEST_DATA_TBL VALUES (67, 17.066);
INSERT INTO PAL_TEST_DATA_TBL VALUES (68, 21.005);
INSERT INTO PAL_TEST_DATA_TBL VALUES (69, 24.945);
INSERT INTO PAL_TEST_DATA_TBL VALUES (70, 25.94);
INSERT INTO PAL_TEST_DATA_TBL VALUES (71, 23.265);
INSERT INTO PAL_TEST_DATA_TBL VALUES (72, 18.956);
INSERT INTO PAL_TEST_DATA_TBL VALUES (73, 16.281);
INSERT INTO PAL_TEST_DATA_TBL VALUES (74, 17.276);
INSERT INTO PAL_TEST_DATA_TBL VALUES (75, 21.215);
INSERT INTO PAL_TEST_DATA_TBL VALUES (76, 25.155);
INSERT INTO PAL_TEST_DATA_TBL VALUES (77, 26.15);
INSERT INTO PAL_TEST_DATA_TBL VALUES (78, 23.475);
INSERT INTO PAL_TEST_DATA_TBL VALUES (79, 19.166);
INSERT INTO PAL_TEST_DATA_TBL VALUES (80, 16.491);
INSERT INTO PAL_TEST_DATA_TBL VALUES (81, 17.486);
INSERT INTO PAL_TEST_DATA_TBL VALUES (82, 21.425);
INSERT INTO PAL_TEST_DATA_TBL VALUES (83, 25.365);
INSERT INTO PAL_TEST_DATA_TBL VALUES (84, 26.36);
INSERT INTO PAL_TEST_DATA_TBL VALUES (85, 23.685);
INSERT INTO PAL_TEST_DATA_TBL VALUES (86, 19.376);
```

```
INSERT INTO PAL_TEST_DATA_TBL VALUES (87, 16.701);
INSERT INTO PAL_TEST_DATA_TBL VALUES (88, 17.696);
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL ( "NAME" VARCHAR (50),"INT_VALUE"
INTEGER,"DOUBLE_VALUE" DOUBLE,"STRING_VALUE" VARCHAR (100));
INSERT INTO PAL_PARAMETER_TBL VALUES ('METHOD',         1, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('MODE',          1, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('LAG',          5, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('PROBABILITY', NULL, 0.95, NULL);
CALL _SYS_AFL.PAL_STATIONARITY_TEST(PAL_TEST_DATA_TBL, PAL_PARAMETER_TBL, ?);
```

**Expected Result**

| | STATS_NAME | STATS_VALUE |
|---|---|---|
| 1 | stationary | 0 |
| 2 | adf_stat | -1.76888 |
| 3 | p-value | 0.7195647879256077 |
| 4 | lags | 5 |
| 5 | number of observations | 83 |
| 6 | critical values | {'1%': -4.0722, '5%': -3.46477, '10%': -3.15881} |

## 5.5.32 Trend Test

The algorithm is able to identify whether a time series has an upward or downward trend or not, and calculate the de-trended time series.

Two methods are provided for identifying the trend: the difference-sign test and the rank test.

**Difference-Sign Test**

The difference-sign test counts the number S of times that `x(t)-x(t-1)` is positive, `t=1,2,...,n`. For an IID (Independent and Identically Distributed) series, the theoretical expectation of S is

$\mu_s$=`(n-1)/2`,

and the corresponding standard deviation is

$$\sigma_S = \sqrt{(n+1)/12}$$

For large n, S approximately behaves Gaussian distribution $N(\mu_S, \sigma_S^2)$. Hence, a large positive or negative value of $S-\mu_S$ indicates the presence of an increasing (or decreasing) trend. The data is considered to have a trend if $|S-\mu_S|/\sigma_S>\Phi_{1-\alpha}$ at some significance level α, otherwise no trend exists. Nevertheless, the difference-sign test must be used with great caution. In the case of large proportion of tie data, for example, it may give a result of negative trend, while in fact it has no trend.

**Rank Test**

The second solution is the rank test, which is usually known as Mann-Kendall (MK) Test (Mann 1945, Kendall 1975, Gilbert 1987). It tests whether to reject the null hypothesis ($H_0$) and accept the alternative hypothesis ($H_a$), where

$H_0$: No monotonic trend

$H_a$: Monotonic trend is present

$\alpha$: The tolerance probability that falsely concludes that a trend exists when there is none. $0<\alpha<0.5$.

MK test still requires calculating S, which is the number of positive pairs minus the negative pairs, and satisfies Gaussian distribution. Compared with the S in difference-sign test, the two elements of a pair need not be adjacent, and S can be positive, zero, or negative.

$$S=\sum_{i=1}^{n}\sum_{j=i+1}^{n} sgn(x_i-x_j)$$

For a small enough significance level $\alpha$, say, 0.05, MK test is expected to achieve a quite satisfactory estimation for trend. At the same time, MK test requires that the length of time series should be at least 4. Provided the length of data set is only 3, therefore, difference-sign strategy is applied.

The resulting trend indicator has three possible numeric values: 1 indicating upward trend, –1 indicating downward trend, and 0 for no trend, respectively.

Should there be a trend, a de-trended time series with first differencing approach is given:

$w(t)=x(t)-x(t-1)$,

where $x(t)$ is the input time series, and $w(t)$ is the de-trended time series. Apparently, the length of de-trended series is exactly one less than the input's (lack of the first period). On the other hand, the output series is just the input one if no trend is identified. Note that the resulting time series is sorted by time periods.

## Prerequisites

- No null data in the inputs. The time periods should be unique (no duplication).
- The length of time series must be at least 3.

## _SYS_AFL.PAL_TREND_TEST

This function identifies the trend and calculates de-trended series of a time series.

### Overview of all Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<STATISTICS table>` | OUT |
| 4 | `<DETRENDED table>` | OUT |

### Input Table

| Table | Column | Reference for Output Table | Data Type | Description |
|---|---|---|---|---|
| DATA | 1st column | ID | INTEGER | Time stamp.<br>Time stamp does not need to be in order, but must be unique. |
| | 2nd column | | INTEGER, DOUBLE, or DECIMAL(p, s) | Time series. |

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| METHOD | INTEGER | 1 | The method used to identify trend:<br>• 1: MK test<br>• 2: Difference-sign test |
| ALPHA | DOUBLE | 0.05 | Significance value. The value range is (0,0.5). |

### Output Tables

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| STATISTICS | 1st column | NVARCHAR(100) | STAT_NAME | The name of the series properties:<br>• TREND: -1 for downward trend, 0 for no trend, and 1 for upward trend<br>• S: Number of S defined as above<br>• P-VALUE: The p-value of the observed S |
| | 2nd column | DOUBLE | STAT_VALUE | Values |
| DETRENDED | 1st column | ID (in input table) data type | ID (in input table) column name | Time stamp that is monotonically increasing sorted |
| | 2nd column | DOUBLE | DETRENDED_SERIES | The corresponding detrended time series.<br>The first value absents if trend presents. |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_TREND_DATA_TBL;
CREATE COLUMN TABLE PAL_TREND_DATA_TBL (
    "TIME_STAMP" INTEGER,
    "SERIES" DOUBLE
);
INSERT INTO PAL_TREND_DATA_TBL VALUES (1, 1500);
INSERT INTO PAL_TREND_DATA_TBL VALUES (2, 1510);
INSERT INTO PAL_TREND_DATA_TBL VALUES (3, 1550);
INSERT INTO PAL_TREND_DATA_TBL VALUES (4, 1650);
INSERT INTO PAL_TREND_DATA_TBL VALUES (5, 1620);
INSERT INTO PAL_TREND_DATA_TBL VALUES (6, 1690);
INSERT INTO PAL_TREND_DATA_TBL VALUES (7, 1695);
INSERT INTO PAL_TREND_DATA_TBL VALUES (8, 1700);
INSERT INTO PAL_TREND_DATA_TBL VALUES (9, 1710);
INSERT INTO PAL_TREND_DATA_TBL VALUES (10, 1705);
INSERT INTO PAL_TREND_DATA_TBL VALUES (11, 1708);
INSERT INTO PAL_TREND_DATA_TBL VALUES (12, 1715);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR(100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
```

```
    "STRING_VALUE" VARCHAR(100)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('ALPHA', NULL, 0.05, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('METHOD', 1, NULL, NULL);
CALL _SYS_AFL.PAL_TREND_TEST(PAL_TREND_DATA_TBL, "#PAL_PARAMETER_TBL", ?, ?);
```

**Expected Result**

STATISTICS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | TREND | 1 |
| 2 | S | 60 |
| 3 | P-VALUE | 0.00002607... |

DETRENDED:

| | TIME_STAMP | DETRENDED_SEARIES |
|---|---|---|
| 1 | 2 | 10 |
| 2 | 3 | 40 |
| 3 | 4 | 100 |
| 4 | 5 | -30 |
| 5 | 6 | 70 |
| 6 | 7 | 5 |
| 7 | 8 | 5 |
| 8 | 9 | 10 |
| 9 | 10 | -5 |
| 10 | 11 | 3 |
| 11 | 12 | 7 |

> **Note**
>
> The wrapper procedure generation approach to call PAL functions is still supported. For more information, refer to the older versions (prior to SAP HANA Platform 2.0 SPS 02) of this document.

## 5.5.33 VARMA

The vector autoregressive moving average models (VARMA) is a vector form of autoregressive integrated moving average (ARIMA) that can be used to examine the relationships among several variables in multivariate time series analysis, comparing to ARIMA which is used in univariate time series.

For a k-dimensional vector of time series,

$\mathbf{y_t} = [y_{1,t}', \ldots, y_{k,t}']'$,

the natural generalization of ARMA$(p,q)(P,Q)_s$ model is the following VARMA$(p,q)(P,Q)_s$ model,

$\boldsymbol{\varphi}(B)\boldsymbol{\Phi}(B^s)\mathbf{y}_t = \mu + \boldsymbol{\theta}(B)\boldsymbol{\Theta}(B^s)\mathbf{a}_t$,

where,

$$\boldsymbol{\phi}(B) = \mathbf{I}_k - \sum_{i}^{p} \boldsymbol{\phi}_i B^i ,$$

$$\boldsymbol{\Phi}(B) = \mathbf{I}_k - \sum_{i}^{P} \boldsymbol{\Phi}_i B^{i \cdot s} ,$$

$$\boldsymbol{\theta}(B) = \mathbf{I}_k - \sum_{i}^{q} \boldsymbol{\theta}_i B^i ,$$

and

$$\boldsymbol{\Theta}(B) = \mathbf{I}_k - \sum_{i}^{Q} \boldsymbol{\Theta}_i B^{i \cdot s} ,$$

are $k \times k$ matrix polynomials. Applying the backshift operator $B$ with order $i$ to matrix $\mathbf{M}$ would get $B^i \mathbf{M}_t = \mathbf{M}_{t-i}$ . $\mathbf{I}_k$ is the k-dimensional identity matrix. The parameters $(p,q)(P,Q)_s$ are categorized to a regular order $(p,q)$ and a seasonal order $(P,Q)$ with $s$ donating the seasonal period, $p$ and $P$ the autoregressive order, and $q$ and $Q$ the moving average order. $\mathbf{a}_t$ is a $k \times 1$ vector of mean zero white 'noise' variable, and is assumed but not restricted to be a multivariate normal process, the components of the covariance matrix $\boldsymbol{\Sigma}$ could be written as

$$\mathrm{E}(\mathbf{a}_t \mathbf{a}_s) = \begin{cases} 0, & s \neq t \\ \sigma_{ts}, & \text{others} \end{cases} .$$

The $\Sigma$ is not necessarily of diagonal form since the components of $\mathbf{a}_t$ are contemporaneously correlated with each other.

It can be noted that above model reduces to the univariate ARMA model when $k=1$. The estimation of VARMA models is carried out using conditional maximum likelihood estimation (MLE) method. The conditional log-likelihood function given the data $\mathbf{y}_t$ is,

$$l(\boldsymbol{\alpha}, \boldsymbol{\Sigma}; \mathbf{y}_t) = -\frac{n}{2} \log 2\pi - \frac{n}{2} \log|\boldsymbol{\Sigma}| - \frac{1}{2} \mathbf{a}_t' (\mathbf{I}_n \otimes \boldsymbol{\Sigma}^{-1}) \mathbf{a}_t ,$$

Where $\boldsymbol{\alpha} = \mathrm{vec}(\boldsymbol{\varphi}_1, \ldots, \boldsymbol{\varphi}_p, \ldots, \boldsymbol{\varphi}_{(s*P+p)}, \boldsymbol{\theta}_1, \ldots \boldsymbol{\theta}_q, \ldots, \boldsymbol{\theta}_{(s*Q+q)})$ stands for the vector of AR, seasonal AR, MA and seasonal MA parameters, each component of which has a dimension of $k \times k$.

Besides VARMA model, PAL also provides pure VAR and VMA models. VARMA model can reduce to VAR when $q, Q=0$, and to VMA when $p, P=0$, but the estimation methods of the three models are different. VAR is simple to estimate but the presence of non-trivial MA term may make the results misleading, while it may take time for VMA and VARMA to employ box-constrained L-BFGS (L-BFGS-B) to minimize the conditional likelihood function.

The orders of VAR/VMA/VARMA models could be specified automatically. There're two options for VARMA model: performing grid search to minimize some information criteria (also applied for seasonal data), or computing the p-value table of the extended cross-correlation matrices (eccm) and comparing its elements with the type I error. VAR model uses grid search to specify orders while VMA model performs multivariate Ljung-Box tests to specify orders.

To explore the relation between variables, PAL support the computation of the impulse response function (IRF) of a give VAR or VARMA model.

The summary of the output models,

| Name | Description |
| --- | --- |
| model | Model type, could be VAR, VMA, or VARMA |
| exogCols | Exogenous variables |
| endogCols | Endogenous variables |
| p | Order of vector AR |
| q | Order of vector MA |
| d | Degree of differencing |
| m | The lag number of exogenous variables |
| s | The seasonal period |
| P | Order of vector seasonal AR |
| Q | Order of vector seasonal MA |
| D | Degree of seasonal differencing |
| c | Number of exogenous variables |
| k | Number of endogenous variables |
| nT | Length of time series |
| AIC | Akaike information criterion |
| BIC | Bayesian information criterion |
| mu | The constant parameters, dimension: `1-by-k` |
| seMu | The standard error of the me estimates, dimension: `1-by-k` |
| var | Vector AR coefficient matrices, dimension: `kp-by-k` |
| seVar | The standard error of the var estimates, dimension: `kp-by-k` |
| vma | Vector MA coefficient matrices, dimension: `kq-by-k` |
| seVma | The standard error of the vma estimates, dimension: `kq-by-k` |
| svar | Seasonal VAR coefficient matrices, dimension: `kP-by-k` |

| Name | Description |
| --- | --- |
| seSVar | The standard error of the svar estimates, dimension: `kP-by-k` |
| svma | Seasonal VMA coefficient matrices, dimension: `kQ-by-k` |
| seSVma | The standard error of the svma estimates, dimension: `kQ-by-k` |
| beta | Coefficient matrix of exogenous variables, dimension: `c(m+1)-by-k` |
| seBeta | The standard error of the beta estimates, dimension: `c(m+1)-by-k` |
| max_p | Maximal order of `p` |
| max_q | Maximal order of `q` |
| eccm | The row-major p-value table of the extended cross-correlation matrices, dimension: `(max_p +1)□(max_q+1)` |
| covariance | Residual covariance matrix, dimension: `k×k` |
| ytHistory | Historical data matrix of endogenous variables |
| xtHistory | Historical data matrix of exogenous variables |
| at | The residual series |

## Prerequisite

- The input data must not have missing value.
- The ID column must not have any identical value.

## _SYS_AFL.PAL_VARMA

### Overview of All Tables

| Position | Table Name | Parameter Type |
| --- | --- | --- |
| 1 | `<DATA table>` | IN |

| Position | Table Name | Parameter Type |
|---|---|---|
| 2 | `<PARAMETER table>` | IN |
| 3 | `<MODEL table>` | OUT |
| 4 | `<FIT table>` | OUT |
| 5 | `<IRF table>` | OUT |

### Input Table

| Table | Column | Reference for Output Table | Data Type | Description |
|---|---|---|---|---|
| DATA | 1st column | ID | INTEGER | Time stamp.<br>No identical value is allowed. |
| | other columns | | INTEGER, DOUBLE, or DECIMAL(p, s) | Data of explanatory variables and dependent variables |

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| P | INTEGER | -1 | The vector auto-regression order:<br>• -1: automatic<br>• >=0: user-defined | |
| D | INTEGER | 0 | The differencing order. | |
| Q | INTEGER | -1 | The vector moving average order:<br>• -1: automatic<br>• >=0: user-defined | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| M | INTEGER | -1 | The lag number of explanatory variables:<br>• -1: automatic identify<br>• >=0: user-defined | Valid only when MODEL = 0. |
| MAX_P | INTEGER | • 6 if MODEL = 0 or if MODEL = 2 and SEARCH_METHOD = 0<br>• 2 if MODEL = 2 and SEARCH_METHOD = 1 | The maximum vector auto-regression order entertained. | Valid only when MODEL = 0 or 2. |
| MAX_Q | INTEGER | • 8 if MODEL = 1<br>• 5 if MODEL = 2 and SEARCH_METHOD = 0<br>• 2 if MODEL = 2 and SEARCH_METHOD = 1 | The maximum vector moving average order entertained. | Valid only when MODEL = 1 or 2. |
| MAX_M | INTEGER | 4 | The maximum lag number of explanatory variables. | Valid only when MODEL = 0. |
| SEASONAL_PERIOD | INTEGER | 0 | Seasonal period. | |
| SEASONAL_P | INTEGER | -1 | The seasonal vector auto-regression order:<br>• -1: automatic<br>• >=0: user-defined | Valid only when SEASONAL_PERIOD is larger than 1. |
| SEASONAL_D | INTEGER | 0 | The seasonal differencing order. | Valid only when SEASONAL_PERIOD is larger than 1. |
| SEASONAL_Q | INTEGER | -1 | The seasonal vector moving average order:<br>• -1: automatic identify<br>• >=0: user-defined | Valid only when SEASONAL_PERIOD is larger than 1. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| MAX_SEASONAL_P | INTEGER | • 3 if MODEL = 0<br>• 1 if MODEL = 2 and SEARCH_METHOD = 1 | The maximum seasonal vector auto-regression order entertained. | Valid only when MODEL = 0 and 2. |
| MAX_SEASONAL_Q | INTEGER | 1 | The maximum seasonal vector moving average order entertained. | Valid only when MODEL = 2 and SEARCH_METHOD = 1. |
| MODEL | INTEGER | 2 | Model type:<br>• 0: VAR<br>• 1: VMA<br>• 2: VARMA | VAR is valid only when Q and SEASONAL_Q vanish.<br>VMA is valid only when P, SEASONAL_P, and SEASONAL_Q vanish.<br>VMA does not support seasonality and explanatory variables. |
| SEARCH_METHOD | INTEGER | 1 | Specifies the orders of a VARMA model:<br>• 0: eccm<br>• 1: grid search | eccm is valid only when SEASONAL_PERIOD <= 1. |
| INITIAL_GUESS | INTEGER | 1 | The model used as initial estimation for VARMA:<br>• 0: ARMA<br>• 1: VAR | Valid only for VARMA. |
| INFORMATION_CRITERION | INTEGER | 0 | Information criteria for order specification:<br>• 0: Akaike information criterion (AIC)<br>• 1: Bayesian information criterion (BIC) | |
| MAX_ITERATION | INTEGER | 200 | Maximum number of iterations of L-BFGS-B optimizer. | Valid only for VMA and VARMA. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| FINITE_DIFFERENCE_ACCURACY | INTEGER | 1 | Polynomial order of finite difference. Approximate the gradient of objective function with finite difference. The valid range is from 1 to 4. | Valid only for VMA and VARMA. |
| DISPLACEMENT | DOUBLE | $2.2\times10^{-6}$ | The step length for finite-difference method. | Valid only for VMA and VARMA. |
| FTOL | DOUBLE | $1.0\times10^{-5}$ | Tolerance for objective convergence test. | Valid only for VMA and VARMA. |
| GTOL | DOUBLE | $1.0\times10^{-5}$ | Tolerance for gradient convergence test. | Valid only for VMA and VARMA. |
| INCLUDE_MEAN | INTEGER | • 1 when `D + SEASONAL_D < 1`<br>• 0 otherwise | An indicator whether to include the mean vector:<br>• 0: no<br>• 1: yes | Nonvanishing only when D + SEASONAL_D < 1. |
| HESSIAN | INTEGER | 0 | Specifies whether to calculate the Hessian matrix:<br>• 0: no<br>• 1: yes | VMA and VARMA will output standard error of parameter estimates only when HESSIAN = 1. |
| OUTPUT_FITTED | INTEGER | 1 | Control fittings and residuals output:<br>• 0: no<br>• 1: yes | |
| CALCULATE_IRF | INTEGER | 0 | A switch whether to calculate impulse response function (IRF):<br>• 0: no<br>• 1: yes | Does not apply to pure VMA model. |
| IRF_LAGS | INTEGER | 8 | The number of lags of the IRF to be calculated. | Valid only when CALCULTE_IRF = 1. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| ALPHA | DOUBLE | 0.05 | Type-I error used in the Ljung-Box tests and eccm. | |
| THREAD_RATIO | DOUBLE | -1 | The ratio of available threads.<br>• 0 : single thread<br>• 0~1: percentage<br>• others: heuristically determined | |
| EXOGENOUS_VARIABLE | VARCHAR or NVARCHAR | No default value | The exogenous variables. | Cannot be the name of the first column (time stamp column). |
| EXCLUDED_FEATURE | VARCHAR or NVARCHAR | No default value | Excludes the indicated feature column. | Cannot be the name of the first column (time stamp column). |

### Output Tables

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| MODEL | 1st column | INTEGER | CONTENT_INDEX | Row index |
| | 2nd column | NVARCHAR (5000) | CONTENT_VALUE | Model content |
| FIT | 1st column | NVARCHAR (5000) | NAMECOL | Column name of endogenous variables |
| | 2nd column | ID (in input table) column type | ID (in input table) column name | Time stamp |
| | 3rd column | DOUBLE | FITTED | Fitted values |
| | 4th column | DOUBLE | RESIDUALS | Residual values |
| IRF | 1st column | NVARCHAR (5000) | COL1 | Column name of the first variable |
| | 2nd column | NVARCHAR (5000) | COL2 | Column name of the second variable |
| | 3rd column | INTEGER | IDX | Lag index |
| | 4th column | DOUBLE | RESPONSE | Impulse response |

> **Note**
>
> - For VAR model, the first `D + SEASONALITY_PERIOD*SEASONAL_D + max(P + SEASONALITY_PERIOD * SEASONAL_P, M)` values of FITTED and RESIDUALS are absent.
> - For VMA and VARMA models, the first `D + SEASONALITY_PERIOD * SEASONAL_D` values of FITTED and RESIDUALS are absent.
> - The RESPONSE in IRF table stands for the effect of a unit change in COL2 on COL1.

**Examples**

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

**Example 1: VARX**

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_VARMA_DATA_TBL;
CREATE COLUMN TABLE PAL_VARMA_DATA_TBL("timestamp" INTEGER, "Y1" DOUBLE, "X"
DOUBLE, "Y2" DOUBLE);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(1 ,9.8,6.4,8.2);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(2 ,9.7,6.4,8.1);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(3 ,9.8,6.3,8);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(4 ,9.7,6.2,7.9);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(5 ,9.6,6.3,7.8);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(6 ,9.6,6.8,7.6);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(7 ,9.6,6.8,7.5);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(8 ,9,6.8,7.5);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(9 ,9.2,6.8,7.4);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(10,9.2,6.7,7.5);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(11,9.1,6.6,7.6);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(12,9,6.6,7.5);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(13,8.8,6,7.2);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(14,8.8,6,7.7);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(15,8.7,5.9,7);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(16,8.3,5.8,6.5);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(17,8.2,5.9,6.4);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(18,8.2,6.3,6.3);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(19,8.2,6.3,6.1);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(20,8.4,6.4,6);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(21,8.1,6.4,6.1);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(22,7.8,6.5,6);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(23,7.7,6.5,5.9);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(24,7.5,6.3,5.9);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(25,7.2,6.5,5.7);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(26,7.2,6.4,5.8);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(27,7,6.3,5.8);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(28,7,6,5.5);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(29,6.9,6.2,5.4);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(30,7,5.9,5.4);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(31,7.1,6,5.3);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(32,7.4,6,5.4);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(33,6.9,5.8,5.5);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(34,6.8,5.8,5.4);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(35,7,5.6,5.4);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(36,7.1,5.6,5.4);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(37,7,5.3,5.7);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(38,7,5.3,5.6);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(39,7.2,5.4,5.5);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(40,7.6,5.5,5.8);
DROP TABLE PAL_VARMA_MODEL_TBL;
```

```
CREATE COLUMN TABLE PAL_VARMA_MODEL_TBL ("INDICES" INTEGER, "CONTENT"
NVARCHAR(5000));
DROP TABLE PAL_VARMA_FIT_TBL;
CREATE COLUMN TABLE PAL_VARMA_FIT_TBL ("colNAME" NVARCHAR(100), "INDICES"
INTEGER, "FITTED" DOUBLE, "RES" DOUBLE);
DROP TABLE PAL_VARMA_IRF_TBL;
CREATE COLUMN TABLE PAL_VARMA_IRF_TBL( "colNAME" NVARCHAR(100), "colNAME2"
NVARCHAR(100), "INDICES" INTEGER, "IRF" DOUBLE);
DROP TABLE PAL_VARMA_CTRL_TBL;
CREATE COLUMN TABLE PAL_VARMA_CTRL_TBL("NAME" NVARCHAR(100), "Intargs" INTEGER,
"Dblargs" DOUBLE, "Strargs" NVARCHAR(100));
INSERT INTO PAL_VARMA_CTRL_TBL VALUES('MODEL', 0, NULL, NULL);
INSERT INTO PAL_VARMA_CTRL_TBL VALUES('CALCULATE_IRF', 1, NULL, NULL);
INSERT INTO PAL_VARMA_CTRL_TBL VALUES('EXOGENOUS_VARIABLE', NULL, NULL, 'X');
do begin
    lt_data = select * from PAL_VARMA_DATA_TBL;
    lt_control = select * from PAL_VARMA_CTRL_TBL;
    CALL _SYS_AFL.PAL_VARMA(:lt_data, :lt_control, lt_model, lt_fit, lt_irf);
    INSERT INTO PAL_VARMA_MODEL_TBL SELECT * FROM :lt_model;
    select * from :lt_model;
    select * from :lt_fit;
    select * from :lt_irf;
end
```

**Expected Result**

MODEL:

| | CONTENT_INDEX | CONTENT_VALUE |
|---|---|---|
| 1 | 0 | {"model":"VAR"} |
| 2 | 1 | {"exogCols":["X"]} |
| 3 | 2 | {"endogCols":["Y1","Y2"]} |
| 4 | 3 | {"D":0,"P":0,"c":1,"d":0,"k":2,"m":2,"nT":40,"p":1,"s":0} |
| 5 | 4 | {"AIC":-6.6759375491341144} |
| 6 | 5 | {"mu":[1.3382245638290442,-0.2833863022212889]} |
| 7 | 6 | {"seMu":[0.4654278039151056 9,0.4638053127648028]} |
| 8 | 7 | {"var":[0.67628712931724522,0.31755912612339898,0.3694257878249... |
| 9 | 8 | {"seVar":[0.12565511601049023,0.12691238409917999,0.12521707962... |
| 10 | 9 | {"beta":[0.24356907462257027,0.019885139690131837,-0.3922026122... |
| 11 | 10 | {"seBeta":[0.16181413185406873,0.2036476194714997,0.16573107327... |
| 12 | 11 | {"covariance":[0.029705363398468566,0.014345278736176192,0.01434... |
| 13 | 12 | {"ytHistory":[7.5999999999999996,5.7999999999999998]} |
| 14 | 13 | {"xtHistory":[5.4000000000000004,5.5]} |
| 15 | 14 | {"savePoint":40} |

FIT:

| | NAMECOL | IDX | FITTING | RESIDUAL |
|---|---|---|---|---|
| 1 | Y1 | 1 | ? | ? |
| 2 | Y1 | 2 | ? | ? |
| 3 | Y1 | 3 | 9.622091985767515 | 0.1779080142324858 |
| 4 | Y1 | 4 | 9.631619364655633 | 0.06838063534436678 |
| 5 | Y1 | 5 | 9.59382339382633 | 0.006176606173669441 |
| 6 | Y1 | 6 | 9.657432080784087 | -0.05743208078408735 |
| 7 | Y1 | 7 | 9.564642564182954 | 0.03535743581704587 |
| 8 | Y1 | 8 | 9.336785345463007 | -0.33678534546300654 |
| 9 | Y1 | 9 | 8.931013067872659 | 0.2689869321273406 |
| 10 | Y1 | 10 | 9.01015767366151 | 0.1898423263384892 |
| 11 | Y1 | 11 | 9.01556816484258 | 0.08443183515741914 |
| 12 | Y1 | 12 | 9.016927111775704 | -0.0169271117757041... |
| 13 | Y1 | 13 | 8.81062130267962 | -0.0106213026796186... |
| 14 | Y1 | 14 | 8.568165055165071 | 0.2318349448349295 |
| 15 | Y1 | 15 | 8.937909278093642 | -0.23790927809364248 |
| 16 | Y1 | 16 | 8.621643755444266 | -0.32164375544426527 |
| 17 | Y1 | 17 | 8.253937995370435 | -0.0539379953704362 |
| 18 | Y1 | 18 | 8.293189774865933 | -0.09318977486593383 |
| 19 | Y1 | 19 | 8.230167656908124 | -0.03016765690812484 |
| 20 | Y1 | 20 | 8.034131694259617 | 0.3658683057403831 |
| 21 | Y1 | 21 | 8.139621721479738 | -0.0396217214797385... |
| 22 | Y1 | 22 | 7.953628141537639 | -0.15362814153763882 |
| 23 | Y1 | 23 | 7.720974604099139 | -0.02097460409913854 |
| 24 | Y1 | 24 | 7.533655902409041 | -0.03365590240904126 |
| 25 | Y1 | 25 | 7.4431352635320795 | -0.24313526353207937 |

IRF:

| | COL1 | COL2 | IDX | RESPONSE |
|---|---|---|---|---|
| 1 | Y1 | X | 0 | 0.24356907462257027 |
| 2 | Y1 | X | 1 | 0.13974896502778206 |
| 3 | Y1 | X | 2 | -0.351428758978154... |
| 4 | Y1 | X | 3 | -0.148630196768491... |
| 5 | Y1 | X | 4 | -0.091541614831017... |
| 6 | Y1 | X | 5 | -0.074284350849734... |
| 7 | Y1 | X | 6 | -0.067954815510170... |
| 8 | Y1 | X | 7 | -0.064661384360696... |
| 9 | Y1 | Y1 | 0 | -0.141261268340542... |
| 10 | Y1 | Y1 | 1 | -0.169217536912334... |
| 11 | Y1 | Y1 | 2 | 0.28037786502085293 |
| 12 | Y1 | Y1 | 3 | 0.02826268727013706 |
| 13 | Y1 | Y1 | 4 | -0.038972065119590... |
| 14 | Y1 | Y1 | 5 | -0.055792020015584... |
| 15 | Y1 | Y1 | 6 | -0.058900581710165... |
| 16 | Y1 | Y1 | 7 | -0.058315033725969... |
| 17 | Y1 | Y2 | 0 | 0.17235243949091225 |
| 18 | Y1 | Y2 | 1 | 0.1429908923269521 |
| 19 | Y1 | Y2 | 2 | 0.13182540894332234 |
| 20 | Y1 | Y2 | 3 | 0.1257303722485269 |
| 21 | Y1 | Y2 | 4 | 0.12111948192065905 |
| 22 | Y1 | Y2 | 5 | 0.11701049872615621 |
| 23 | Y1 | Y2 | 6 | 0.11313212249981333 |
| 24 | Y1 | Y2 | 7 | 0.1094072226432048 |
| 25 | Y2 | X | 0 | 0.08323223493992138 |
| 26 | Y2 | X | 1 | 0.11060147847669939 |

**Example 2: VMA**

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_VARMA_DATA_TBL;
CREATE COLUMN TABLE PAL_VARMA_DATA_TBL("timestamp" INTEGER, "Y1" DOUBLE, "Y2"
DOUBLE);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(1 ,0.058011,  0.081767);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(2 ,0.029241,  0.055524);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(3 ,0.025896,  0.041304);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(4 ,0.005667,  0.00078);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(5 ,0.019208,  0.04959);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(6 ,-0.02467,  -0.040046);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(7 ,0.035668,  0.00937);
```

```
INSERT INTO PAL_VARMA_DATA_TBL VALUES(8 ,0.024092,  0.015614);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(9 ,-0.016177, -0.016758);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(10,0.028087,  0.023155);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(11,0.048685,  0.043721);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(12,0.005025,  -0.00404);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(13,-0.04526,  -0.029078);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(14,0.02362,   0.015649);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(15,-0.003898, -0.005748);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(16,-0.06129,  -0.059029);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(17,-0.078846, -0.10734);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(18,-0.083259, -0.085625);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(19,0.068084,  0.051869);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(20,0.020698,  0.022439);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(21,-0.044818, -0.059073);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(22,0.012499,  -0.017068);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(23,0.099604,  0.125354);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(24,0.019592,  -0.007943);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(25,0.045429,  0.066401);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(26,-0.023884, -0.017365);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(27,0.038267,  0.018539);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(28,0.052166,  0.032948);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(29,0.015735,  0.036258);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(30,-0.020691, -0.020601);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(31,0.003275,  -0.020085);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(32,0.052318,  0.044707);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(33,-0.009125, -0.017719);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(34,0.035121,  0.013527);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(35,-0.005387, -0.017011);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(36,0.025565,  0.005644);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(37,0.031144,  0.007448);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(38,0.01576,   0.027548);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(39,0.012396,  0.026343);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(40,0.008592,  -0.011836);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(41,0.017664,  0.012549);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(42,0.015844,  0.016338);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(43,0.016735,  0.029591);
DROP TABLE PAL_VARMA_MODEL_TBL;
CREATE COLUMN TABLE PAL_VARMA_MODEL_TBL ("INDICES" INTEGER, "CONTENT"
NVARCHAR(5000));
DROP TABLE PAL_VARMA_FIT_TBL;
CREATE COLUMN TABLE PAL_VARMA_FIT_TBL ("colNAME" NVARCHAR(100), "INDICES"
INTEGER, "FITTED" DOUBLE, "RES" DOUBLE);
DROP TABLE PAL_VARMA_IRF_TBL;
CREATE COLUMN TABLE PAL_VARMA_IRF_TBL( "colNAME" NVARCHAR(100), "colNAME2"
NVARCHAR(100), "INDICES" INTEGER, "IRF" DOUBLE);
DROP TABLE PAL_VARMA_CTRL_TBL;
CREATE COLUMN TABLE PAL_VARMA_CTRL_TBL("NAME" NVARCHAR(100), "Intargs" INTEGER,
"Dblargs" DOUBLE, "Strargs" NVARCHAR(100));
INSERT INTO PAL_VARMA_CTRL_TBL VALUES('MODEL',1, NULL, NULL);
INSERT INTO PAL_VARMA_CTRL_TBL VALUES('D', 1, NULL, NULL);
do begin
    lt_data = select * from PAL_VARMA_DATA_TBL;
    lt_control = select * from PAL_VARMA_CTRL_TBL;
    CALL _SYS_AFL.PAL_VARMA(:lt_data, :lt_control, lt_model, lt_fit, lt_irf);
    INSERT INTO PAL_VARMA_MODEL_TBL SELECT * FROM :lt_model;
    select * from :lt_model;
    select * from :lt_fit;
end
```

*Expected Result*

MODEL:

| | CONTENT_INDEX | CONTENT_VALUE |
|---|---|---|
| 1 | 0 | {"model":"VMA"} |
| 2 | 1 | {"exogCols":null} |
| 3 | 2 | {"endogCols":["Y1","Y2"]} |
| 4 | 3 | {"D":0,"d":1,"k":2,"nT":42,"q":1,"s":0} |
| 5 | 4 | {"AIC":6.5048015631382796} |
| 6 | 5 | {"vma":[0.38067602152227514,-1.0529641504662395,-0.9969487694357065... |
| 7 | 6 | {"covariance":[897277.79480768391,1444377.9648685637,1444377.9648685... |
| 8 | 7 | {"residual":[3082.9089173079656,4962.6545766266081]} |
| 9 | 8 | {"savePoint":43} |

FIT:

| | NAMECOL | IDX | FITTING | RESIDUAL |
|---|---|---|---|---|
| 1 | Y1 | 1 | ? | ? |
| 2 | Y1 | 2 | 0.058011 | -0.02877 |
| 3 | Y1 | 3 | 0.0125... | 0.01333... |
| 4 | Y1 | 4 | 0.0552... | -0.04958... |
| 5 | Y1 | 5 | -0.0560... | 0.07526... |
| 6 | Y1 | 6 | 0.1500... | -0.17468... |
| 7 | Y1 | 7 | -0.2423... | 0.27805... |
| 8 | Y1 | 8 | 0.3627... | -0.33866... |
| 9 | Y1 | 9 | -0.4331... | 0.41701... |
| 10 | Y1 | 10 | 0.5539... | -0.52582... |
| 11 | Y1 | 11 | -0.6740... | 0.72269... |
| 12 | Y1 | 12 | 0.9742... | -0.96925... |
| 13 | Y1 | 13 | -1.2694... | 1.22416... |
| 14 | Y1 | 14 | 1.6220... | -1.59839... |
| 15 | Y1 | 15 | -2.0884... | 2.08459... |
| 16 | Y1 | 16 | 2.7487... | -2.80999... |
| 17 | Y1 | 17 | -3.7005... | 3.62170... |
| 18 | Y1 | 18 | 4.7140... | -4.79732... |
| 19 | Y1 | 19 | -6.3252... | 6.39334... |
| 20 | Y1 | 20 | 8.4223... | -8.40169... |
| 21 | Y1 | 21 | -11.021... | 10.9761... |
| 22 | Y1 | 22 | 14.407... | -14.3948... |
| 23 | Y1 | 23 | -18.933... | 19.0333... |
| 24 | Y1 | 24 | 25.090... | -25.0712... |
| 25 | Y1 | 25 | -32.961... | 33.0067... |
| 26 | Y1 | 26 | 43.431... | -43.4557... |

### Example 3: VARMA

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_VARMA_DATA_TBL;
CREATE COLUMN TABLE PAL_VARMA_DATA_TBL("timestamp" INTEGER, "Y1" DOUBLE, "Y2"
DOUBLE);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(1  ,42, 79);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(2  ,35, 89.6);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(3  ,44, 124.8);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(4  ,52, 164.2);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(5  ,58, 172.7);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(6  ,48, 154.2);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(7  ,62, 151.3);
```

```
INSERT INTO PAL_VARMA_DATA_TBL VALUES(8  ,56, 144);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(9  ,49, 143.7);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(10 ,44, 165.3);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(11 ,39, 119.3);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(12 ,31, 95);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(13 ,39, 97.9);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(14 ,46, 101.3);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(15 ,53, 129.1);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(16 ,49, 147.1);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(17 ,52, 152.8);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(18 ,53, 157.2);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(19 ,54, 140.6);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(20 ,56, 138.3);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(21 ,48, 119.8);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(22 ,45, 141.2);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(23 ,37, 110.6);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(24 ,33, 93);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(25 ,38, 81.7);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(26 ,44, 80.9);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(27 ,53, 119.9);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(28 ,49, 148.6);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(29 ,54, 153.3);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(30 ,57, 151.8);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(31 ,51, 139.1);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(32 ,58, 128.3);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(33 ,48, 124.6);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(34 ,44, 133.1);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(35 ,42, 110.5);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(36 ,37, 101.1);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(37 ,42, 79.4);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(38 ,43, 76.2);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(39 ,53, 118.1);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(40 ,49, 140.9);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(41 ,49, 130);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(42 ,40, 120.6);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(43 ,40, 99.2);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(44 ,36, 101.8);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(45 ,29, 89.1);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(46 ,31, 76.6);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(47 ,26, 72.8);
INSERT INTO PAL_VARMA_DATA_TBL VALUES(48 ,23, 60.2);
DROP TABLE PAL_VARMA_MODEL_TBL;
CREATE COLUMN TABLE PAL_VARMA_MODEL_TBL ("INDICES" INTEGER, "CONTENT"
NVARCHAR(5000));
DROP TABLE PAL_VARMA_FIT_TBL;
CREATE COLUMN TABLE PAL_VARMA_FIT_TBL ("colNAME" NVARCHAR(100), "INDICES"
INTEGER, "FITTED" DOUBLE, "RES" DOUBLE);
DROP TABLE PAL_VARMA_IRF_TBL;
CREATE COLUMN TABLE PAL_VARMA_IRF_TBL( "colNAME" NVARCHAR(100), "colNAME2"
NVARCHAR(100), "INDICES" INTEGER, "IRF" DOUBLE);
DROP TABLE PAL_VARMA_CTRL_TBL;
CREATE COLUMN TABLE PAL_VARMA_CTRL_TBL("NAME" NVARCHAR(100), "Intargs" INTEGER,
"Dblargs" DOUBLE, "Strargs" NVARCHAR(100));
INSERT INTO PAL_VARMA_CTRL_TBL VALUES('MODEL',2, NULL, NULL);
INSERT INTO PAL_VARMA_CTRL_TBL VALUES('SEASONAL_PERIOD', 12, NULL, NULL);
INSERT INTO PAL_VARMA_CTRL_TBL VALUES('D', 1, NULL, NULL);
INSERT INTO PAL_VARMA_CTRL_TBL VALUES('SEASONAL_D', 1, NULL, NULL);
INSERT INTO PAL_VARMA_CTRL_TBL VALUES('INITIAL_GUESS', 0, NULL, NULL);
INSERT INTO PAL_VARMA_CTRL_TBL VALUES('SEARCH_METHOD', 1, NULL, NULL);
do begin
    lt_data = select * from PAL_VARMA_DATA_TBL;
    lt_control = select * from PAL_VARMA_CTRL_TBL;
    CALL _SYS_AFL.PAL_VARMA(:lt_data, :lt_control, lt_model, lt_fit, lt_irf);
    INSERT INTO PAL_VARMA_MODEL_TBL SELECT * FROM :lt_model;
    select * from :lt_model;
    select * from :lt_fit;
end
```

*Expected Result*

MODEL:

| | CONTENT_INDEX | CONTENT_VALUE |
|---|---|---|
| 1 | 0 | {"model":"VARMA"} |
| 2 | 1 | {"exogCols":null} |
| 3 | 2 | {"endogCols":["Y1","Y2"]} |
| 4 | 3 | {"D":1,"P":1,"Q":0,"c":0,"d":1,"k":2,"nT":35,"p":2,"q":2,"s":12} |
| 5 | 4 | {"AIC":5.8073537732506226} |
| 6 | 5 | {"var":[-0.21604754742201762,-0.083680453817106759,-0.047541318418001709,-0.3... |
| 7 | 6 | {"svar":[-0.29285236382873098,0.24922282391268516,-0.1538995429754392,-0.1300... |
| 8 | 7 | {"vma":[-1.2615168842378517,0.63224989576544344,1.4366102098968756,0.1717958... |
| 9 | 8 | {"covariance":[4.8931118154524942,9.0907984175327314,9.0907984175327314,55.29... |
| 10 | 9 | {"ytHistory":[45.0,37.0,33.0,38.0,44.0,53.0,49.0,54.0,57.0,51.0,58.0,48.0,44.0,42.0,37.0,42.0... |
| 11 | 10 | {"residual":[-0.91302872465509943,-1.838517255398711,-2.8621053933257059,-12.5... |
| 12 | 11 | {"savePoint":48} |

FIT:

| | NAMECOL | IDX | FITTING | RESIDUAL |
|---|---|---|---|---|
| 1 | Y1 | 1 | ? | ? |
| 2 | Y1 | 2 | ? | ? |
| 3 | Y1 | 3 | ? | ? |
| 4 | Y1 | 4 | ? | ? |
| 5 | Y1 | 5 | ? | ? |
| 6 | Y1 | 6 | ? | ? |
| 7 | Y1 | 7 | ? | ? |
| 8 | Y1 | 8 | ? | ? |
| 9 | Y1 | 9 | ? | ? |
| 10 | Y1 | 10 | ? | ? |
| 11 | Y1 | 11 | ? | ? |
| 12 | Y1 | 12 | ? | ? |
| 13 | Y1 | 13 | ? | ? |
| 14 | Y1 | 14 | 32 | 14 |
| 15 | Y1 | 15 | 51.03187627301236 | 1.9681237269876437 |
| 16 | Y1 | 16 | 59.250493107505875 | -10.250493107505873 |
| 17 | Y1 | 17 | 59.61073960474649 | -7.61073960474649 |
| 18 | Y1 | 18 | 46.82361166017079 | 6.176388339829208 |
| 19 | Y1 | 19 | 64.2678285919938 | -10.267828591993798 |
| 20 | Y1 | 20 | 48.517710919031146 | 7.482289080968853 |
| 21 | Y1 | 21 | 45.157716128176176 | 2.842283871823822 |
| 22 | Y1 | 22 | 40.62845693139659 | 4.371543068603406 |
| 23 | Y1 | 23 | 39.377268265664846 | -2.377268265664844 |

## _SYS_AFL.PAL_VARMA_FORECAST

This function makes time series forecast based on the estimated VAR/VMA/VARMA model.

**Overview of All Tables**

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<MODEL table>` | IN |

| Position | Table Name | Parameter Type |
|---|---|---|
| 3 | `<PARAMETER table>` | IN |
| 4 | `<FORECAST table>` | OUT |

> **Note**
>
> The first input table (data table) can be empty, which means no exogenous variable involved.

### Input Tables

| Table | Column | Reference for Output Table | Data Type | Description |
|---|---|---|---|---|
| DATA | 1st column | ID | INTEGER | Time stamp |
| | Other columns | | INTEGER, DOUBLE, or DECIMAL(p, s) | Exogenous data |
| MODEL | 1st column | | INTEGER | INDEX |
| | 2nd column | | VARCHAR or NVARCHAR | Model content |

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

The following parameter is optional. If it is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| FORECAST_LENGTH | INTEGER | 1 | Number of steps of forecasts. | Valid only when the first input table is empty. |

### Output Table

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| RESULT | 1st column | NVARCHAR (5000) | COLNAME | Column name of endogenous variables |
| | 2nd column | INTEGER | ID (in input table) column name | Time stamp |

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| | 3rd column | DOUBLE | FORECAST | Forecast value |
| | 4th column | DOUBLE | SE | Standard error |
| | 5th column | DOUBLE | LO95 | 95% lower bound |
| | 6th column | DOUBLE | HI95 | 95% upper bound |

### Examples

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

### Example 1: VARX forecast

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_VARMA_PDATA_TBL;
CREATE COLUMN TABLE PAL_VARMA_PDATA_TBL("timestamp" INTEGER, "X" DOUBLE);
INSERT INTO PAL_VARMA_PDATA_TBL VALUES(41,5.2);
INSERT INTO PAL_VARMA_PDATA_TBL VALUES(42,5.2);
INSERT INTO PAL_VARMA_PDATA_TBL VALUES(43,5.2);
INSERT INTO PAL_VARMA_PDATA_TBL VALUES(44,5.2);
INSERT INTO PAL_VARMA_PDATA_TBL VALUES(45,5.7);
DROP TABLE PAL_VARMA_RES_TBL;
CREATE COLUMN TABLE PAL_VARMA_RES_TBL("COLNAME" NVARCHAR(100), "IDX" INTEGER,
"RES" DOUBLE, "SE" DOUBLE, "LO95" DOUBLE, "HI95" DOUBLE);
do begin
   DECLARE lt_control TABLE("NAME" NVARCHAR(100), "Intargs" INTEGER, "Dblargs"
DOUBLE, "Strargs" NVARCHAR(100));
   lt_data = select * from PAL_VARMA_PDATA_TBL;
   lt_model = SELECT * FROM PAL_VARMA_MODEL_TBL;
   CALL _SYS_AFL.PAL_VARMA_FORECAST(:lt_data, :lt_model, :lt_control, lt_res);
   INSERT INTO PAL_VARMA_RES_TBL SELECT * FROM :lt_res;
   select * from :lt_res;
end;
```

### Expected Result

RESULT:

| | COLNAME | IDX | FORECAST | SE | LO95 | HI95 |
|---|---|---|---|---|---|---|
| 1 | Y1 | 41 | 7.577883028526753 | 0.17235243949091225 | 7.240072247124566 | 7.915693809928941 |
| 2 | Y1 | 42 | 7.2027588621793015 | 0.2334209937924623 | 6.745253714346076 | 7.660264010012527 |
| 3 | Y1 | 43 | 7.074507484409191 | 0.27935793836183875 | 6.5269659252199... | 7.622049043598395 |
| 4 | Y1 | 44 | 6.856649822979443 | 0.31664050952252276 | 6.2360344243152... | 7.477265221643587 |
| 5 | Y1 | 45 | 6.773184527727023 | 0.3479969778702801 | 6.091110451101273 | 7.455258604352772 |
| 6 | Y2 | 41 | 5.82295315359521 | 0.17175161524823737 | 5.486319987708665 | 6.159586319481756 |
| 7 | Y2 | 42 | 5.83750232633883 | 0.21681696965320055 | 5.412541065818557 | 6.262463586859104 |
| 8 | Y2 | 43 | 5.577919978010712 | 0.24924316038005895 | 5.089403383665797 | 6.066436572355627 |
| 9 | Y2 | 44 | 5.395542917012289 | 0.2757313982930155 | 4.855109376357978 | 5.9359764576666 |
| 10 | Y2 | 45 | 5.141597731257795 | 0.2982985360267413 | 4.556932600645382 | 5.726262861870207 |

### Example 2: VMA forecast

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_VARMA_PDATA_TBL;
CREATE COLUMN TABLE PAL_VARMA_PDATA_TBL("timestamp" INTEGER);
DROP TABLE PAL_VARMA_RES_TBL;
CREATE COLUMN TABLE PAL_VARMA_RES_TBL("COLNAME" NVARCHAR(100), "IDX" INTEGER,
"RES" DOUBLE, "SE" DOUBLE, "LO95" DOUBLE, "HI95" DOUBLE);
do begin
   lt_data = select * from PAL_VARMA_PDATA_TBL;
   lt_control = SELECT TO_NVARCHAR('FORECAST_LENGTH') AS name, TO_INTEGER(10)
AS intargs, TO_DOUBLE(NULL) AS doubleargs, TO_NVARCHAR(NULL) AS stringargs FROM
DUMMY;
   lt_model = SELECT * FROM PAL_VARMA_MODEL_TBL;
   CALL _SYS_AFL.PAL_VARMA_FORECAST(:lt_data, :lt_model, :lt_control, lt_res);
   INSERT INTO PAL_VARMA_RES_TBL SELECT * FROM :lt_res;
   select * from :lt_res;
end;
```

### Expected Result

RESULT:

| | COLNAME | IDX | FORECAST | SE | LO95 | HI95 |
|---|---|---|---|---|---|---|
| 1 | Y1 | 0 | 4,051.9078589786905 | 947.2474834000267 | 2,195.302791514638 | 5,908.512926442743 |
| 2 | Y1 | 1 | 0 | 947.2474834000267 | -1,856.6050674640524 | 1,856.6050674640524 |
| 3 | Y1 | 2 | 0 | ? | ? | ? |
| 4 | Y1 | 3 | 0 | ? | ? | ? |
| 5 | Y1 | 4 | 0 | ? | ? | ? |
| 6 | Y1 | 5 | 0 | ? | ? | ? |
| 7 | Y1 | 6 | 0 | ? | ? | ? |
| 8 | Y1 | 7 | 0 | ? | ? | ? |
| 9 | Y1 | 8 | 0 | ? | ? | ? |
| 10 | Y1 | 9 | 0 | ? | ? | ? |
| 11 | Y2 | 0 | 6,522.495805903587 | 1,524.8158378784... | 3,533.8567636617395 | 9,511.134848145435 |
| 12 | Y2 | 1 | 0 | 1,524.8158378784... | -2,988.6390422418476 | 2,988.6390422418476 |
| 13 | Y2 | 2 | 0 | ? | ? | ? |
| 14 | Y2 | 3 | 0 | ? | ? | ? |
| 15 | Y2 | 4 | 0 | ? | ? | ? |
| 16 | Y2 | 5 | 0 | ? | ? | ? |
| 17 | Y2 | 6 | 0 | ? | ? | ? |
| 18 | Y2 | 7 | 0 | ? | ? | ? |
| 19 | Y2 | 8 | 0 | ? | ? | ? |
| 20 | Y2 | 9 | 0 | ? | ? | ? |

### Example 3: VARMA forecast

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_VARMA_PDATA_TBL;
CREATE COLUMN TABLE PAL_VARMA_PDATA_TBL("timestamp" INTEGER);
DROP TABLE PAL_VARMA_RES_TBL;
CREATE COLUMN TABLE PAL_VARMA_RES_TBL("COLNAME" NVARCHAR(100), "IDX" INTEGER,
"RES" DOUBLE, "SE" DOUBLE, "LO95" DOUBLE, "HI95" DOUBLE);
do begin
   lt_data = select * from PAL_VARMA_PDATA_TBL;
```

```
   lt_control = SELECT TO_NVARCHAR('FORECAST_LENGTH') AS name, TO_INTEGER(10)
AS intargs, TO_DOUBLE(NULL) AS doubleargs, TO_NVARCHAR(NULL) AS stringargs FROM
DUMMY;
   lt_model = SELECT * FROM PAL_VARMA_MODEL_TBL;
   CALL _SYS_AFL.PAL_VARMA_FORECAST(:lt_data, :lt_model, :lt_control, lt_res);
   INSERT INTO PAL_VARMA_RES_TBL SELECT * FROM :lt_res;
   select * from :lt_res;
end;
```

**Expected Result**

RESULT:

| | COLNAME | IDX | FORECAST | SE | LO95 | HI95 |
|---|---|---|---|---|---|---|
| 1 | Y1 | 0 | 25.73879688450897 | 2.212037932643221 | 21.40320253652826 | 30.074391232489685 |
| 2 | Y1 | 1 | 29.990008240627766 | 5.073678049285382 | 20.045599264028418 | 39.934417217227114 |
| 3 | Y1 | 2 | 44.39705373289216 | 10.642208988397... | 23.53832411563238 | 65.25578335015194 |
| 4 | Y1 | 3 | 39.27934546443837 | 10.690369274001... | 18.32622168739456 | 60.23246924148218 |
| 5 | Y1 | 4 | 31.965570804263074 | 11.362319402528... | 9.695424775307902 | 54.23571683321825 |
| 6 | Y1 | 5 | 21.54744976111207 | 11.714633576746... | -1.41323204931192... | 44.50813157153606 |
| 7 | Y1 | 6 | 18.427838757097646 | 14.612466961979... | -10.2125964883827... | 47.068274002578 |
| 8 | Y1 | 7 | 26.293017035090436 | 15.433385637666... | -3.95641881473578... | 56.542452884916656 |
| 9 | Y1 | 8 | 13.76992325160704 | 15.670096705692... | -16.9434662915493... | 44.48331279476341 |
| 10 | Y1 | 9 | 6.342479220751837 | 15.89472077134253 | -24.8111734910795... | 37.4961319325832 |
| 11 | Y2 | 0 | 17.36428858143141 | 7.435797929183921 | 2.7901246402309248 | 31.938452522631895 |
| 12 | Y2 | 1 | 10.239734892582021 | 18.40017391753896 | -25.8246059857943... | 46.304075770958384 |
| 13 | Y2 | 2 | 61.544162304418165 | 18.938285610574... | 24.42512250769215 | 98.66320210114418 |
| 14 | Y2 | 3 | 92.25950653752 | 21.080538889275... | 50.94165031454087 | 133.5773627604991 |
| 15 | Y2 | 4 | 78.37269838878133 | 21.378447665282... | 36.47094096482666 | 120.274455812736 |
| 16 | Y2 | 5 | 60.50679347317791 | 24.38861010029906 | 12.705117676591748 | 108.30846926976406 |
| 17 | Y2 | 6 | 37.61797417164027 | 27.461980986321... | -16.2075085615501... | 91.4434569048307 |
| 18 | Y2 | 7 | 43.32049547567473 | 27.722493181255... | -11.0155911595856... | 97.65658211093513 |
| 19 | Y2 | 8 | 39.34371276839906 | 29.16945623299477 | -17.82842144827069 | 96.5158469850688 |
| 20 | Y2 | 9 | 25.755989830435436 | 29.280498514918... | -31.6337872588052... | 83.14576691967613 |

## 5.5.34 White Noise Test

This algorithm is used to identify whether a time series is a white noise series. If white noise exists in the raw time series, the algorithm returns the value of 1. If not, the value of 0 will be returned.

PAL uses Ljung-Box test to test for autocorrelation at different lags. The Ljung-Box test can be defined as follows:

$H_0$: White noise exists in the time series ($\rho_1=\rho_2=\rho_3=...=\rho_m=0$).

$H_1$: White noise does not exists in the time series.

Where $\rho_i$ is the ACF at lag `i`, and `m` is the user-defined maximum lag.

The Ljung-Box test statistic is given by the following formula:

$$Q = n(n+2)\sum_{h=1}^{m}\frac{\hat{\rho}_h^2}{n-h}$$

Where n is the sample size, $\hat{\rho}_h$ is the sample autocorrelation at lag h, and m is the number of lags being tested. The statistic of Q follows a chi-square distribution. Based on the significance level α, the critical region for rejection of the hypothesis of randomness is

$$Q > x^2_{1-\alpha,m}$$

Where $x^2_{1-\alpha,m}$ is the chi-squared distribution with m degrees of freedom and α quantile.

## Prerequisites

- No null data in the inputs. The time periods should be unique.

## _SYS_AFL.PAL_WN_TEST

This function identifies the white noise of a time series.

### Overview of all Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<STATISTICS table>` | OUT |

### Input Table

| Table | Column | Data Type | Description |
|---|---|---|---|
| DATA | 1st column | INTEGER | Time periods.<br>Time periods do not need to be in order, but must be unique. |
| | 2nd column | INTEGER, DOUBLE, or DECIMAL(p, s) | Time series. |

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| LAG | INTEGER | Half of the sample size (n/2) | Specifies the lag autocorrelation coefficient that the statistic will be based on. It corresponds to the degree of freedom of chi-square distribution. |
| THREAD_RATIO | DOUBLE | -1 | The ratio of available threads.<br>• 0: single thread<br>• 0~1: percentage<br>• Others: heuristically determined |
| PROBABILITY | DOUBLE | 0.9 | The confidence level used for chi-square distribution. The value is $1 - \alpha$, where $\alpha$ is the significance level. |
| MODEL_DF | INTEGER | 0 | Degrees of freedom consumed by the data. |

### Output Table

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| STATISTICS | 1st column | NVARCHAR(100) | STAT_NAME | Name of the statistics of the series. |
| | 2nd column | DOUBLE | STAT_VALUE | Values.<br>• WN: 1 for white noise, 0 for not white noise<br>• Q: Q statistics defined as above<br>• chi^2: chi-square distribution |

### Example

Assume that:

- DM_PAL is a schema belonging to USER1;
- USER1 has been assigned the AFL__SYS_AFL_AFLPAL_EXECUTE or AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION role.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_WN_DATA_TBL;
CREATE COLUMN TABLE PAL_WN_DATA_TBL (
    "TIME_STAMP" INTEGER,
    "SERIES" DOUBLE
);
INSERT INTO PAL_WN_DATA_TBL VALUES (0, 1356.00);
INSERT INTO PAL_WN_DATA_TBL VALUES (1, 826.00);
INSERT INTO PAL_WN_DATA_TBL VALUES (2, 1586.00);
INSERT INTO PAL_WN_DATA_TBL VALUES (3, 1010.00);
INSERT INTO PAL_WN_DATA_TBL VALUES (4, 1337.00);
INSERT INTO PAL_WN_DATA_TBL VALUES (5, 1415.00);
INSERT INTO PAL_WN_DATA_TBL VALUES (6, 1514.00);
INSERT INTO PAL_WN_DATA_TBL VALUES (7, 1474.00);
INSERT INTO PAL_WN_DATA_TBL VALUES (8, 1662.00);
INSERT INTO PAL_WN_DATA_TBL VALUES (9, 1805.00);
INSERT INTO PAL_WN_DATA_TBL VALUES (10, 2218.00);
INSERT INTO PAL_WN_DATA_TBL VALUES (11, 2400.00);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR(100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR(100)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 0.2, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PROBABILITY', NULL, 0.9, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('LAG', 3, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MODEL_DF', 1, NULL, NULL);
CALL _SYS_AFL.PAL_WN_TEST(PAL_WN_DATA_TBL, "#PAL_PARAMETER_TBL", ?);
```

### Expected Result

PAL_WHITENOISETEST_RESULT_TBL:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | WN | 1 |
| 2 | Q | 5.576053368671719 |
| 3 | chi^2 | 6.2513885498046... |

## 5.6 Pipeline

Pipeline is used for constructing a composite estimator using a chain of transformers and estimators.

When training a specific algorithm model, such as SVM, HGBT, or MLG, it is often necessary to apply a fixed sequence of steps of pre-processing on the original data, such as feature selection, normalization, or Principal Component Analysis (PCA), to get a model with more accuracy. And the predicted data need to be pre-processed in the same way as the training when using the specific algorithm model to make predictions.

Pipeline can combine these operators into a composite estimator. You can obtain a single estimator model with only one training process and get the predicted result with one model and one predicting process.

## Pipeline Structure

In PAL, we use JSON to define the structure of pipeline.

A single operator is represented as a JSON object with only one member. The key of the member is the name of the operator, and the value is a JSON object containing two members, `args` and `inputs`. The definition format of operator is as follows:

```
{"<name>":{"args":{"arg1": value1, ...},"inputs":{"input1": data1, ...}}}
```

Example

```
{"RDT_Classifier":{"args":
{"TREES_NUM":100,"NODE_SIZE":9,"CALCULATE_OOB":0,"SAMPLE_FRACTION":0.75},"inputs"
:{"data":"ROWDATA"}}},
{"FS_unsupervised":{"args":{"FS_METHOD":9, "TOP_K_BEST":10}, "inputs":
{"data":"ROWDATA"}}},
{"SCALE":{"args":{"SCALING_METHOD": 1, "Z-SCORE_METHOD":2}, "inputs":
{"data":"ROWDATA"}}}
```

See Pipeline Operator [page 988] for more details.

A pipeline can have only one estimator, which means that the input data will not go through any pre-processing. We can replace the value of data in inputs with a transformer to construct a composite estimator. In the same way, by replacing the data value of the transformer, a more complex pipeline can be constructed.

- Example

```
{"RDT_Classifier":{"args":{...},"inputs":{"data":"ROWDATA"}}},
{"RDT_Classifier":{"args":{...},"inputs":{"data":{"CATPCA":{"args":
{...},"inputs":{"data":"ROWDATA"}}}}}},
{"RDT_Classifier":{"args":{...},"inputs":{"data":{"CATPCA":{"args":
{...},"inputs":{"data":{"SCALE":{"args":{...}, "inputs":
{"data":"ROWDATA"}}}}}}}}
```

## _SYS_AFL.PAL_PIPELINE_FIT

This procedure is used to fit PAL pipeline model.

### Overview of All Tables

| Position | Table Name | Type |
|---|---|---|
| 1 | `<Data Table>` | IN |
| 2 | `<Parameter Table>` | IN |
| 3 | `<Model Table>` | OUT |

| Position | Table Name | Type |
|---|---|---|
| 4 | `<Statistic Table>` | OUT |

### Input Table

- For classification and regression

| Table | Column | Data Type | Description |
|---|---|---|---|
| DATA | 1st Column | INTEGER, BIGINT, VARCHAR, or NVARCHAR | Record ID<br>If this column is absent, the HAS_ID parameter must be set to 0. |
| | Other columns | VARCHAR, NVARCHAR, DOUBLE, DECIMAL (p, s), or INTEGER<br>For classification, dependent variable data type cannot be DOUBLE. | Attribute data and dependent variable. |

- For time series

| Table | Column | Data Type | Description |
|---|---|---|---|
| DATA | 1st column | INTEGER, BIGINT, TIMESTAMP, SECONDDATE, or DATE | Time stamp |
| | 2nd column | INTEGER, DOUBLE, or DECIMAL (p, s) | Target time series data |
| | Other columns | INTEGER, DOUBLE, DECIMAL (p, s), VARCHAR, or NVARCHAR | External data feature |

### Parameter Table

Mandatory Parameters

| Name | Data Type | Description |
|---|---|---|
| PIPELINE | VARCHAR or NVARCHAR | JSON string of pipeline structure. |

Optional Parameters

The following parameter is optional. If it is not specified, PAL will use its default value.

- For classification and regression

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| HAS_ID | INTEGER | 1 | Specifies whether the input data table has an ID column. If it exists, the ID column must be the first column.<br>• 0: No ID column<br>• 1: Has an ID column | |
| DEPENDENT_VARIABLE | VARCHAR or NVARCHAR | The last column name of the Data table | Specifies the variable used as dependent | Cannot be the first column name if HAS_ID is 1. |
| CATEGORICAL_VARIABLE | VARCHAR or NVARCHAR | Detected from input data | Indicates which variables are treated as categorical. The default behavior is:<br>• STRING: categorical<br>• INTEGER and DOUBLE: continuous | Valid only for INTEGER variables; omitted otherwise. |
| BACKGROUND_SIZE | INTEGER | 0 | The row size of background data. | Applicable for PAL_PIPELINE_EXPLAIN. It should not be larger than the row size of train data. |
| BACKGROUND_SAMPLING_SEED | INTEGER | 0 | Specifies the seed for random number generator in the background sampling.<br>• 0: Uses the current time (in second) as seed<br>• Others: Uses the specified value as seed | Applicable for PAL_PIPELINE_EXPLAIN. |

### Output Tables

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| MODEL | 1st column | INTEGER | ROW_INDEX | Pipeline model row index |

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| | 2nd column | VARCHAR or NVARCHAR | MODEL_CONTENT | Deserialized pipeline model.<br>The table must be a column table. The maximum length of every unit (row) is 5000. |
| STATISTICS | 1st column | VARCHAR or NVARCHAR | STAT_NAME | Statistic name |
| | 2nd column | VARCHAR or NVARCHAR | STAT_VALUE | Statistic value |

### Examples

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- `USER1` has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

#### Example: Pipeline Fit with Background Data

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_PIPELINE_DATA_TBL;
CREATE COLUMN TABLE PAL_PIPELINE_DATA_TBL (
    ID INTEGER,
    ATTRIBUTE1 DOUBLE,
    ATTRIBUTE2 DOUBLE,
    ATTRIBUTE3 DOUBLE,
    ATTRIBUTE4 VARCHAR(10),
    LABEL INTEGER
);
INSERT INTO PAL_PIPELINE_DATA_TBL VALUES (0,1,10,100,'A',1);
INSERT INTO PAL_PIPELINE_DATA_TBL VALUES (1,1.1,10.1,100,'A',1);
INSERT INTO PAL_PIPELINE_DATA_TBL VALUES (2,1.2,10.2,100,'A',1);
INSERT INTO PAL_PIPELINE_DATA_TBL VALUES (3,1.3,10.4,100,'A',1);
INSERT INTO PAL_PIPELINE_DATA_TBL VALUES (4,1.2,10.3,100,'AB',1);
INSERT INTO PAL_PIPELINE_DATA_TBL VALUES (5,4,40,400,'AB',2);
INSERT INTO PAL_PIPELINE_DATA_TBL VALUES (6,4.1,40.1,400,'AB',2);
INSERT INTO PAL_PIPELINE_DATA_TBL VALUES (7,4.2,40.2,400,'AB',2);
INSERT INTO PAL_PIPELINE_DATA_TBL VALUES (8,4.3,40.4,400,'AB',2);
INSERT INTO PAL_PIPELINE_DATA_TBL VALUES (9,4.2,40.3,400,'AB',2);
INSERT INTO PAL_PIPELINE_DATA_TBL VALUES (10,9,90,900,'B',3);
INSERT INTO PAL_PIPELINE_DATA_TBL VALUES (11,9.1,90.1,900,'A',3);
INSERT INTO PAL_PIPELINE_DATA_TBL VALUES (12,9.2,90.2,900,'B',3);
INSERT INTO PAL_PIPELINE_DATA_TBL VALUES (13,9.3,90.4,900,'A',3);
INSERT INTO PAL_PIPELINE_DATA_TBL VALUES (14,9.2,90.3,900,'A',3);
DROP TABLE PAL_PARAMETER_TAB;
CREATE COLUMN TABLE PAL_PARAMETER_TAB (
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR(1000)
);
INSERT INTO PAL_PARAMETER_TAB VALUES ('PIPELINE', null,
null, '{"NB_Classifier":{"args":{"LAPLACE":0.1},"inputs":{"data":{"TomekLinks":
{"args":{"SAMPLING_STRATEGY":2,"DISTANCE_LEVEL":2,"METHOD":1},"inputs":
{"data":"ROWDATA"}}}}}}');
INSERT INTO PAL_PARAMETER_TAB VALUES ('HAS_ID', 1, null, null);
```

```
INSERT INTO PAL_PARAMETER_TAB VALUES ('BACKGROUND_SIZE', 10, null, null);
INSERT INTO PAL_PARAMETER_TAB VALUES ('BACKGROUND_SAMPLING_SEED', 1, null, null);
DROP TABLE PAL_PIPELINE_MODEL_TBL;
CREATE COLUMN TABLE PAL_PIPELINE_MODEL_TBL(
    ROW_INDEX INTEGER,
    MODEL_CONTENT VARCHAR(5000)
);
DO BEGIN
    lt_data = select * from PAL_PIPELINE_DATA_TBL;
    lt_para = select * from PAL_PARAMETER_TAB;
    CALL _SYS_AFL.PAL_PIPELINE_FIT(:lt_data, :lt_para, lt_model, lt_stat);
    INSERT INTO PAL_PIPELINE_MODEL_TBL SELECT * FROM :lt_model;
END;
SELECT * FROM PAL_PIPELINE_MODEL_TBL;
```

**Expected Result**

RESULT:

| | ROW_INDEX | |
|---|---|---|
| 1 | 0 | {"pipeline":{"NB_Classifier":{"args":{"LAPLACE":0.1},"model":{"Laplace": |

## _SYS_AFL.PAL_PIPELINE_PREDICT

This procedure uses the pipeline model generated by PAL_PIPELINE_FIT to make predictive analysis.

**Overview of All Tables**

| Position | Table Name | Type |
|---|---|---|
| 1 | `<Data Table>` | IN |
| 3 | `<Model Table>` | IN |
| 2 | `<Parameter Table>` | IN |
| 3 | `<Result Table>` | OUT |
| 4 | `<Statistic Table>` | OUT |

**Data Table**

- For classification and regression

| Table | Column | Data Type | Description |
|---|---|---|---|
| DATA | 1st column | INTEGER, BIGINT, VARCHAR, or NVARCHAR | Record ID |
| | Other columns | VARCHAR, NVARCHAR, DOUBLE, DECIMAL (p, s), or INTEGER | Attribute data to be predicted |

- For time series

| Table | Column | Data Type | Description |
|---|---|---|---|
| DATA | 1st column | INTEGER, BIGINT, TIMESTAMP, SECONDDATE, or DATE | Time stamp |
| | other columns | INTEGER, DOUBLE, DECIMAL (p, s), VARCHAR, or NVARCHAR | External data feature |

### Model Table

| Table | Column | Data Type | Description |
|---|---|---|---|
| MODEL | 1st column | INTEGER | Row index |
| | 2nd column | VARCHAR or NVARCHAR | Saved model |

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

None.

### Output Table

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| RESULT | 1st column | NVARCHAR | ID (in input table) column name | Record ID or time stamp |
| | 2nd column | NVARCHAR | SCORES | Predictive result |
| STATISTICS | 1st column | NVARCHAR | STAT_NAME | Statistic name |
| | 2nd column | NVARCHAR | STAT_VALUE | Statistic value |

### Examples

### Example: Pipeline Predict

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_PIPELINE_PREDICT_DATA_TBL;
CREATE COLUMN TABLE PAL_PIPELINE_PREDICT_DATA_TBL(
    ID INTEGER,
    ATTRIBUTE1 DOUBLE,
    ATTRIBUTE2 DOUBLE,
    ATTRIBUTE3 DOUBLE,
    ATTRIBUTE4 VARCHAR(10)
);
INSERT INTO PAL_PIPELINE_PREDICT_DATA_TBL VALUES(0,1,10,100,'A');
INSERT INTO PAL_PIPELINE_PREDICT_DATA_TBL VALUES(1,1.2,10.2,100,'A');
INSERT INTO PAL_PIPELINE_PREDICT_DATA_TBL VALUES(2,4.1,40.1,400,'AB');
INSERT INTO PAL_PIPELINE_PREDICT_DATA_TBL VALUES(3,4.2,40.3,400,'AB');
INSERT INTO PAL_PIPELINE_PREDICT_DATA_TBL VALUES(4,9.1,90.1,900,'A');
INSERT INTO PAL_PIPELINE_PREDICT_DATA_TBL VALUES(5,9.2,90.2,900,'A');
INSERT INTO PAL_PIPELINE_PREDICT_DATA_TBL VALUES(6,4,40,400,'A');
DROP TABLE PAL_PARAMETER_TAB;
CREATE COLUMN TABLE PAL_PARAMETER_TAB (
```

```
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR(1000)
);
CALL _SYS_AFL.PAL_PIPELINE_PREDICT(PAL_PIPELINE_PREDICT_DATA_TBL,
PAL_PIPELINE_MODEL_TBL, PAL_PARAMETER_TAB, ?, ?);
```

**Expected Result**

RESULT:

Rows (7) SQL

| | ID | SCORES |
|---|---|---|
| 1 | 0 | 1 |
| 2 | 1 | 1 |
| 3 | 2 | 2 |
| 4 | 3 | 2 |
| 5 | 4 | 3 |
| 6 | 5 | 3 |
| 7 | 6 | 2 |

## _SYS_AFL.PAL_PIPELINE_EVALUATE

This procedure is used to evaluate a pipeline structure.

**Overview of All Tables**

| Position | Table Name | Type |
|---|---|---|
| 1 | `<Data Table>` | IN |
| 2 | `<Parameter Table>` | IN |
| 3 | `<Result Table>` | OUT |

**Data Table**

Same requirements as in PAL_PIPELINE_FIT.

**Parameter Table**

Mandatory Parameters

| Name | Data Type | Description |
| --- | --- | --- |
| PIPELINE | VARCHAR or NVARCHAR | JSON string of pipeline structure. |

Optional Parameters

The following parameter is optional. If it is not specified, PAL will use its default value.

- For classification and regression

| Name | Data Type | Default Value | Description | Dependency |
| --- | --- | --- | --- | --- |
| HAS_ID | INTEGER | 1 | Specifies whether the input data table has an ID column. If it exists, the ID column must be the first column.<br>• 0: No ID column<br>• 1: Has an ID column | |
| DEPENDENT_VARIABLE | VARCHAR or NVARCHAR | The last column name of the Data table | Specifies the variable used as dependent | Cannot be the first column name if HAS_ID is 1. |
| CATEGORICAL_VARIABLE | VARCHAR or NVARCHAR | Detected from input data | Indicates which variables are treated as categorical. The default behavior is:<br>• STRING: categorical<br>• INTEGER and DOUBLE: continuous | Valid only for INTEGER variables; omitted otherwise. |
| RESAMPLING_METHOD | VARCHAR or NVARCHAR | • "stratified_cv" for classification<br>• "cv" for Regression | Specifies the resampling method for pipeline evaluation.<br>• cv<br>• stratified_cv | |
| FOLD_NUM | INTEGER | 5 | Specifies the fold number for the cross validation method. | |
| RANDOM_SEED | INTEGER | Empty value | Specifies the seed for random number generator. Uses system time if not provided. | |

- For time series

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| SPLIT_METHOD | INTEGER | 1 | Specifies the split method for pipeline evaluation.<br>• 1:ROCV<br>• 2:Blocked | |
| FOLD_NUM | INTEGER | 5 | Specifies the fold number for the cross validation method. | |
| GAP_NUM | INTEGER | 0 | Number of samples to exclude from the end of each train set before the test set. | |
| PERCENTAGE | DOUBLE | 0.7 | Percentage between training data and test data. Only applicable to blocked type. | Only valid when SPLIT_METHOD is 2. |

### Output Table

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| RESULT | 1st column | STRING | NAME | Score name |
| | 2nd column | DOUBLE | VALUE | Score Value |

### Examples

### Example: Pipeline Evaluate

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_PIPELINE_DATA_TBL;
CREATE COLUMN TABLE PAL_PIPELINE_DATA_TBL (
    ID INTEGER,
    ATTRIBUTE1 DOUBLE,
    ATTRIBUTE2 DOUBLE,
    ATTRIBUTE3 DOUBLE,
    ATTRIBUTE4 VARCHAR(10),
    LABEL INTEGER
);
INSERT INTO PAL_PIPELINE_DATA_TBL VALUES (0,1,10,100,'A',1);
INSERT INTO PAL_PIPELINE_DATA_TBL VALUES (1,1.1,10.1,100,'A',1);
INSERT INTO PAL_PIPELINE_DATA_TBL VALUES (2,1.2,10.2,100,'A',1);
INSERT INTO PAL_PIPELINE_DATA_TBL VALUES (3,1.3,10.4,100,'A',1);
INSERT INTO PAL_PIPELINE_DATA_TBL VALUES (4,1.2,10.3,100,'AB',1);
INSERT INTO PAL_PIPELINE_DATA_TBL VALUES (5,4,40,400,'AB',2);
INSERT INTO PAL_PIPELINE_DATA_TBL VALUES (6,4.1,40.1,400,'AB',2);
INSERT INTO PAL_PIPELINE_DATA_TBL VALUES (7,4.2,40.2,400,'AB',2);
INSERT INTO PAL_PIPELINE_DATA_TBL VALUES (8,4.3,40.4,400,'AB',2);
INSERT INTO PAL_PIPELINE_DATA_TBL VALUES (9,4.2,40.3,400,'AB',2);
INSERT INTO PAL_PIPELINE_DATA_TBL VALUES (10,9,90,900,'B',3);
INSERT INTO PAL_PIPELINE_DATA_TBL VALUES (11,9.1,90.1,900,'A',3);
INSERT INTO PAL_PIPELINE_DATA_TBL VALUES (12,9.2,90.2,900,'B',3);
INSERT INTO PAL_PIPELINE_DATA_TBL VALUES (13,9.3,90.4,900,'A',3);
INSERT INTO PAL_PIPELINE_DATA_TBL VALUES (14,9.2,90.3,900,'A',3);
INSERT INTO PAL_PIPELINE_DATA_TBL VALUES (15,1,10,100,'A',1);
INSERT INTO PAL_PIPELINE_DATA_TBL VALUES (16,1.2,10.2,100,'A',1);
INSERT INTO PAL_PIPELINE_DATA_TBL VALUES (17,4.1,40.1,400,'AB',2);
```

```
INSERT INTO PAL_PIPELINE_DATA_TBL VALUES (18,4.2,40.3,400,'AB',2);
INSERT INTO PAL_PIPELINE_DATA_TBL VALUES (19,9.1,90.1,900,'A',3);
INSERT INTO PAL_PIPELINE_DATA_TBL VALUES (20,9.2,90.2,900,'A',3);
INSERT INTO PAL_PIPELINE_DATA_TBL VALUES (21,4,40,400,'A',2);
DROP TABLE PAL_PARAMETER_TAB;
CREATE COLUMN TABLE PAL_PARAMETER_TAB (
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR(1000)
);
INSERT INTO PAL_PARAMETER_TAB VALUES ('PIPELINE', null,
null, '{"NB_Classifier":{"args":{"LAPLACE":0.1},"inputs":{"data":{"TomekLinks":
{"args":{"SAMPLING_STRATEGY":2,"DISTANCE_LEVEL":2,"METHOD":1},"inputs":
{"data":"ROWDATA"}}}}}}');
INSERT INTO PAL_PARAMETER_TAB VALUES ('HAS_ID', 1, null, null);
INSERT INTO PAL_PARAMETER_TAB VALUES ('FOLD_NUM', 5, null, null);
INSERT INTO PAL_PARAMETER_TAB VALUES ('RANDOM_SEED', 1, null, null);
CALL _SYS_AFL.PAL_PIPELINE_EVALUATE(PAL_PIPELINE_DATA_TBL, PAL_PARAMETER_TAB, ?);
```

**Expected Result**

RESULT:

Rows (5) SQL

| | NAME | EVALUATE |
|---|---|---|
| 1 | ACCURACY | 1 |
| 2 | AUC | 1 |
| 3 | KAPPA | 1 |
| 4 | LAYERS | 2 |
| 5 | MCC | 1 |

## _SYS_AFL.PAL_PIPELINE_EXPLAIN

This procedure uses the pipeline model generated by PAL_PIPELINE_FIT to make predictive analysis. To make explanation, BACKGROUND_SIZE in PAL_PIPELINE_FIT should be larger than 0.

**Overview of All Tables**

| Position | Table Name | Type |
|---|---|---|
| 1 | <Data Table> | IN |
| 3 | <Model Table> | IN |

| Position | Table Name | Type |
|---|---|---|
| 2 | <Parameter Table> | IN |
| 3 | <Result Table> | OUT |
| 4 | <Statistic Table> | OUT |

### Data Table

| Table | Column | Data Type | Description |
|---|---|---|---|
| DATA | 1st column | INTEGER, BIGINT, VARCHAR, or NVARCHAR | Record ID |
| | other columns | INTEGER, VARCHAR, NVARCHAR, DOUBLE, or DECIMAL(p, s) | Attribute data to be predicted |

### Model Table

| Table | Column | Data Type | Description |
|---|---|---|---|
| MODEL | 1st column | INTEGER | Row index |
| | 2nd column | VARCHAR or NVARCHAR | Saved model |

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If it is not specified, PAL will use its default value.

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| TOP_K_ATTRIBUTIONS | INTEGER | 10 | Outputs the attributions of top k features which contribute the most. | |
| SAMPLESIZE | INTEGER | 0 | Number of sampled combinations of features. If it equals to 0, it is determined by algorithm heuristically. | It is better to use a number larger than the number of columns of the data table. |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| SEED | INTEGER | 0 | Specifies the seed for random number generator.<br>• 0: Uses the current time (in second) as seed<br>• Others: Uses the specified value as seed | |
| VERBOSE_OUTPUT | INTEGER | 0 | Specifies whether to output all classes and the corresponding confidences for each data.<br>• 0: No<br>• 1: Yes | Valid only for classification. |

### Output Table

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| RESULT | 1st Column | NVARCHAR | ID (in the input table) column name | Record ID |
| | 2nd Column | NVARCHAR | SCORE | Predictive result |
| | 3rd Column | DOUBLE | CONFIDENCE | Confidence of a class |
| | 4th Column | NCLOB | REASON_CODE | Attributions of features |
| | 5th Column | NVARCHAR | PLACEHOLDER_1 | Placeholder |
| | 6th Column | NVARCHAR | PLACEHOLDER_2 | Placeholder |
| STATISTICS | 1st Column | NVARCHAR | STAT_NAME | Statistic name |
| | 2nd Column | NVARCHAR | STAT_VALUE | Statistic value |

### Examples

Assume that:

- `DM_PAL` is a schema belonging to `USER1`;
- `USER1` has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

### Example: Pipeline Explain

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_PIPELINE_PREDICT_DATA_TBL;
CREATE COLUMN TABLE PAL_PIPELINE_PREDICT_DATA_TBL(
    ID INTEGER,
    ATTRIBUTE1 DOUBLE,
    ATTRIBUTE2 DOUBLE,
    ATTRIBUTE3 DOUBLE,
    ATTRIBUTE4 VARCHAR(10)
```

```
);
INSERT INTO PAL_PIPELINE_PREDICT_DATA_TBL VALUES(0,1,10,100,'A');
INSERT INTO PAL_PIPELINE_PREDICT_DATA_TBL VALUES(1,1.2,10.2,100,'A');
INSERT INTO PAL_PIPELINE_PREDICT_DATA_TBL VALUES(2,4.1,40.1,400,'AB');
INSERT INTO PAL_PIPELINE_PREDICT_DATA_TBL VALUES(3,4.2,40.3,400,'AB');
INSERT INTO PAL_PIPELINE_PREDICT_DATA_TBL VALUES(4,9.1,90.1,900,'A');
INSERT INTO PAL_PIPELINE_PREDICT_DATA_TBL VALUES(5,9.2,90.2,900,'A');
INSERT INTO PAL_PIPELINE_PREDICT_DATA_TBL VALUES(6,4,40,400,'A');
DROP TABLE PAL_PARAMETER_TAB;
CREATE COLUMN TABLE PAL_PARAMETER_TAB (
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR(1000)
);
INSERT INTO PAL_PARAMETER_TAB VALUES ('TOP_K_ATTRIBUTIONS', 10, null, null);
INSERT INTO PAL_PARAMETER_TAB VALUES ('SEED', 1, null, null);
INSERT INTO PAL_PARAMETER_TAB VALUES ('VERBOSE_OUTPUT', 0, null, null);
CALL _SYS_AFL.PAL_PIPELINE_EXPLAIN(PAL_PIPELINE_PREDICT_DATA_TBL,
PAL_PIPELINE_MODEL_TBL, PAL_PARAMETER_TAB, ?, ?);
```

**Expected Result**

RESULT:

| | ID | SCORE | CONFIDENCE | |
|---|---|---|---|---|
| 1 | 0 | 1 | 1 | [{"attr":"ATTRIBUTE2","val":0.1700000000000001,"pct":43.6},{"attr":"ATTRI |
| 2 | 1 | 1 | 1 | [{"attr":"ATTRIBUTE2","val":0.1700000000000001,"pct":43.6},{"attr":"ATTRI |
| 3 | 2 | 2 | 1 | [{"attr":"ATTRIBUTE2","val":0.31000000000000019,"pct":46.30000000000 |
| 4 | 3 | 2 | 1 | [{"attr":"ATTRIBUTE2","val":0.31000000000000019,"pct":46.30000000000 |
| 5 | 4 | 3 | 1 | [{"attr":"ATTRIBUTE2","val":0.2700000000000001,"pct":27.3},{"attr":"ATTRI |
| 6 | 5 | 3 | 1 | [{"attr":"ATTRIBUTE2","val":0.2700000000000001,"pct":27.3},{"attr":"ATTRI |
| 7 | 6 | 2 | 1 | [{"attr":"ATTRIBUTE2","val":0.31000000000000019,"pct":46.30000000000 |

## 5.6.1 Pipeline Operator

Most of the operators are adapted from original PAL procedures. Some parameters are slightly modified, and a few of them are built-in operators of Pipeline, which can only be used in Pipeline.

### Operator List

| Name | Type | PAL Procedure or built-in |
|---|---|---|
| NB_Classifier | Classifier | Naive Bayes |
| M_LOGR_Classifier | Classifier | Multi-Class Logistic Regression |
| SVM_Classifier | Classifier | Support Vector Machine |
| RDT_Classifier | Classifier | Random Decision Trees |
| DT_Classifier | Classifier | Decision Trees |

| Name | Type | PAL Procedure or built-in |
|---|---|---|
| HGBT_Classifier | Classifier | Hybrid Gradient Boosting Tree |
| MLP_Classifier | Classifier | Multilayer Perceptron |
| EXP_Regressor | Regressor | Exponential Regression |
| GEO_Regressor | Regressor | Bi-Variate Geometric Regression |
| GLM_Regressor | Regressor | Generalized Linear Models |
| HGBT_Regressor | Regressor | Hybrid Gradient Boosting Tree |
| LOG_Regressor | Regressor | Bi-Variate Natural Logarithmic Regression |
| MLP_Regressor | Regressor | Multilayer Perceptron |
| MLR_Regressor | Regressor | Multiple Linear Regression |
| POL_Regressor | Regressor | Polynomial Regression |
| RDT_Regressor | Regressor | Random Decision Trees |
| DT_Regressor | Regressor | Decision Trees |
| SVM_Regressor | Regressor | Support Vector Machine |
| Outlier | Resampler | Outlier Detection |
| SAMPLING | Resampler | Sampling |
| SMOTE | Resampler | SMOTE |
| SMOTETomek | Resampler | SMOTETomek |
| TomekLinks | Resampler | Tomek's Links |
| SingleExpSm | TimeSeries | Single Exponential Smoothing |
| DoubleExpSm | TimeSeries | Double Exponential Smoothing |
| TripleExpSm | TimeSeries | Triple Exponential Smoothing |
| BrownExpSm | TimeSeries | Brown Exponential Smoothing |
| BSTS | TimeSeries | Bayesian Structural Time Series |
| ARIMA | TimeSeries | ARIMA |
| AMTSA | TimeSeries | Additive Model Time Series Analysis |
| OneHotEncoder | Transformer | (built-in) |
| LabelEncoder | Transformer | (built-in) |
| ImputeTS | TsTransformer | Time Series Missing Value Handling |
| FS_unsupervised | Transformer | Feature Selection |
| FS_supervised | Transformer | Feature Selection |
| SCALE | Transformer | Scale |
| CBEncoder | TsTransformer | (built-in) |
| CATPCA | Transformer | Categorical Principal Component Analysis |

| Name | Type | PAL Procedure or built-in |
|---|---|---|
| PolynomialFeatures | Transformer | (built-in) |

You can use the `PAL_PIPELINE_INFO` procedure to query all available operators with their types and parameters.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_PARAMETER_TAB;
CREATE COLUMN TABLE PAL_PARAMETER_TAB (
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR(5000)
);
CALL _SYS_AFL.PAL_PIPELINE_INFO(PAL_PARAMETER_TAB,?)
```

Result:

Rows (40)

| | NAME | CATEGORY | |
|---|---|---|---|
| 1 | NB_Classifier | Classifier | {"LAPLACE":{"type":"number"},"DISCRETIZATION":{"enum":[0,1]}} |
| 2 | M_LOGR_Classifier | Classifier | {"MAX_ITERATION":{"type":"integer"},"ENET_ALPHA":{"type":"number"},"ENET_LAMBDA":{"type":"number"},"EXIT_THRESHOLD":{"type" |
| 3 | SVM_Classifier | Classifier | {"KERNEL_TYPE":{"enum":[0,1,2,3]},"POLY_DEGREE":{"type":"integer"},"RBF_GAMMA":{"type":"number"},"NU":{"type":"number"},"COEF |
| 4 | MLP_Classifier | Classifier | {"ACTIVATION":{"enum":[1,2,3,4,5,6,7,8,9,10,11,12,13]},"OUTPUT_ACTIVATION":{"enum":[1,2,3,4,5,6,7,8,9,10,11,12,13]},"HIDDEN_LAY |
| 5 | DT_Classifier | Classifier | {"ALGORITHM":{"enum":[1,2,3]},"ALLOW_MISSING_DEPENDENT":{"type":"integer"},"PERCENTAGE":{"type":"number"},"MIN_RECORDS_ |
| 6 | RDT_Classifier | Classifier | {"TREES_NUM":{"type":"integer"},"TRY_NUM":{"type":"integer"},"NODE_SIZE":{"type":"integer"},"MAX_DEPTH":{"type":"integer"},"SPLIT_ |
| 7 | HGBT_Classifier | Classifier | {"ITER_NUM":{"type":"integer"},"SEED":{"type":"integer"},"ROW_SAMPLE_RATE":{"type":"number"},"MAX_DEPTH":{"type":"integer"},"GAI |
| 8 | MLR_Regressor | Regressor | {"ADJUSTED_R2":{"type":"integer"},"ALG":{"enum":[1,2,4,5,6]},"ALPHA_TO_ENTER":{"type":"number"},"ALPHA_TO_REMOVE":{"type":"nu |
| 9 | MLP_Regressor | Regressor | {"ACTIVATION":{"enum":[1,2,3,4,5,6,7,8,9,10,11,12,13]},"OUTPUT_ACTIVATION":{"enum":[1,2,3,4,5,6,7,8,9,10,11,12,13]},"HIDDEN_LAY |
| 10 | POL_Regressor | Regressor | {"POLYNOMIAL_NUM":{"type":"integer"},"ADJUSTED_R2":{"type":"integer"},"ALG":{"enum":[0,1,2,5]}} |
| 11 | LOG_Regressor | Regressor | {"ADJUSTED_R2":{"type":"integer"},"ALG":{"enum":[0,1,2,5]}} |
| 12 | HGBT_Regressor | Regressor | {"ITER_NUM":{"type":"integer"},"SEED":{"type":"integer"},"SUBSAMPLE":{"type":"number"},"MAX_DEPTH":{"type":"integer"},"GAMMA":{"ty |
| 13 | GLM_Regressor | Regressor | {"FAMILY":{"enum":["gaussian","gaussian","poisson","binomial","gamma","inversegaussian","negativebinomial","ordinal"]},"LINK":{"enum":["] |
| 14 | GEO_Regressor | Regressor | {"ADJUSTED_R2":{"type":"integer"},"ALG":{"enum":[0,1,2,5]}} |
| 15 | RDT_Regressor | Regressor | {"TREES_NUM":{"type":"integer"},"TRY_NUM":{"type":"integer"},"NODE_SIZE":{"type":"integer"},"MAX_DEPTH":{"type":"integer"},"SPLIT_ |
| 16 | EXP_Regressor | Regressor | {"ADJUSTED_R2":{"type":"integer"},"ALG":{"enum":[0,1,2,5]}} |
| 17 | DT_Regressor | Regressor | {"ALLOW_MISSING_DEPENDENT":{"type":"integer"},"PERCENTAGE":{"type":"number"},"MIN_RECORDS_OF_PARENT":{"type":"integer"} |
| 18 | SVM_Regressor | Regressor | {"SVM_C":{"type":"number"},"KERNEL_TYPE":{"enum":[0,1,2,3]},"POLY_DEGREE":{"type":"integer"},"RBF_GAMMA":{"type":"number"},"C |
| 19 | SAMPLING | Resampler | {"SAMPLING_METHOD":{"enum":[0,1,2,3,4,5,6,7,8]},"INTERVAL":{"type":"integer"},"COLUMN_CHOOSE":{"type":"string"},"PERCENTAGE" |
| 20 | SMOTE | Resampler | {"SMOTE_AMOUNT":{"type":"integer"},"K_NEAREST_NEIGHBOURS":{"type":"integer"},"MINORITY_CLASS":{"type":"integer"},"RANDOM_ |
| 21 | SMOTETomek | Resampler | {"SMOTE_AMOUNT":{"type":"integer"},"K_NEAREST_NEIGHBOURS":{"type":"integer"},"MINORITY_CLASS":{"type":"integer"},"RANDOM |

> **Note**
>
> MLP_Classifier, SVM_Classifier, MLP_Regressor, SVM_Regressor are adapted from Unified Function, so they use parameters from Unified Function.

### Parameter Type

When calling PAL's original procedure, parameters are entered into a PARAMETER_TABLE with the following format:

```
DROP TABLE PAL_PARAMETER_TAB;
CREATE COLUMN TABLE PAL_PARAMETER_TAB (
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR(1000)
);
```

In operator, the arguments are represented by the `args` part of operator. According to the way the arguments are entered in the original procedure, the input type of the arguments, and their JSON forms in the operator are as follows.

1. Single row of single type
   - in PARAMETER_TABLE:

| NAME | INT | DOUBLE | STRING |
|---|---|---|---|
| 'int_value' | 1 | null | null |
| 'double_value' | null | 1.0 | null |
| 'string_value' | null | null | 'str' |

   - in JSON

```
{"int_value": 1, "double_value":1.0, "string_value":"str"}
```

2. Multiple rows of a single type
   - in PARAMETER_TABLE:

| NAME | INT | DOUBLE | STRING |
|---|---|---|---|
| 'int_vec' | 1 | null | null |
| 'int_vec' | 2 | null | null |
| 'int_vec' | 3 | null | null |
| 'double_vec' | null | 1.0 | null |
| 'double_vec' | null | 2.0 | null |
| 'double_vec' | null | 3.0 | null |
| 'string_vec' | null | null | 'A' |
| 'string_vec' | null | null | 'B' |
| 'string_vec' | null | null | 'C' |

   - in JSON

```
{"int_vec": [1, 2, 3], "double_vec":[1.0, 2.0, 3.0], "string_vec":["A", "B",
"C"]}
```

3. Single row of multiple types
   - in PARAMETER_TABLE

| NAME | INT | DOUBLE | STRING |
|---|---|---|---|
| 'multi0' | 0 | null | 'A' |
| 'multi1' | 1 | 2.0 | null |
| 'multi2' | null | 3.0 | 'B' |
| 'multi4' | 3 | 4.0 | 'C' |

- in JSON

```
{"multi0": [0, "A"], "multi1":[1, 2.0], "multi2":[3.0, "B"], "multi3":[3,
4.0, "C"]}
```

4. Multiple rows of multiple types

- in PARAMETER_TABLE

| NAME | INT | DOUBLE | STRING |
|---|---|---|---|
| 'multi_vec0' | 0 | null | 'A' |
| 'multi_vec0' | 1 | 2.0 | null |
| 'multi_vec1' | null | 3.0 | 'B' |
| 'multi_vec1' | 3 | 4.0 | 'C' |

- in JSON

```
{"multi_vec0": [[0, "A"],[1, 2.0]], "multi_vec1":[[3.0, "B"],[3, 4.0, "C"]]}
```

**Example**

An `args` content in operator:

```
{"A":0, "B":1.0, "C":"STR", "D":[1, 2, 3], "E":[[1, "A"], [2, "B"]]}
```

will be expanded to a PAL_PARAMETER_TABLE as:

| NAME | INT | DOUBLE | STRING |
|---|---|---|---|
| 'A' | 0 | null | null |
| 'B' | null | 1.0 | null |
| 'C' | null | null | 'STR' |
| 'D' | 1 | null | null |
| 'D' | 2 | null | null |
| 'D' | 3 | null | null |
| 'E' | 1 | null | 'A' |
| 'E' | 2 | null | 'B' |

## Differences Between Pipeline Operators and Original PAL Procedures

To facilitate parameter selection for AutoML, some operator parameters may differ from the original PAL procedure parameters. This section lists the operators and their parameter tables that are inconsistent with the original PAL procedure.

**Outlier**

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| WINDOW_SIZE | INTEGER | 3 | The window size for median filter, odd number, no less than 1. The value 1 means median filter is not applied. | |
| OUTLIER_METHOD | INTEGER | 0 | The method to calculate the outlier score from residual.<br>• 0: Z1 score<br>• 1: Z2 score<br>• 2: IQR score<br>• 3: MAD score<br>• 4: Isolation Forest score<br>• 5: DBSCAN | |
| THRESHOLD | DOUBLE | • 3 for OUTLIER_METHOD = 0,1,3<br>• 1.5 for OUTLIER_METHOD = 2 | The threshold for outlier score. If the absolute value of outlier score is beyond the threshold, PAL considers the corresponding data point as an outlier. | Only valid when OUTLIER_METHOD = 0, 1, 2, or 3. |
| THRESHOLD_ISOLATION | DOUBLE | 0.7 | The threshold for outlier score of isolation forest. If the absolute value of outlier score is beyond the threshold, PAL considers the corresponding data point as an outlier. | Only valid when OUTLIER_METHOD = 4 (Isolation Forest) |
| DETECT_SEASONALITY | INTEGER | 0 | When calculating the residual,<br>• 0: Does not consider the seasonal decomposition.<br>• 1: Considers the seasonal decomposition. | |
| ALPHA | DOUBLE | 0.2 | The criterion for the autocorrelation coefficient. The value range is (0, 1). A larger value indicates stricter requirements for seasonality. | Only valid when DETECT_SEASONALITY is 1. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| EXTRAPOLATION | INTEGER | 0 | Specifies whether to extrapolate the end-points.<br>• 0: No<br>• 1: Yes | Set to 1 when there's an end-point issue. Only valid when DETECT_SEASONALITY is 1. |
| SEED | INTEGER | 0 | Specifies the seed for random number generator.<br>• 0: Uses the current time (in second) as seed.<br>• Others: Uses the specified value as seed. | Only valid when OUTLIER_METHOD is 4 (Isolation Forest). |
| N_ESTIMATORS | INTEGER | 100 | Specifies the number of trees to grow. | Only valid when OUTLIER_METHOD is 4 (Isolation Forest). |
| MAX_SAMPLES | INTEGER | 256 | Specifies the number of samples to draw from input to train each tree. If MAX_SAMPLES is larger than the number of samples provided, all samples are used for all trees. | Only valid when OUTLIER_METHOD is 4 (Isolation Forest). |
| BOOTSTRAP | INTEGER | 0 | Specifies the sampling method.<br>• 0: Sampling without replacement.<br>• 1: Sampling with replacement. | Only valid when OUTLIER_METHOD is 4 (Isolation Forest). |
| MINPTS | INTEGER | 1 | Specifies the minimum number of points required to form a cluster. The point itself is not included in MINPTS. | Only valid when OUTLIER_METHOD is 5 (DBSCAN). |
| RADIUS | DOUBLE | 0.5 | Specifies the scan radius (eps). | Only valid when OUTLIER_METHOD is 5 (DBSCAN). |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| IMPUTE_FLAG | INTEGER | 0 | Specifies whether to impute after detecting outliers. Valid values are:<br>• 0<br>• 1<br>The imputation method is to drop the residual of outlier points. | |

## Built-in Operators

Built-in operators are only available in pipeline without corresponding PAL procedures.

| Name | Type | Description |
|---|---|---|
| OneHotEncoder | Transformer | Encodes categorical features with value between 1 and n_classes of each feature. |
| LabelEncoder | Transformer | Encodes categorical features using a one-hot schema. |
| CBEncoder | Transformer | Encodes categorical features using Bayesian encoding. |
| PolynomialFeatures | Transformer | Generate polynomial features from existing features. |

### LabelEncoder

Input data requirements:

- No missing value
- Contains at least one categorical feature.

Parameters

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| IGNORE_UNKNOWN | int | 1 | • 0: Categorical features in predict data must not contain unknown categorical value<br>• 1: Unknown categorical values will be encoded from -1 to -n, n is number of unknown values | |

### OneHotEncoder

Input data requirements:

- No missing value
- Contains at least one categorical feature.

Parameters

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| IGNORE_UNKNOWN | int | 1 | • 0: Categorical features in predict data must not contain unknown categorical value<br>• 1: Unknown categorical value will be encoded to array 0. | |
| MINIMUM_FRACTION | double | 0.0 | Minimum fraction of unique values in a feature to consider the feature to be categorical. | |

### PolynoimalFeatures

Generates polynomial and interaction Features. Additional features will be created that includes all possible polynomial combinations of features with a degree ranging from MIN_DEGREE to MAX_DEGREE.

Input data requirements:

- No missing value
- No categorical feature

Parameters

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| MIN_DEGREE | int | 1 | The minimum polynomial degree of generated features. | |
| MAX_DEGREE | int | 2 | The maximum polynomial degree of generated features. | |
| INTERACTION_ONLY | bool | False | When set to True, only interaction features are produced. | |
| INCLUDE_BIAS | bool | True | When set to True, a bias column is included in the output table. For features where all polynomial powers are zero, set the value of this bias column to 1. | |

Examples

Create a data table which contains numerical features [a, b, c].

Example 1

```
{"MIN_DEGREE":1, "MAX_DEGREE": 2, "INTERACTION_ONLY": false, "INCLUDE_BIAS":
true}
```

Result:

```
1, a,  b, c, a^2, a*b, a*c,  b^2, b*c, c^2
```

Example 2

```
{"MIN_DEGREE":1, "MAX_DEGREE": 2, "INTERACTION_ONLY": true, "INCLUDE_BIAS": true}
```

Result:

```
1, a,  b, c, a*b, a*c, b*c
```

Example 3

```
{"MIN_DEGREE":1, "MAX_DEGREE": 2, "INTERACTION_ONLY": true, "INCLUDE_BIAS":
false}
```

Result:

```
a,  b, c, a*b, a*c, b*c
```

## 5.6.2 Customizable Output Interface

Customizable output interface is a procedure used to perform single or composite operator with variable output table format.

Due to the limitations of the AFL framework, the number of output tables and column types of the PAL procedure are determined by the definition of the procedure and the format of the input table before executing the procedure. This means that the procedure cannot dynamically adjust the output table format based on the input model and parameters.

By using this procedure, you can customize the output table format to match the output results of the algorithm model, thus allowing the support of outputs of different algorithm models and complex combination models, such as PAL pipeline and Massive Function.

### _SYS_AFL.PAL_PIPELINE_EXECUTE

#### Overview of All Tables

| Position | Table Name | Type |
|---|---|---|
| 1 | <Model Table> | IN |
| 2 | <Parameters Table> | IN |
| 3 | <Input Table 1> | IN |
| 4 | <Input Table 2> | IN |
| 5 | <Input Table 3> | IN |
| 6 | <Template Table 1> | IN |
| 7 | <Template Table 2> | IN |
| 8 | <Template Table 3> | IN |
| 9 | <UpdatedModel Table> | OUT |
| 10 | <Output Table 1> | OUT |
| 11 | <Output Table 2> | OUT |
| 12 | <Output Table 3> | OUT |

#### Input Table

- Model Table
  The Model Table is used to input user-defined or trained JSON Model.

  | Column | Data Type | Description |
  |---|---|---|
  | 1st column | INTEGER | Row index |
  | 2nd column | VARCHAR or NVARCHAR | Saved model |

- **Parameters Table**
  Mandatory Parameters

| Name | Data Type | Description |
| --- | --- | --- |
| EXECUTE | VARCHAR or NVARCHAR | Specifies the name of the method to be executed. All operator models support only the fit and predict methods. |

Optional Parameters
The following parameters are optional. If it is not specified, PAL will use its default value.

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
| --- | --- | --- | --- | --- |
| INPUT_NUMBER | INTEGER | 1 | Specifies the number of input tables, indicating the number of tables [1, 2, 3] to be selected as algorithm inputs, usually 1. | |
| OUTPUT_MODEL | INTEGER | • 0: when EXECUTE is set to predict.<br>• 1: when EXECUTE is set to fit. | Specifies whether to output the model to the updated model table:<br>• 0: No<br>• 1: Yes | |

- Input Tables[1, 2, 3]
  These three input tables are used as the input tables of the EXECUTE method of the model. The specific input format depends on the EXECUTE method of the model. If the model requires fewer than three input tables, only the format of the required input table needs to be specified. The remaining input tables can be replaced with any table as a dummy input.
- Template Tables [1, 2, 3]
  These three Template Tables are used to define the output format of Output Tables [1, 2, 3]. You need to define the Output Tables based on the output of the EXECUTE method of the model, and pass them into the Template Tables. The AFL framework generates Output Tables in the same format as the Template Tables. Similarly to the Input Tables, if the number of output tables generated by the model's EXECUTE method is less than three, you only need to specify the format of the Template Tables for the required output tables, and the remaining Template Tables can be filled with any form of a dummy table.

Output Table

- Updated Model Table
  The Output Model Table has a fixed format and is used to output the JSON string of a trained or updated model.

| Column | Data Type | Description |
| --- | --- | --- |
| 1st column | INTEGER | Row index |
| 2nd column | VARCHAR or NVARCHAR | Saved model |

- Output Tables [1, 2, 3]
  Output Tables are used to output the results of a Model's EXECUTE method, and their format is consistent with that of Template Tables.

## Operator Model

Unlike traditional PAL procedures, the PAL_PIPELINE_EXECUTE procedure does not represent any specific algorithm. Its input and output are determined by the algorithm model in the Model Table and the EXECUTE method in the Parameter Table. One algorithm model can have many EXECUTE methods, such as `fit`, `predict`, and other possible methods.

The procedure supports both single and composite models. Single models support all algorithms listed in Pipeline Operator [page 988], while composite models currently support the pipeline and the massive operator.

The model is described in JSON format, usually containing a member with the model name as the key.

```
{"<name>":<content>}
```

### Single Operator

A single operator model does not include any operators. Its content typically consists of a JSON object that includes an "args" member. If it is a trained model, it also includes a "model" member.

```
{"<name>":{"args":<args content>, "model":<model content>}}
```

- unfitted operator model

```
{"RDT_Classifier":{"args":
{"TREES_NUM":100,"NODE_SIZE":9,"CALCULATE_OOB":0,"SAMPLE_FRACTION":0.75}}}
{"SCALE":{"args":{"SCALING_METHOD": 1, "Z-SCORE_METHOD":2}}}
```

- fitted operator model

```
{"RDT_Classifier":{"args":
{"TREES_NUM":100,"NODE_SIZE":9,"CALCULATE_OOB":0,"SAMPLE_FRACTION":0.75},
"model":<model content>}}
{"SCALE":{"args":{"SCALING_METHOD": 1, "Z-SCORE_METHOD":2}, "model":<model
content>}}
```

#### Examples

We will use a small classification dataset to demonstrate how to use this procedure to handle classification problems, and how to construct and use a classification pipeline and a massive operator.

- classification dataset

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_CLASSIFICATION_TRAIN_DATA;
CREATE COLUMN TABLE PAL_CLASSIFICATION_TRAIN_DATA(
"ID" INTEGER,
"X0" INTEGER,
"X1" INTEGER,
"X2" INTEGER,
"X3" DOUBLE,
"X4" DOUBLE,
"X5" INTEGER,
"TARGET" NVARCHAR(2)
);
INSERT INTO PAL_CLASSIFICATION_TRAIN_DATA VALUES(0,72,35,0,33.6,0.627,50,'1');
INSERT INTO PAL_CLASSIFICATION_TRAIN_DATA
VALUES(1,66,29,0,26.6,0.35103,31,'0');
INSERT INTO PAL_CLASSIFICATION_TRAIN_DATA VALUES(2,64,0,0,23.3,0.672,32,'1');
INSERT INTO PAL_CLASSIFICATION_TRAIN_DATA
VALUES(3,66,23,94,28.1,0.166998,21,'0');
```

```
INSERT INTO PAL_CLASSIFICATION_TRAIN_DATA
VALUES(4,40,35,168,43.1,2.2883,33,'1');
INSERT INTO PAL_CLASSIFICATION_TRAIN_DATA VALUES(5,74,0,0,25.6,0.201,30,'0');
INSERT INTO PAL_CLASSIFICATION_TRAIN_DATA
VALUES(6,50,32,88,31.0,0.248,26,'1');
INSERT INTO PAL_CLASSIFICATION_TRAIN_DATA
VALUES(8,70,45,543,30.5,0.158,53,'1');
INSERT INTO PAL_CLASSIFICATION_TRAIN_DATA
VALUES(9,96,0,0,0.0,0.231998,54,'1');
INSERT INTO PAL_CLASSIFICATION_TRAIN_DATA VALUES(10,92,0,0,37.6,0.191,30,'0');
INSERT INTO PAL_CLASSIFICATION_TRAIN_DATA VALUES(11,74,0,0,38.0,0.537,34,'1');
INSERT INTO PAL_CLASSIFICATION_TRAIN_DATA
VALUES(12,80,0,0,27.1,1.44098,57,'0');
INSERT INTO PAL_CLASSIFICATION_TRAIN_DATA
VALUES(13,60,23,846,30.1,0.397996,59,'1');
INSERT INTO PAL_CLASSIFICATION_TRAIN_DATA
VALUES(14,72,19,175,25.8,0.5871,51,'1');
INSERT INTO PAL_CLASSIFICATION_TRAIN_DATA VALUES(15,0,0,0,30.0,0.484,32,'1');
INSERT INTO PAL_CLASSIFICATION_TRAIN_DATA
VALUES(18,30,38,83,43.3,0.183,33,'0');
INSERT INTO PAL_CLASSIFICATION_TRAIN_DATA
VALUES(19,70,30,96,34.6,0.529,32,'1');
INSERT INTO PAL_CLASSIFICATION_TRAIN_DATA
VALUES(20,88,41,235,39.3,0.7041,27,'0');
DROP TABLE PAL_CLASSIFICATION_TEST_DATA;
CREATE COLUMN TABLE PAL_CLASSIFICATION_TEST_DATA(
"ID" INTEGER,
"X0" INTEGER,
"X1" INTEGER,
"X2" INTEGER,
"X3" DOUBLE,
"X4" DOUBLE,
"X5" INTEGER
);
INSERT INTO PAL_CLASSIFICATION_TEST_DATA VALUES(0,0,0,0,35.3,0.134,29);
INSERT INTO PAL_CLASSIFICATION_TEST_DATA VALUES(1,74,0,0,29.6,0.254,31);
INSERT INTO PAL_CLASSIFICATION_TEST_DATA VALUES(2,66,15,140,23.2,0.48704,22);
INSERT INTO PAL_CLASSIFICATION_TEST_DATA VALUES(3,60,33,192,24.0,0.966,33);
INSERT INTO PAL_CLASSIFICATION_TEST_DATA VALUES(4,110,24,240,45.4,0.721,54);
INSERT INTO PAL_CLASSIFICATION_TEST_DATA VALUES(5,68,39,304,37.7,0.254,41);
INSERT INTO PAL_CLASSIFICATION_TEST_DATA VALUES(6,74,27,0,29.0,0.203,32);
INSERT INTO PAL_CLASSIFICATION_TEST_DATA VALUES(7,66,22,0,25.0,0.307,24);
INSERT INTO PAL_CLASSIFICATION_TEST_DATA VALUES(8,72,54,0,36.6,0.17802,45);
INSERT INTO PAL_CLASSIFICATION_TEST_DATA VALUES(9,90,51,220,49.7,0.325,31);
```

- single classifier

```
SET SCHEMA DM_PAL;
DROP TABLE MODEL_TABLE;
CREATE COLUMN TABLE MODEL_TABLE (
    "INDEX" INTEGER,
    "CONTENT" NVARCHAR(5000)
);
INSERT INTO MODEL_TABLE VALUES (0, '{"NB_Classifier":{"args":
{"LAPLACE":0.1}}}'); -- using a single operator NB_Classifier as input model
DROP TABLE PAL_PARAMETER_TAB;
CREATE COLUMN TABLE PAL_PARAMETER_TAB (
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR(5000)
);
INSERT INTO PAL_PARAMETER_TAB VALUES ('EXECUTE', null, null, 'fit'); --
execute "fit" method which take one table as input
-- Define the updated model or directly use the model table to store the
fitted model.
DROP TABLE UPDATED_MODEL_TABLE;
CREATE COLUMN TABLE UPDATED_MODEL_TABLE (
```

```
    "INDEX" INTEGER,
    "CONTENT" NVARCHAR(5000)
);
-- Since the fit method of the classifier only requires one output table and
does not produce any output tables, a dummy table needs to be used to fill in
the extra input and template tables.
-- Here we use the parameter table as the dummy table.
DO BEGIN
lt_model = SELECT * FROM MODEL_TABLE;
lt_args = SELECT * FROM PAL_PARAMETER_TAB;
lt_dummy = SELECT * FROM PAL_PARAMETER_TAB;
lt_data = SELECT * FROM PAL_CLASSIFICATION_TRAIN_DATA;
CALL _SYS_AFL.PAL_PIPELINE_EXECUTE(:lt_model, :lt_args,
    :lt_data, :lt_dummy, :lt_dummy,
    :lt_dummy, :lt_dummy, :lt_dummy,
    lt_output_model,
    lt_outp0, lt_output1, lt_output2);
INSERT INTO UPDATED_MODEL_TABLE SELECT * FROM :lt_output_model;
END;
SELECT * FROM UPDATED_MODEL_TABLE;
-- The result format of classifier predict method is as follows:
DROP TABLE RESULT_TABLE;
CREATE COLUMN TABLE RESULT_TABLE (
    "ID" INTEGER,
    "SCORE" NVARCHAR(256),
    "PROB" DOUBLE
);
DROP TABLE PAL_PARAMETER_TAB;
CREATE COLUMN TABLE PAL_PARAMETER_TAB (
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR(5000)
);
INSERT INTO PAL_PARAMETER_TAB VALUES ('EXECUTE', NULL, NULL, 'predict');
DO BEGIN
lt_model = SELECT * FROM UPDATED_MODEL_TABLE;
lt_args = SELECT * FROM PAL_PARAMETER_TAB;
lt_dummy = SELECT * FROM PAL_PARAMETER_TAB;
lt_data = SELECT * FROM PAL_CLASSIFICATION_TEST_DATA;
lt_result = SELECT * FROM RESULT_TABLE;
CALL _SYS_AFL.PAL_PIPELINE_EXECUTE(:lt_model, :lt_args,
    :lt_data, :lt_dummy, :lt_dummy,
    :lt_result, :lt_dummy, :lt_dummy,
    lt_updated_model,
    lt_output0, lt_output1, lt_output2
);
INSERT INTO RESULT_TABLE SELECT * FROM :lt_output0;
END;
SELECT * FROM RESULT_TABLE;
```

**Expected Result**

- UPDATED_MODEL_TABLE

Rows (1) SQL

| | INDEX | CONTENT |
|---|---|---|
| 1 | 0 | {"NB_Classifier":{"args":{"LAPLACE":0.1},"model":{"Laplace":0.1,"( |

- RESULT_TABLE

Rows (10)

| | ID | SCORE | PROB |
|---|---|---|---|
| 1 | 0 | 1 | 0.6459358662835943 |
| 2 | 1 | 0 | 0.8927038362386024 |
| 3 | 2 | 0 | 0.7877061016583848 |
| 4 | 3 | 1 | 0.6315312772599805 |
| 5 | 4 | 1 | 0.9039560764835465 |
| 6 | 5 | 1 | 0.9143199154624979 |
| 7 | 6 | 0 | 0.8637613773314402 |
| 8 | 7 | 0 | 0.8624372038623515 |
| 9 | 8 | 0 | 0.7896080672956086 |
| 10 | 9 | 1 | 0.5958503280640775 |

- single transformer

```
SET SCHEMA DM_PAL;
DROP TABLE MODEL_TABLE;
CREATE COLUMN TABLE MODEL_TABLE (
    "INDEX" INTEGER,
    "CONTENT" NVARCHAR(5000)
);
INSERT INTO MODEL_TABLE VALUES (0, '{"CATPCA":{"args":
{"COMPONENTS_PERCENTAGE":0.5}}}'); -- using a single operator CATPCA as input
model
DROP TABLE PAL_PARAMETER_TAB;
CREATE COLUMN TABLE PAL_PARAMETER_TAB (
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR(5000)
);
INSERT INTO PAL_PARAMETER_TAB VALUES ('EXECUTE', null, null, 'fit');
DROP TABLE UPDATED_MODEL_TABLE;
CREATE COLUMN TABLE UPDATED_MODEL_TABLE (
    "INDEX" INTEGER,
    "CONTENT" NVARCHAR(5000)
);
-- The fit method of transformer will generate an output table. Here, the
COMPONENTS_PERCENTAGE of CATPCA operator is set to 0.5, which will reduce the
original dataset's 6 features to 3 features while keeping the target column
unchanged.
DROP TABLE FIT_TRANSFORM_RESULT;
CREATE COLUMN TABLE FIT_TRANSFORM_RESULT (
    "INDEX" INTEGER,
    "CONPONENT0" DOUBLE,
    "CONPONENT1" DOUBLE,
    "CONPONENT2" DOUBLE,
    "TARGET" NVARCHAR(2)
);
DO BEGIN
lt_model = SELECT * FROM MODEL_TABLE;
lt_args = SELECT * FROM PAL_PARAMETER_TAB;
lt_result = SELECT * FROM FIT_TRANSFORM_RESULT;
lt_dummy = SELECT * FROM PAL_PARAMETER_TAB;
lt_data = SELECT * FROM PAL_CLASSIFICATION_TRAIN_DATA;
CALL _SYS_AFL.PAL_PIPELINE_EXECUTE(:lt_model, :lt_args,
    :lt_data, :lt_dummy, :lt_dummy,
    :lt_result, :lt_dummy, :lt_dummy,
    lt_output_model,
```

```
    lt_output0, lt_output1, lt_output2);
INSERT INTO UPDATED_MODEL_TABLE SELECT * FROM :lt_output_model;
INSERT INTO FIT_TRANSFORM_RESULT SELECT * FROM :lt_output0;
END;
SELECT * FROM UPDATED_MODEL_TABLE;
SELECT * FROM FIT_TRANSFORM_RESULT;
-- predict process
-- The result of predict has one less column compared to
FIT_TRANSFORM_RESULT, which is the TARGET column.
DROP TABLE RESULT_TABLE;
CREATE COLUMN TABLE RESULT_TABLE (
    "INDEX" INTEGER,
    "CONPONENT0" DOUBLE,
    "CONPONENT1" DOUBLE,
    "CONPONENT2" DOUBLE
);
DROP TABLE PAL_PARAMETER_TAB;
CREATE COLUMN TABLE PAL_PARAMETER_TAB (
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR(5000)
);
INSERT INTO PAL_PARAMETER_TAB VALUES ('EXECUTE', NULL, NULL, 'predict');
DO BEGIN
lt_model = SELECT * FROM UPDATED_MODEL_TABLE;
lt_args = SELECT * FROM PAL_PARAMETER_TAB;
lt_dummy = SELECT * FROM PAL_PARAMETER_TAB;
lt_data = SELECT * FROM PAL_CLASSIFICATION_TEST_DATA;
lt_result = SELECT * FROM RESULT_TABLE;
CALL _SYS_AFL.PAL_PIPELINE_EXECUTE(:lt_model, :lt_args,
    :lt_data, :lt_dummy, :lt_dummy,
    :lt_result, :lt_dummy, :lt_dummy,
    lt_updated_model,
    lt_output0, lt_output1, lt_output2
);
INSERT INTO RESULT_TABLE SELECT * FROM :lt_output0;
END;
SELECT * FROM RESULT_TABLE;
```

**Expected Result**

- UPDATED_MODEL_TABLE

Rows (1) SQL

| | INDEX | |
|---|---|---|
| 1 | 0 | {"CATPCA":{"args":{"COMPONENTS_PERCENTAGE":0.5},"model":{"component-dime |

- FIT_TRANSFORM_RESULT

Rows (18)

| | INDEX | CONPONENT0 | CONPONENT1 | CONPONENT2 | TARGET |
|---|---|---|---|---|---|
| 1 | 0 | -128.4087956497776 | 4.575161661620838 | -16.474925118596257 | 1 |
| 2 | 1 | -129.09746011459157 | -2.064926616374198 | -12.416008965239735 | 0 |
| 3 | 2 | -130.05351569046982 | 2.98182592003181 | 14.497447178610958 | 1 |
| 4 | 3 | -35.61696661786938 | -2.5086664093216777 | -8.147263287133166 | 0 |
| 5 | 4 | 39.1517605701951 | -29.670581586747883 | -9.980613131094877 | 1 |
| 6 | 5 | -130.11354892330297 | 11.524959492223577 | 10.194567126281553 | 0 |
| 7 | 6 | -41.13651656319294 | -19.049424144203186 | -10.635428929283178 | 1 |
| 8 | 8 | 414.5122662126202 | 4.781731585472647 | -10.199786709379651 | 1 |
| 9 | 9 | -129.71112472629474 | 41.836683282788904 | 17.28241248684955 | 1 |
| 10 | 10 | -130.09216862934767 | 26.03886082858129 | 1.0919205891697148 | 0 |
| 11 | 11 | -129.9503631424407 | 9.842596992140116 | 7.608496764193727 | 1 |
| 12 | 12 | -129.45814674815585 | 22.04502842248527 | 15.059951970508372 | 0 |
| 13 | 13 | 716.6813318736487 | 2.912480974675982 | 20.64042905938402 | 1 |
| 14 | 14 | 45.88778717949281 | 10.626221485127575 | 3.967163894938573 | 1 |
| 15 | 15 | -129.86986353601958 | -58.37817909837736 | 32.34226679462531 | 1 |
| 16 | 18 | -45.64931842356288 | -40.180582358003655 | -11.503577807420672 | 0 |
| 17 | 19 | -33.09070802034709 | 0.6167657155326027 | -14.380897351096435 | 1 |
| 18 | 20 | 106.01535094941585 | 14.070043852347295 | -28.946154565317766 | 0 |

- RESULT_TABLE

Rows (10)

| | INDEX | CONPONENT0 | CONPONENT1 | CONPONENT2 |
|---|---|---|---|---|
| 1 | 0 | -129.9154519651458 | -60.00131285034733 | 29.987580991669667 |
| 2 | 1 | -130.06801486978293 | 10.926613942946217 | 9.283366090626778 |
| 3 | 2 | 10.074541752402531 | 0.5811513995083555 | 1.5381236197525396 |
| 4 | 3 | 62.91738877869125 | -6.578968306229116 | -8.268786404089866 |
| 5 | 4 | 111.08631201733938 | 42.28558028770847 | -15.580722629529037 |
| 6 | 5 | 175.2693643608918 | -0.8825379881261031 | -15.239359828819875 |
| 7 | 6 | -129.14560781267326 | 5.577423797932285 | -13.60508899946787 |
| 8 | 7 | -129.51099864766095 | -1.627233493157114 | -7.754509188710337 |
| 9 | 8 | -127.88140015627842 | -0.9608522064579736 | -35.1014661540596 |
| 10 | 9 | 91.51030817761169 | 12.482478757220006 | -40.55095646303168 |

## Composite Operator

The composite operator is composed of sub-operators, which can be single operators or composite operators. Currently supported composite operators include the pipeline and the massive operator.

### Pipeline

The pipeline model here is in the form of an array of operators, which is different from the format of PAL_PIPELINE_FIT and PAL_PIPELINE_PREDICT. A pipeline of a certain type can also be used as a sub-

operator in other composite operators of the corresponding type. For example, a classification pipeline can be used as a sub-operator in a classifier massive operator.

```
[<operator 0>, ..., <operator n>]
```

Pipeline Structure

- Classification Pipeline

```
[
    {"FS_unsupervised":{ "args":{ "FS_METHOD":9, "TOP_K_BEST":10 }}},
    {"SCALE":{ "args":{ "SCALING_METHOD": 1, "Z-SCORE_METHOD":2 }}},
    {"RDT_Classifier":{ "args":{ "TREES_NUM":100, "NODE_SIZE":9,
"CALCULATE_OOB":0, "SAMPLE_FRACTION":0.75 }}}
]
```

- Regression Pipeline

```
[
    {"FS_unsupervised":{"args":{"FS_METHOD":9,"TOP_K_BEST":10}}},
    {"SCALE":{"args":{"SCALING_METHOD":1,"Z-SCORE_METHOD":2}}},
    {"GEO_Regressor":{"args":{"ALG":2}}}
]
```

- TimeSeries Pipeline

```
[
    {"Outlier":{"args":{"FS_METHOD":9,"TOP_K_BEST":10}}},
    {"ImputeTS":{"args":{"IMPUTATION_TYPE":9}}},
    {"BSTS":{"args":{"BURN_IN":0.2, "SEASONAL_PERIOD":-1}}}
]
```

- Transform Pipeline

```
[
    {"SCALE":{"args":{"SCALING_METHOD":1,"Z-SCORE_METHOD":2}}},
    {"PolynomialFeatures":{"args":{"MIN_DEGREE":1,"MAX_DEGREE":3,
"INTERACTION_ONLY":true, "INCLUDE_BIAS":false}}},
    {"FS_unsupervised":{"args":{"FS_METHOD":9,"TOP_K_BEST":10}}},
]
```

**Examples**

- Classification Pipeline

```
SET SCHEMA DM_PAL;
DROP TABLE MODEL_TABLE;
CREATE COLUMN TABLE MODEL_TABLE (
    "INDEX" INTEGER,
    "CONTENT" NVARCHAR(5000)
);
INSERT INTO MODEL_TABLE VALUES(0, '[');
INSERT INTO MODEL_TABLE VALUES(1, '    {"SCALE":{ "args":{ "SCALING_METHOD":
1, "Z-SCORE_METHOD":2 }}},');
INSERT INTO MODEL_TABLE VALUES(2, '    {"FS_unsupervised":{ "args":
{ "FS_METHOD":9, "TOP_K_BEST":10 }}},');
INSERT INTO MODEL_TABLE VALUES(3, '    {"NB_Classifier":{ "args":
{"LAPLACE":0.3}}}');
INSERT INTO MODEL_TABLE VALUES(4, ']');
DROP TABLE PAL_PARAMETER_TAB;
CREATE COLUMN TABLE PAL_PARAMETER_TAB (
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
```

```
    "STRING_VALUE" NVARCHAR(5000)
);
INSERT INTO PAL_PARAMETER_TAB VALUES ('EXECUTE', null, null, 'fit');
DROP TABLE UPDATED_MODEL_TABLE;
CREATE COLUMN TABLE UPDATED_MODEL_TABLE (
    "INDEX" INTEGER,
    "CONTENT" NVARCHAR(5000)
);
DROP TABLE RESULT_TABLE;
CREATE COLUMN TABLE RESULT_TABLE (
    "ID" INTEGER,
    "SCORE" NVARCHAR(256),
    "PROB" DOUBLE
);
DO BEGIN
lt_model = SELECT * FROM MODEL_TABLE;
lt_args = SELECT * FROM PAL_PARAMETER_TAB;
lt_dummy = SELECT * FROM PAL_PARAMETER_TAB;
lt_data = SELECT * FROM PAL_CLASSIFICATION_TRAIN_DATA;
lt_result = SELECT * FROM RESULT_TABLE;
CALL _SYS_AFL.PAL_PIPELINE_EXECUTE(:lt_model, :lt_args,
    :lt_data, :lt_dummy, :lt_dummy,
    :lt_dummy, :lt_dummy, :lt_dummy,
    lt_output_model,
    lt_outp0, lt_output1, lt_output2);
INSERT INTO UPDATED_MODEL_TABLE SELECT * FROM :lt_output_model;
END;
SELECT * FROM UPDATED_MODEL_TABLE;
DROP TABLE RESULT_TABLE;
CREATE COLUMN TABLE RESULT_TABLE (
    "ID" INTEGER,
    "SCORE" NVARCHAR(256),
    "PROB" DOUBLE
);
DROP TABLE PAL_PARAMETER_TAB;
CREATE COLUMN TABLE PAL_PARAMETER_TAB (
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR(5000)
);
INSERT INTO PAL_PARAMETER_TAB VALUES ('EXECUTE', NULL, NULL, 'predict');
DO BEGIN
lt_model = SELECT * FROM UPDATED_MODEL_TABLE;
lt_args = SELECT * FROM PAL_PARAMETER_TAB;
lt_dummy = SELECT * FROM PAL_PARAMETER_TAB;
lt_data = SELECT * FROM PAL_CLASSIFICATION_TEST_DATA;
lt_result = SELECT * FROM RESULT_TABLE;
CALL _SYS_AFL.PAL_PIPELINE_EXECUTE(:lt_model, :lt_args,
    :lt_data, :lt_dummy, :lt_dummy,
    :lt_result, :lt_dummy, :lt_dummy,
    lt_updated_model,
    lt_output0, lt_output1, lt_output2
);
INSERT INTO RESULT_TABLE SELECT * FROM :lt_output0;
END;
SELECT * FROM RESULT_TABLE;
```

**Expected Result**

- UPDATED_MODEL_TABLE

Rows (1) SQL

| | INDEX | |
|---|---|---|
| 1 | 0 | [{"SCALE":{"args":{"SCALING_METHOD":1,"Z-SCORE_METHOD":2},"model":{"name" |

- RESULT_TABLE

Rows (10) SQL

| | ID | SCORE | PROB |
|---|---|---|---|
| 1 | 0 | 0 | 0.8719596120632569 |
| 2 | 1 | 0 | 0.8519847086231257 |
| 3 | 2 | 0 | 0.7297268541071674 |
| 4 | 3 | 1 | 0.5967006516411374 |
| 5 | 4 | 1 | 0.9004896305119955 |
| 6 | 5 | 1 | 0.9610937385615514 |
| 7 | 6 | 0 | 0.8248178381751159 |
| 8 | 7 | 0 | 0.8361784374683826 |
| 9 | 8 | 0 | 0.8141078376593401 |
| 10 | 9 | 1 | 0.6906605072814597 |

- Transformer Pipeline

```
SET SCHEMA DM_PAL;
DROP TABLE MODEL_TABLE;
CREATE COLUMN TABLE MODEL_TABLE (
    "INDEX" INTEGER,
    "CONTENT" NVARCHAR(5000)
);
INSERT INTO MODEL_TABLE VALUES(0, '[');
INSERT INTO MODEL_TABLE VALUES(1, '    {"SCALE":{ "args":{ "SCALING_METHOD":
1, "Z-SCORE_METHOD":2 }}},');
INSERT INTO MODEL_TABLE VALUES(2, '    {"CATPCA":{ "args":
{ "COMPONENTS_PERCENTAGE":0.5}}},');
INSERT INTO MODEL_TABLE VALUES(3, '    {"PolynomialFeatures":
{"args":{"MIN_DEGREE":1,"MAX_DEGREE":3, "INTERACTION_ONLY":true,
"INCLUDE_BIAS":false}}},');
INSERT INTO MODEL_TABLE VALUES(4, '    {"FS_unsupervised":{ "args":
{ "FS_METHOD":9, "TOP_K_BEST":10 }}}');
INSERT INTO MODEL_TABLE VALUES(5, ']');
DROP TABLE PAL_PARAMETER_TAB;
CREATE COLUMN TABLE PAL_PARAMETER_TAB (
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR(5000)
);
INSERT INTO PAL_PARAMETER_TAB VALUES ('EXECUTE', null, null, 'fit');
DROP TABLE UPDATED_MODEL_TABLE;
CREATE COLUMN TABLE UPDATED_MODEL_TABLE (
    "INDEX" INTEGER,
    "CONTENT" NVARCHAR(5000)
);
DROP TABLE FIT_TRANSFORM_RESULT;
CREATE COLUMN TABLE FIT_TRANSFORM_RESULT (
    "INDEX" INTEGER,
    "X0" DOUBLE,
    "X1" DOUBLE,
    "X2" DOUBLE,
    "X3" DOUBLE,
    "X4" DOUBLE,
    "X5" DOUBLE,
    "TARGET" NVARCHAR(2)
);
DO BEGIN
lt_model = SELECT * FROM MODEL_TABLE;
```

```
lt_args = SELECT * FROM PAL_PARAMETER_TAB;
lt_result = SELECT * FROM FIT_TRANSFORM_RESULT;
lt_dummy = SELECT * FROM PAL_PARAMETER_TAB;
lt_data = SELECT * FROM PAL_CLASSIFICATION_TRAIN_DATA;
CALL _SYS_AFL.PAL_PIPELINE_EXECUTE(:lt_model, :lt_args,
    :lt_data, :lt_dummy, :lt_dummy,
    :lt_result, :lt_dummy, :lt_dummy,
    lt_output_model,
    lt_output0, lt_output1, lt_output2);
INSERT INTO UPDATED_MODEL_TABLE SELECT * FROM :lt_output_model;
INSERT INTO FIT_TRANSFORM_RESULT SELECT * FROM :lt_output0;
END;
SELECT * FROM UPDATED_MODEL_TABLE;
SELECT * FROM FIT_TRANSFORM_RESULT;
DROP TABLE RESULT_TABLE;
CREATE COLUMN TABLE RESULT_TABLE (
    "INDEX" INTEGER,
    "X0" DOUBLE,
    "X1" DOUBLE,
    "X2" DOUBLE,
    "X3" DOUBLE,
    "X4" DOUBLE,
    "X5" DOUBLE
);
DROP TABLE PAL_PARAMETER_TAB;
CREATE COLUMN TABLE PAL_PARAMETER_TAB (
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR(5000)
);
INSERT INTO PAL_PARAMETER_TAB VALUES ('EXECUTE', NULL, NULL, 'predict');
DO BEGIN
lt_model = SELECT * FROM UPDATED_MODEL_TABLE;
lt_args = SELECT * FROM PAL_PARAMETER_TAB;
lt_dummy = SELECT * FROM PAL_PARAMETER_TAB;
lt_data = SELECT * FROM PAL_CLASSIFICATION_TEST_DATA;
lt_result = SELECT * FROM RESULT_TABLE;
CALL _SYS_AFL.PAL_PIPELINE_EXECUTE(:lt_model, :lt_args,
    :lt_data, :lt_dummy, :lt_dummy,
    :lt_result, :lt_dummy, :lt_dummy,
    lt_updated_model,
    lt_output0, lt_output1, lt_output2
);
INSERT INTO RESULT_TABLE SELECT * FROM :lt_output0;
END;
SELECT * FROM RESULT_TABLE;
```

**Expected Result**

- UPDATED_MODEL_TABLE

Rows (1) SQL

| | INDEX | |
|---|---|---|
| 1 | 0 | [{"SCALE":{"args":{"SCALING_METHOD":1,"Z-SCORE_METHOD":2},"model":{"nam |

- FIT_TRANSFORM_RESULT

Rows (18) SQL

| | INDEX | X0 | X1 | X2 | X3 | X4 | X5 | TARGET |
|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1.541298 | -1.747829 | -3.3114 | 3.755131 | -2.693926 | 5.7877 | 1 |
| 2 | 1 | 0.734335 | 1.221201 | -2.3851 | -3.966451 | 0.896772 | -2.912 | 0 |
| 3 | 2 | 0.913909 | -0.044520 | -3.0705 | 0.149579 | -0.040687 | 0.1367 | 1 |
| 4 | 3 | -0.528581 | 2.993931 | 0.97615 | -5.529004 | -1.582538 | 2.9225 | 0 |
| 5 | 4 | -5.527918 | -6.549593 | -2.5344 | -3.002854 | 36.20561 | 16.599 | 1 |
| 6 | 5 | 1.049540 | 1.822179 | -3.4657 | -6.017111 | 1.912450 | -6.315 | 0 |
| 7 | 6 | -1.732938 | 2.200009 | 2.75771 | -3.500987 | -3.812480 | 6.0669 | 1 |
| 8 | 8 | 1.068367 | 1.145159 | 11.4206 | 12.241498 | 1.223451 | 13.078 | 1 |
| 9 | 9 | 8.582415 | 0.057349 | -17.581 | -0.117480 | 0.492195 | -1.008 | 1 |
| 10 | 10 | 0.782645 | 1.597031 | -2.5130 | -5.128009 | 1.249909 | -4.013 | 0 |
| 11 | 11 | -0.831583 | 0.203676 | 2.71637 | -0.665312 | -0.169374 | 0.5532 | 1 |
| 12 | 12 | 3.277783 | -5.81398 | -5.7975 | 10.283500 | -19.05697 | 33.707 | 0 |
| 13 | 13 | -0.115201 | 0.444761 | -2.0250 | 7.818322 | -0.05123 | -0.900 | 1 |
| 14 | 14 | 2.037114 | -1.019825 | 3.36387 | -1.684029 | -2.077500 | -3.430 | 1 |
| 15 | 15 | -4.762903 | 0.664953 | 16.2115 | -2.263311 | -3.167109 | 10.779 | 1 |
| 16 | 18 | -4.623760 | 1.817575 | 5.93195 | -2.331821 | -8.404034 | 10.781 | 0 |
| 17 | 19 | -0.709885 | 0.537939 | 0.92427 | -0.700397 | -0.381875 | 0.4972 | 1 |
| 18 | 20 | -1.154637 | 0.469980 | -1.6183 | 0.658737 | -0.54265 | -0.760 | 0 |

- RESULT_TABLE

Rows (10) SQL

| | INDEX | X0 | X1 | X2 | X3 | X4 | X5 |
|---|---|---|---|---|---|---|---|
| 1 | 0 | -5.1184618003 | 1.9787385370 | 17.10209438 | -6.61147323 | -10.128097 | 33.840573220 |
| 2 | 1 | 0.49304781989 | 1.5251212978 | -1.61983751 | -5.01056609 | 0.7519577 | -2.4704486915 |
| 3 | 2 | -0.3295636117 | 1.8953299175 | 0.311272644 | -1.79013803 | -0.624631 | 0.5899643556 |
| 4 | 3 | -0.6501882805 | -0.8074708830 | -0.32848295 | -0.40794401 | 0.5250081 | 0.2652404192 |
| 5 | 4 | 2.90108975618 | -2.3403973283 | 8.815745488 | -7.11192997 | -6.789702 | -20.632347187 |
| 6 | 5 | -0.5449745261 | 1.2477413360 | -2.22529360 | 5.094900177 | -0.679987 | -2.7765908096 |
| 7 | 6 | 0.56382437751 | 1.7736218532 | -1.81591385 | -5.71231864 | 1.0000112 | -3.2207445037 |
| 8 | 7 | 0.00403086265 | 2.2851598878 | -0.01530260 | -8.67529035 | 0.0092111 | -0.0349689039 |
| 9 | 8 | 1.02933082550 | 0.4703177778 | -2.36537212 | -1.08077649 | 0.4841125 | -1.1124765630 |
| 10 | 9 | -1.6450675060 | 1.3716219347 | -2.90628388 | 2.423197049 | -2.256410 | -3.986322726 |

## Massive Operator

The massive operator provides the functionality of parallel execution with different operators specified based on `GROUP_ID`. The model of massive operator includes an operator map with the key `operators` and the value `<group_id, group operator model>`, which specifies the operator model for each `GROUP_ID`.

Make sure the EXECUTE method performed by each group's model has the same input and output, generally of the same type, such as all classifiers or all regressors.

```
{"operators":{"GROUP_ID_1":<sub_operator_1>, ..., "GROUP_ID_n":<sub_operator_n>}}
```

### Examples

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_MASSIVE_TRAIN_DATA;
CREATE COLUMN TABLE PAL_MASSIVE_TRAIN_DATA(
"GROUP_ID" NVARCHAR(256),
"ID" INTEGER,
"X0" INTEGER,
"X1" INTEGER,
"X2" INTEGER,
"X3" DOUBLE,
"X4" DOUBLE,
"X5" INTEGER,
"TARGET" NVARCHAR(2)
);
INSERT INTO PAL_MASSIVE_TRAIN_DATA VALUES('A',0,72,35,0,33.6,0.627,50,'1');
INSERT INTO PAL_MASSIVE_TRAIN_DATA VALUES('A',1,66,29,0,26.6,0.35103,31,'0');
INSERT INTO PAL_MASSIVE_TRAIN_DATA VALUES('A',2,64,0,0,23.3,0.672,32,'1');
INSERT INTO PAL_MASSIVE_TRAIN_DATA VALUES('A',3,66,23,94,28.1,0.166998,21,'0');
INSERT INTO PAL_MASSIVE_TRAIN_DATA VALUES('A',4,40,35,168,43.1,2.2883,33,'1');
INSERT INTO PAL_MASSIVE_TRAIN_DATA VALUES('A',5,74,0,0,25.6,0.201,30,'0');
INSERT INTO PAL_MASSIVE_TRAIN_DATA VALUES('B',6,50,32,88,31.0,0.248,26,'1');
INSERT INTO PAL_MASSIVE_TRAIN_DATA VALUES('B',8,70,45,543,30.5,0.158,53,'1');
INSERT INTO PAL_MASSIVE_TRAIN_DATA VALUES('B',9,96,0,0,0.0,0.231998,54,'1');
INSERT INTO PAL_MASSIVE_TRAIN_DATA VALUES('B',10,92,0,0,37.6,0.191,30,'0');
INSERT INTO PAL_MASSIVE_TRAIN_DATA VALUES('B',11,74,0,0,38.0,0.537,34,'1');
INSERT INTO PAL_MASSIVE_TRAIN_DATA VALUES('B',12,80,0,0,27.1,1.44098,57,'0');
INSERT INTO PAL_MASSIVE_TRAIN_DATA VALUES('C',13,60,23,846,30.1,0.397996,59,'1');
INSERT INTO PAL_MASSIVE_TRAIN_DATA VALUES('C',14,72,19,175,25.8,0.5871,51,'1');
INSERT INTO PAL_MASSIVE_TRAIN_DATA VALUES('C',15,0,0,0,30.0,0.484,32,'1');
INSERT INTO PAL_MASSIVE_TRAIN_DATA VALUES('C',18,30,38,83,43.3,0.183,33,'0');
INSERT INTO PAL_MASSIVE_TRAIN_DATA VALUES('C',19,70,30,96,34.6,0.529,32,'1');
INSERT INTO PAL_MASSIVE_TRAIN_DATA VALUES('C',20,88,41,235,39.3,0.7041,27,'0');
DROP TABLE PAL_MASSIVE_TEST_DATA;
CREATE COLUMN TABLE PAL_MASSIVE_TEST_DATA(
"GROUP_ID" NVARCHAR(256),
"ID" INTEGER,
"X0" INTEGER,
"X1" INTEGER,
"X2" INTEGER,
"X3" DOUBLE,
"X4" DOUBLE,
"X5" INTEGER
);
INSERT INTO PAL_MASSIVE_TEST_DATA VALUES('A',0,0,0,0,35.3,0.134,29);
INSERT INTO PAL_MASSIVE_TEST_DATA VALUES('A',1,74,0,0,29.6,0.254,31);
INSERT INTO PAL_MASSIVE_TEST_DATA VALUES('A',2,66,15,140,23.2,0.48704,22);
INSERT INTO PAL_MASSIVE_TEST_DATA VALUES('B',3,60,33,192,24.0,0.966,33);
INSERT INTO PAL_MASSIVE_TEST_DATA VALUES('B',4,110,24,240,45.4,0.721,54);
INSERT INTO PAL_MASSIVE_TEST_DATA VALUES('B',5,68,39,304,37.7,0.254,41);
INSERT INTO PAL_MASSIVE_TEST_DATA VALUES('B',6,74,27,0,29.0,0.203,32);
INSERT INTO PAL_MASSIVE_TEST_DATA VALUES('C',7,66,22,0,25.0,0.307,24);
INSERT INTO PAL_MASSIVE_TEST_DATA VALUES('C',8,72,54,0,36.6,0.17802,45);
INSERT INTO PAL_MASSIVE_TEST_DATA VALUES('C',9,90,51,220,49.7,0.325,31);
DROP TABLE MODEL_TABLE;
CREATE COLUMN TABLE MODEL_TABLE (
    "INDEX" INTEGER,
    "CONTENT" NVARCHAR(5000)
);
-- Corresponding classifier operators are specified for each GROUP_ID, where
group A is a classification pipeline, and groups B and C are single classifiers.
```

```
INSERT INTO MODEL_TABLE VALUES (0,
'{"Massive":{"operators":{
    "A":[{"SCALE":{"args":{"SCALING_METHOD":2}}},
         {"CATPCA":{"args":{"COMPONENTS_PERCENTAGE":0.5, "SCALING":0}}},
         {"NB_Classifier":{"args":{"LAPLACE":0.1}}}
     ],
    "B":{"NB_Classifier":{"args":{"LAPLACE":0.1}}},
    "C":{"DT_Classifier": {"args": {"ALGORITHM":1,
"CATEGORICAL_VARIABLE":"TARGET"}}}
}}}');
DROP TABLE PAL_PARAMETER_TAB;
CREATE COLUMN TABLE PAL_PARAMETER_TAB (
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR(5000)
);
INSERT INTO PAL_PARAMETER_TAB VALUES ('EXECUTE', null, null, 'fit');
DROP TABLE UPDATED_MODEL_TABLE;
CREATE COLUMN TABLE UPDATED_MODEL_TABLE (
    "INDEX" INTEGER,
    "CONTENT" NVARCHAR(5000)
);
DROP TABLE RESULT_TABLE;
CREATE COLUMN TABLE RESULT_TABLE (
    "GROUP" NVARCHAR(256),
    "ID" INTEGER,
    "SCORE" NVARCHAR(256),
    "PROB" DOUBLE
);
DO BEGIN
lt_model = SELECT * FROM MODEL_TABLE;
lt_args = SELECT * FROM PAL_PARAMETER_TAB;
lt_dummy = SELECT * FROM PAL_PARAMETER_TAB;
lt_data = SELECT * FROM PAL_MASSIVE_TRAIN_DATA;
lt_result = SELECT * FROM RESULT_TABLE;
CALL _SYS_AFL.PAL_PIPELINE_EXECUTE(:lt_model, :lt_args,
    :lt_data, :lt_dummy, :lt_dummy,
    :lt_dummy, :lt_dummy, :lt_dummy,
    lt_output_model,
    lt_outp0, lt_output1, lt_output2);
INSERT INTO UPDATED_MODEL_TABLE SELECT * FROM :lt_output_model;
END;
SELECT * FROM UPDATED_MODEL_TABLE;
--- PREDICT PROCESS
DROP TABLE RESULT_TABLE;
CREATE COLUMN TABLE RESULT_TABLE (
    "GROUP" NVARCHAR(256),
    "ID" INTEGER,
    "SCORE" NVARCHAR(256),
    "PROB" DOUBLE
);
DROP TABLE PAL_PARAMETER_TAB;
CREATE COLUMN TABLE PAL_PARAMETER_TAB (
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR(5000)
);
INSERT INTO PAL_PARAMETER_TAB VALUES ('EXECUTE', null, null, 'predict');
DO BEGIN
lt_model = SELECT * FROM UPDATED_MODEL_TABLE;
lt_args = SELECT * FROM PAL_PARAMETER_TAB;
lt_data = SELECT * FROM PAL_MASSIVE_TEST_DATA;
lt_dummy =SELECT * FROM PAL_PARAMETER_TAB;
lt_result = SELECT * FROM RESULT_TABLE;
CALL _SYS_AFL.PAL_PIPELINE_EXECUTE(:lt_model, :lt_args,
    :lt_data, :lt_dummy, :lt_dummy,
```

```
    :lt_result, :lt_dummy, :lt_dummy,
    lt_output_model,
    lt_output0, lt_output1, lt_output2);
INSERT INTO RESULT_TABLE SELECT * FROM :lt_output0;
END;
SELECT * FROM RESULT_TABLE;
```

**Expected Result**

- UPDATED_MODEL_TABLE

Rows (1) SQL

| | INDEX | |
|---|---|---|
| 1 | 0 | {"Massive":{"operators":{"A":[{"SCALE":{"args":{"SCALING_METHOD":2},"model":{"name":"MODEL","c |

- RESULT_TABLE

Rows (10) SQL

| | GROUP | ID | SCORE | PROB |
|---|---|---|---|---|
| 1 | C | 7 | 1 | 1 |
| 2 | C | 8 | 0 | 1 |
| 3 | C | 9 | 0 | 1 |
| 4 | A | 0 | 1 | 0.9999999998221603 |
| 5 | A | 1 | 0 | 0.7189680182072661 |
| 6 | A | 2 | 1 | 0.792192837862209 |
| 7 | B | 3 | 1 | 1 |
| 8 | B | 4 | 1 | 1 |
| 9 | B | 5 | 1 | 1 |
| 10 | B | 6 | 1 | 1 |

## 5.7 AutoML

PAL AutoML uses Genetic Programming Algorithm to generate optimized PAL pipelines for specified multiple optimization objectives in the giving searching space, specified by config_dict. See Configuration [page 1022] and Optimization [page 1038] for more details.

AutoML Tutorial [page 1043] demonstrates the process of using AutoML to deal with machine learning problems.

AutoML can generate the pipeline structure by specifying the type of PIPELINE_TYPE, classifier, regressor, and timeseries, corresponding to the three types supported by Pipeline, Classification, Regression, and TimeSeries.

### _SYS_AFL.PAL_AUTOML_FIT

This procedure is used to generate optimized PAL pipeline structure.

**Overview of All Tables**

| Position | Table Name | Type |
|---|---|---|
| 1 | `<Data Table>` | IN |
| 2 | `<Parameter Table>` | IN |
| 3 | `<Pipeline Table>` | OUT |
| 4 | `<Model Table>` | OUT |
| 5 | `<Statistic Table>` | OUT |

### Input Table

- For classification and regression

| Table | Column | Data Type | Description |
|---|---|---|---|
| DATA | 1st column | INTEGER, BIGINT, VARCHAR or NVARCHAR | Record ID<br>If this column is absent, the HAS_ID parameter must be set to 0. |
| | Other columns | VARCHAR, NVARCHAR, DOUBLE, DECIMAL(p, s), or INTEGER<br>For classification, dependent variable data type cannot be DOUBLE. | Attribute data and dependent variable. |

- For timeseries

| Table | Column | Data Type | Description |
|---|---|---|---|
| DATA | 1st column | INTEGER, BIGINT, TIMESTAMP, SECONDDATE, or DATE | Time stamp |
| | 2nd column | INTEGER, DOUBLE, or DECIMAL (p, s) | Target time series data |
| | Other columns | INTEGER, DOUBLE, DECIMAL (p, s), VARCHAR or NVARCHAR | External data feature |

### Parameter Table

Mandatory Parameters

| Name | Data Type | Description |
|---|---|---|
| PIPELINE_TYPE | VARCHAR or NVARCHAR | Target pipeline type:<br>• classifier<br>• regressor<br>• timeseries |

Optional Parameters

The following parameter is optional. If it is not specified, PAL will use its default value.

- Configuration related parameters, refer to Configuration [page 1022] for more details.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| CONFIG_DICT | VARCHAR or NVARCHAR | 'default' | Specifies Config Dict:<br>• 'default'<br>• 'light'<br>Only for classification and regression<br>• 'empty'<br>Fully define by parameters<br>• customized config_dict<br>Could be further modified with parameters | |
| CONFIG_REMOVE | VARCHAR or NVARCHAR | No default value | Removes the specified estimator configs in Config Dict | |
| CONFIG_ADD | VARCHAR or NVARCHAR | No default value | Adds a estimator config into Config Dict | |
| CONFIG_REPLACE | VARCHAR or NVARCHAR | No default value | Replaces the specified estimator config in Config Dict | |
| CONFIG_MODIFY | VARCHAR or NVARCHAR | No default value | Modifies the specified estimator config in Config Dict | |

- Optimization related parameters, refer to Optimization [page 1038] for more details.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| GENERATIONS | INTEGER | 5 | The number of iterations of the pipeline optimization. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| POPULATION_SIZE | INTEGER | 20 | The number of individuals in each generation in genetic programming algorithm.<br>• If there are too few individuals, GA has a few possibilities to perform crossover and only a small part of search space is explored. On the other hand, if there are too many individuals, GA slows down. | |
| OFFSPRING_SIZE | INTEGER | POPULATION_SIZE | The number of offsprings to produce in each generation.<br>• It controls the number of new individuals generated in each iteration by genetic operations, from population. | |
| MUTATION_RATE | DOUBLE | 0.9 | The mutation rate for the genetic programming algorithm.<br>• Represents the random search ability.<br>• A suitable value can prevent the GA from falling into a local optimum. | The sum of MUTATION_RATE and CROSSOVER_RATE cannot be greater than 1.0.<br>When the sum is less than 1.0, the remaining probability will be used to regenerate. |
| CROSSOVER_RATE | DOUBLE | 0.1 | The crossover rate for genetic programming algorithm.<br>• Represents the local search ability.<br>• A larger crossover rate will cause GA to converge to a local optimum faster. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| MIN_LAYER | INTEGER | 1 | The minimum number of operators in pipeline. | |
| MAX_LAYER | INTEGER | 5 | The maximum number of operators in pipeline | |
| SCORINGS | VARCHAR or NVARCHAR | '{"ACCURACY":1.0, "AUC":1.0}' for classifier;<br>'{MAE:-1.0, "EVAR":1.0}' for regressor and timeseries. | Dict of optimization target scores and weights. Represented in json format. | The direction of multi-objective optimization is maximization. If you want to minimize a score, set the weight negative. |
| MAX_EVAL_TIME_MINS | DOUBLE | 0.0 | Time limit to evaluate a single pipeline.<br>• 0.0 means not time limit. | |
| EARLY_STOP | INTEGER | 5 | Stops optimization progress when the best pipeline is not updated for the given consecutive generations.<br>• 0 means not early stop. | |
| ELITE_NUMBER | INTEGER | POPULATION_SIZE/4 | The number of elite to output into result table. | |
| OUTPUT_MODEL | INTEGER | 1 | Whether to train and output the optimal pipeline model. | |
| PROGRESS_INDICATOR_ID | VARCHAR or NVARCHAR | Empty value | Sets the ID used to output monitoring information of the optimization progress. | |
| RANDOM_SEED | INTEGER | Empty value | Specifies the seed for random number generator. Use system time if not provided. | |
| SUCCESSIVE_HALVING | INTEGER | 1 | Indicates whether to use successive halving.<br>• 0: do not use successive halving<br>• 1: use successive halving | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| MIN_BUDGET | INTEGER | (data size)/5 | Minimum data size used in successive halving iteration. | |
| MAX_BUDGET | INTEGER | data size | Maximum data size used in successive halving iteration | |
| MIN_INDIVIDUALS | DOUBLE | 3 | Minimum population size in successive halving iteration. | |

- Only valid when PIPELINE_TYPE is classifier or regressor

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| HAS_ID | INTEGER | 1 | Specifies whether the input data table has an ID column. If it exists, the ID column must be the first column.<br>• 0: No ID column<br>• 1: Has an ID column | |
| DEPENDENT_VARIABLE | VARCHAR or NVARCHAR | The last column name of the the Data table | Specifies the variable used as dependent | Cannot be the first column name if HAS_ID is 1. |
| CATEGORICAL_VARIABLE | VARCHAR or NVARCHAR | Detected from input data | Indicates which variables are treated as categorical. The default behavior is:<br>• STRING: categorical<br>• INTEGER and DOUBLE: continuous | Valid only for INTEGER variables; omitted otherwise. |
| RESAMPLING_METHOD | VARCHAR or NVARCHAR | • "stratified_cv" for classification<br>• "cv" for regression | Specifies the resampling method for pipeline evaluation.<br>• cv<br>• stratified_cv | |
| FOLD_NUM | INTEGER | 5 | Specifies the fold number for the cross validation method. | |

- Only valid when PIPELINE_TYPE is timeseries

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| SPLIT_METHOD | INTEGER | 1 | Specifies the split method for pipeline evaluation.<br>• 1: ROCV<br>• 2: Blocked | |
| FOLD_NUM | INTEGER | 5 | Specifies the fold number for the cross validation method. | |
| GAP_NUM | INTEGER | 0 | Number of samples to exclude from the end of each train set before the test set. | |
| PERCENTAGE | DOUBLE | 0.7 | Percentage between training data and test data. Only applicable to blocked type. | Only valid when SPLIT_METHOD is 2. |

### Output Tables

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| PIPELINES | 1st column | INTEGER | ID | Pipeline ID |
| | 2nd column | VARCHAR or NVARCHAR | PIPELINE | Optimized Pipeline structure |
| | 3rd column | VARCHAR or NVARCHAR | SCORES | Pipeline scores |
| MODEL | 1st column | INTEGER | ROW_INDEX | Pipeline model row index |
| | 2nd column | VARCHAR or NVARCHAR | MODEL_CONTENT | Deserialized pipeline model.<br>The table must be a column table. The maximum length of every unit (row) is 5000. |
| STATISTICS | 1st column | VARCHAR or NVARCHAR | STAT_NAME | Statistic name |
| | 2nd column | VARCHAR or NVARCHAR | STAT_VALUE | Statistic value |

### Example

Assume that:

- DM_PAL is a schema belonging to USER1;

- USER1 has been assigned the AFL__SYS_AFL_AFLPAL_EXECUTE or AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION role.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_AUTOML_DATA_TBL;
CREATE COLUMN TABLE PAL_AUTOML_DATA_TBL (
    ID INTEGER,
    ATTRIBUTE1 DOUBLE,
    ATTRIBUTE2 DOUBLE,
    ATTRIBUTE3 DOUBLE,
    ATTRIBUTE4 VARCHAR(10),
    LABEL INTEGER
);
INSERT INTO PAL_AUTOML_DATA_TBL VALUES (0,1,10,100,'A',1);
INSERT INTO PAL_AUTOML_DATA_TBL VALUES (1,1.1,10.1,100,'A',1);
INSERT INTO PAL_AUTOML_DATA_TBL VALUES (2,1.2,10.2,100,'A',1);
INSERT INTO PAL_AUTOML_DATA_TBL VALUES (3,1.3,10.4,100,'A',1);
INSERT INTO PAL_AUTOML_DATA_TBL VALUES (4,1.2,10.3,100,'AB',1);
INSERT INTO PAL_AUTOML_DATA_TBL VALUES (5,4,40,400,'AB',2);
INSERT INTO PAL_AUTOML_DATA_TBL VALUES (6,4.1,40.1,400,'AB',2);
INSERT INTO PAL_AUTOML_DATA_TBL VALUES (7,4.2,40.2,400,'AB',2);
INSERT INTO PAL_AUTOML_DATA_TBL VALUES (8,4.3,40.4,400,'AB',2);
INSERT INTO PAL_AUTOML_DATA_TBL VALUES (9,4.2,40.3,400,'AB',2);
INSERT INTO PAL_AUTOML_DATA_TBL VALUES (10,9,90,900,'B',3);
INSERT INTO PAL_AUTOML_DATA_TBL VALUES (11,9.1,90.1,900,'A',3);
INSERT INTO PAL_AUTOML_DATA_TBL VALUES (12,9.2,90.2,900,'B',3);
INSERT INTO PAL_AUTOML_DATA_TBL VALUES (13,9.3,90.4,900,'A',3);
INSERT INTO PAL_AUTOML_DATA_TBL VALUES (14,9.2,90.3,900,'A',3);
INSERT INTO PAL_AUTOML_DATA_TBL VALUES (15,1,10,100,'A',1);
INSERT INTO PAL_AUTOML_DATA_TBL VALUES (16,1.2,10.2,100,'A',1);
INSERT INTO PAL_AUTOML_DATA_TBL VALUES (17,4.1,40.1,400,'AB',2);
INSERT INTO PAL_AUTOML_DATA_TBL VALUES (18,4.2,40.3,400,'AB',2);
INSERT INTO PAL_AUTOML_DATA_TBL VALUES (19,9.1,90.1,900,'A',3);
INSERT INTO PAL_AUTOML_DATA_TBL VALUES (20,9.2,90.2,900,'A',3);
INSERT INTO PAL_AUTOML_DATA_TBL VALUES (21,4,40,400,'A',2);
DROP TABLE PAL_PARAMETER_TAB;
CREATE COLUMN TABLE PAL_PARAMETER_TAB (
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR(1000)
);
INSERT INTO PAL_PARAMETER_TAB VALUES ('PIPELINE_TYPE', null, null, 'classifier');
INSERT INTO PAL_PARAMETER_TAB VALUES ('CONFIG_DICT', null, null, 'default');
INSERT INTO PAL_PARAMETER_TAB VALUES ('CONFIG_REMOVE', null, null,
'SVM_Classifier');
INSERT INTO PAL_PARAMETER_TAB VALUES ('CONFIG_MODIFY', null, null,
'{"RDT_Classifier":{"TREES_NUM":[10, 20]}}');
INSERT INTO PAL_PARAMETER_TAB VALUES ('CONFIG_MODIFY',
null, null, '{"HGBT_Classifier":{"MAX_DEPTH":{"range":[1,1,4]},"ETA":
[0.01,0.1,0.5],"NODE_SIZE":{"range":[1,2,10]}}}');
INSERT INTO PAL_PARAMETER_TAB VALUES ('GENERATIONS', 20, null, null);
INSERT INTO PAL_PARAMETER_TAB VALUES ('POPULATION_SIZE', 10, null, null);
INSERT INTO PAL_PARAMETER_TAB VALUES ('OFFSPRING_SIZE', 20, null, null);
INSERT INTO PAL_PARAMETER_TAB VALUES ('MUTATION_RATE', null, 0.7, null);
INSERT INTO PAL_PARAMETER_TAB VALUES ('CROSSOVER_RATE', null, 0.1, null);
INSERT INTO PAL_PARAMETER_TAB VALUES ('MIN_LAYER', 2, null, null);
INSERT INTO PAL_PARAMETER_TAB VALUES ('MAX_LAYER', 5, null, null);
INSERT INTO PAL_PARAMETER_TAB VALUES ('SCORINGS', null, null, '{"ACCURACY":1.0,
"AUC":1.0, "KAPPA":1.0}');
INSERT INTO PAL_PARAMETER_TAB VALUES ('EARLY_STOP', 3, null, null);
INSERT INTO PAL_PARAMETER_TAB VALUES ('OUTPUT_MODEL', 1, null, null);
INSERT INTO PAL_PARAMETER_TAB VALUES ('ELITE_NUMBER', 3, null, null);
INSERT INTO PAL_PARAMETER_TAB VALUES ('FOLD_NUM', 5, null, null);
INSERT INTO PAL_PARAMETER_TAB VALUES ('HAS_ID', 1, null, null);
INSERT INTO PAL_PARAMETER_TAB VALUES ('RANDOM_SEED', 1, null, null);
```

```
INSERT INTO PAL_PARAMETER_TAB VALUES ('PROGRESS_INDICATOR_ID', null, null,
'AUTOML_TEST');
CALL _SYS_AFL.PAL_AUTOML_FIT(PAL_AUTOML_DATA_TBL, PAL_PARAMETER_TAB, ?, ?, ?);
```

**Expected Result**

PIPELINES:

Rows (3)

| | ID | PIPELINE | SCORES |
|---|---|---|---|
| 1 | 0 | {"NB_Classifier":{"args":{"LAPLACE":100.0},"input | {"ACCURACY":1.0,"AUC":1.0,"KAPPA":1.0,"LAYERS":4.0,"MCC":1.0} |
| 2 | 1 | {"NB_Classifier":{"args":{"LAPLACE":100.0},"input | {"ACCURACY":1.0,"AUC":1.0,"KAPPA":1.0,"LAYERS":3.0,"MCC":1.0} |
| 3 | 2 | {"NB_Classifier":{"args":{"LAPLACE":0.01},"inputs' | {"ACCURACY":1.0,"AUC":1.0,"KAPPA":1.0,"LAYERS":2.0,"MCC":1.0} |

MODEL:

Rows (1)

| | ROW_INDEX | |
|---|---|---|
| 1 | 0 | {"pipeline":{"NB_Classifier":{"args":{"LAPLACE":100.0},"model":{"L |

## Progress Status Monitor

AutoML uses AFL progress indicator mechanism to output log information at runtime.

When we set the PROGRESS_INDICATOR_ID for PAL_AUTOML_FIT procedure, we can obtain the corresponding log information in another session.

**Example**

When PAL_AUTOML_FIT in running, use the following SQL script to obtain log information in another HANA session.

```
SET SCHEMA PAL_USER;
SELECT * FROM _SYS_AFL.FUNCTION_PROGRESS_IN_AFLPAL WHERE EXECUTION_ID
='AUTOML_TEST';
```

Since PROGRESS_MESSAGE has a length limit, to output the complete log information, information that exceeds the limit will be cut into multiple PROGRESS_MESSAGE, and PROGRESS_CURRENT is used as the id of log information, that is, several lines of PROGRESS_MESSAGE with the same PROGRESS_CURRENT form a complete log information.

In the following example

Log Information:

Rows (33)

| | EXECUTION_ID | | PROGRESS_CURRENT | | | PROGRESS_MESSAGE |
|---|---|---|---|---|---|---|
| 1 | AUTOML_TEST | 10 | 0 | 1 | n/a | {"pipeline_num":410} |
| 2 | AUTOML_TEST | 77 | 1 | 1 | n/a | {"generation_num":20} |
| 3 | AUTOML_TEST | 11 | 2 | 1 | n/a | {"generation":0} |
| 4 | AUTOML_TEST | 14 | 3 | 1 | n/a | {"evaluating":{"M_LOGR_Classifier":{"args":{"MAX_ITERATION":100,"ENET_ALPHA":0.0,"E |
| 5 | AUTOML_TEST | 14 | 3 | 1 | n/a | :3,"METHOD":1,"SAMPLING_STRATEGY":1},"inputs":{"data":{"SAMPLING":{"args":{"PERCE |
| 6 | AUTOML_TEST | 27 | 4 | 1 | n/a | {"pipeline_evaluated":{"state":"failed","error_msg":"PAL error[73002108]:wrong: cannot ma |
| 7 | AUTOML_TEST | 27 | 4 | 1 | n/a | "TomekLinks":{"args":{"SAMPLING_STRATEGY":0,"DISTANCE_LEVEL":2,"METHOD":1},"inp |
| 8 | AUTOML_TEST | 27 | 4 | 1 | n/a | s":{"data":"ROWDATA"}}}}}}}}}}}}} |
| 9 | AUTOML_TEST | 36 | 5 | 1 | n/a | {"evaluating":{"NB_Classifier":{"args":{"LAPLACE":100.0},"inputs":{"data":{"SMOTE":{"args" |
| 10 | AUTOML_TEST | 36 | 5 | 1 | n/a | {"args":{"FS_METHOD":9,"HAS_ID":1,"TOP_K_BEST":10},"inputs":{"data":"ROWDATA"}}}}}} |
| 11 | AUTOML_TEST | 54 | 6 | 1 | n/a | {"pipeline_evaluated":{"state":"successed","scores":{"ACCURACY":1.0,"AUC":1.0,"KAPPA": |
| 12 | AUTOML_TEST | 54 | 6 | 1 | n/a | :{"SAMPLING":{"args":{"PERCENTAGE":0.8999999999999999,"SAMPLING_METHOD":0}," |
| 13 | AUTOML_TEST | 61 | 7 | 1 | n/a | {"evaluating":{"M_LOGR_Classifier":{"args":{"MAX_ITERATION":100,"ENET_ALPHA":0.0,"E |
| 14 | AUTOML_TEST | 69 | 8 | 1 | n/a | {"pipeline_evaluated":{"state":"successed","scores":{"ACCURACY":0.96,"AUC":0.99600000 |
| 15 | AUTOML_TEST | 69 | 8 | 1 | n/a | "inputs":{"data":"ROWDATA"}}}}} |
| 16 | AUTOML_TEST | 76 | 9 | 1 | n/a | {"evaluating":{"RDT_Classifier":{"args":{"TREES_NUM":20,"NODE_SIZE":18,"CALCULATE_ |
| 17 | AUTOML_TEST | 92 | 10 | 1 | n/a | {"pipeline_evaluated":{"state":"successed","scores":{"ACCURACY":0.18,"AUC":0.35750000 |
| 18 | AUTOML_TEST | 92 | 10 | 1 | n/a | },"inputs":{"data":"ROWDATA"}}}}} |
| 19 | AUTOML_TEST | 99 | 11 | 1 | n/a | {"evaluating":{"RDT_Classifier":{"args":{"TREES_NUM":20,"NODE_SIZE":18,"CALCULATE_ |
| 20 | AUTOML_TEST | 99 | 11 | 1 | n/a | {"args":{"K_NEAREST_NEIGHBOURS":3,"METHOD":1},"inputs":{"data":"ROWDATA"}}}}}}}}} |

## 5.7.1 Configuration

Config Dict specifies the operators available to construct the pipeline and their corresponding parameter ranges.

It is a nested dictionary format JSON, where the first level key is the name of the operator (for example, HGBT_Classifier, SCALE, FS_superivised) and the second level key is the corresponding parameter name for the operator (for example, FS_METHOD). The second level value is the range of that parameter.

AutoML provides `default` and `light` built-in config_dict for Classification and Regression PIPELINE_TYPE, and `default` for TimeSeries. Users can use built-in Config Dicts, or provide custom Config Dict by defining directly or modifying from built-in Config Dicts by the giving parameters.

Config Type

Config is a list of parameter values, or a range that can be converted to a list. Each value in the list represents an alternative option for that parameter.

- List

```
{"NAME":[(OPTION_0), (OPTION_1), ..., (OPTION_n)]}
```

The four parameter types of operator in pipeline, their corresponding list form config representations are as follows.

| Parameter Type | Parameter Example | Config Example |
|---|---|---|
| single row of single type | `1` | `[1, 2, 3 ,4]` |

| Parameter Type | Parameter Example | Config Example |
| --- | --- | --- |
| multiple rows of single type | `[1,2]` | `[[1,2 ], [2, 3], [3, 4]]` |
| single row of multiple types | `[1, "A"]` | `[[1, "A"], [2, "B"], [3, "C"]]` |
| multiples row of multiple types | `[1, "A"], [2, "B"]]` | `[[[1, "A"], [2, "B"]], [[2, "B"], [3, "C"]]]` |

- Range
  For parameters single row of int or double, they can be represented by the following range format.

```
{"NAME":{"range":[min_value, step, max_value]}}
```

List converted from the above range:

```
{"NAME":[min_value, min_value+step*1,..., min_value+step*n, ..., max_value(may
not included)]}
```

| Example | Converted List |
| --- | --- |
| `{"range":{0, 2, 8}}` | `[0, 2, 4, 6, 8]` |
| `{"range":{0, 2, 11}}` | `[0, 2, 4, 6, 8, 10]` |
| `{"range":{0.0, 0.5, 3.0}}` | `[0.0, 0.5, 1.0, 1.5, 2.0]` |
| `{"range":{0.0, 2.0, 7.0}}` | `[0.0, 2.0, 4.0, 6.0]` |

Notice: min_value, step, max_value have to be represented with the same type.

## _SYS_AFL.PAL_AUTOML_CONFIG

- This procedure is used to check, visualize, or save config_dict.
  Optional Parameters
  The following parameter is optional. If it is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| CONFIG_DICT | VARCHAR or NVARCHAR | 'default' | Specifies Config Dict:<br>• 'default'<br>• 'light'<br>Only for classification and regression<br>• 'empty'<br>Fully define by parameters<br>• customized config_dict<br>Could be further modified with parameters |
| CONFIG_REMOVE | VARCHAR or NVARCHAR | No default value | Removes the specified estimator configs in Config Dict |
| CONFIG_ADD | VARCHAR or NVARCHAR | No default value | Adds a estimator config into Config Dict |
| CONFIG_REPLACE | VARCHAR or NVARCHAR | No default value | Replaces the specified estimator config in Config Dict |
| CONFIG_MODIFY | VARCHAR or NVARCHAR | No default value | Modifies the specified estimator config in Config Dict |

Users can use the same parameters in PAL_AUTOML_FIT to define config dict.

The execution order of these parameters in the Config Dict Customizing process is:

1. CONFIG_DICT
2. CONFIG_REMOVE
3. CONFIG_ADD
4. CONFIG_REPLACE
5. CONFIG_MODIFY

### Output Tables

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| RESULT | 1st column | INTEGER | ROW_INDEX | Config Dict row index |
| | 2nd column | VARCHAR or NVARCHAR | CONTENT | Config Dict JSON content.<br>The table must be a column table. The maximum length of every unit (row) is 5000. |
| INFO | 1st column | VARCHAR or NVARCHAR | NAME | Operator name |
| | 2nd column | VARCHAR or NVARCHAR | TYPE | Operator type |

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| | 2nd column | VARCHAR or NVARCHAR | CONFIG | Operator config |

### Examples

Assume that:

- DM_PAL is a schema belonging to USER1;
- USER1 has been assigned the AFL__SYS_AFL_AFLPAL_EXECUTE or AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION role.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_PARAMETER_TAB;
CREATE COLUMN TABLE PAL_PARAMETER_TAB (
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR(5000)
);
INSERT INTO PAL_PARAMETER_TAB VALUES ('PIPELINE_TYPE', null, null, 'regressor');
INSERT INTO PAL_PARAMETER_TAB VALUES ('CONFIG_DICT', null, null, 'default');
INSERT INTO PAL_PARAMETER_TAB VALUES ('CONFIG_REMOVE', null, null,
'EXP_Regressor');
INSERT INTO PAL_PARAMETER_TAB VALUES ('CONFIG_REMOVE', null, null,
'RDT_Regressor');
INSERT INTO PAL_PARAMETER_TAB VALUES
('CONFIG_ADD', null, null, '{"RDT_Regressor":{"TREES_NUM":
[100],"NODE_SIZE":{"range":[1,1,21]},"SPLIT_THRESHOLD":[1e-9],"CALCULATE_OOB":
[0],"SAMPLE_FRACTION":[0.75,1.0]}}');
INSERT INTO PAL_PARAMETER_TAB VALUES
('CONFIG_REPLACE', null, null, '{"RDT_Regressor":
{"TREES_NUM":[100, 200],"NODE_SIZE":{"range":[1,1,21]},"SPLIT_THRESHOLD":
[1e-9],"CALCULATE_OOB":[0],"SAMPLE_FRACTION":[0.75,1.0]}}');
INSERT INTO PAL_PARAMETER_TAB VALUES ('CONFIG_MODIFY', null, null,
'{"RDT_Regressor":{"TREES_NUM":[100]}}');
CALL _SYS_AFL.PAL_AUTOML_CONFIG(PAL_PARAMETER_TAB, ?, ?);
```

### Expected Result

RESULT:

Rows (1) SQL

| | ROW_INDEX | CONTENT |
|---|---|---|
| 1 | 0 | {"LabelEncoder":{"IGNORE_UNKNOWN":[1]}," |

INFO:

Rows (17)

| | NAME | CATEGORY | |
|---|---|---|---|
| 1 | RDT_Regressor | Regressor | {"TREES_NUM":[100],"NODE_SIZE":{"range":[1,1,21]},"SPLIT_THRESHOLD":[ |
| 2 | DT_Regressor | Regressor | {"ALGORITHM":[3],"MAX_DEPTH":{"range":[1,1,11]},"MIN_RECORDS_OF_PA |
| 3 | POL_Regressor | Regressor | {"ALG":[2],"POLYNOMIAL_NUM":[1,2,5,8,10]} |
| 4 | MLR_Regressor | Regressor | {"ALG":[2,4],"MAX_ITERATION":[100000],"THRESHOLD":[1e-7,0.00001,0.001 |
| 5 | MLP_Regressor | Regressor | {"ACTIVATION":[13],"OUTPUT_ACTIVATION":[13],"HIDDEN_LAYER_SIZE":["1 |
| 6 | LOG_Regressor | Regressor | {"ALG":[2]} |
| 7 | HGBT_Regressor | Regressor | {"ITER_NUM":[100],"OBJ_FUNC":[0,1,2],"ETA":[0.001,0.01,0.1,0.5,1.0],"MAX_ |
| 8 | GLM_Regressor | Regressor | {"SOLVER":["irls","cd"],"FAMILY":["gaussian","poisson","gamma","inversegaussi |
| 9 | GEO_Regressor | Regressor | {"ALG":[2]} |
| 10 | SVM_Regressor | Regressor | {"TYPE":[2],"KERNEL_TYPE":[0],"ERROR_TOL":[0.00001,0.0001,0.001,0.01,0 |
| 11 | SAMPLING | Resampler | {"SAMPLING_METHOD":[0,1,2,3,4,5,6],"PERCENTAGE":{"range":[0.1,0.1,1.0] |
| 12 | FS_unsupervised | Transformer | {"FS_METHOD":[9],"TOP_K_BEST":[10]} |
| 13 | FS_supervised | Transformer | {"FS_METHOD":[0,1,2,3,4,5,6,7,11,12,13],"TOP_K_BEST":[10]} |
| 14 | CATPCA | Transformer | {"COMPONENTS_PERCENTAGE":[0.1,0.2,0.5,0.7,1.0],"SCALING":[1],"COMP |
| 15 | SCALE | Transformer | {"SCALING_METHOD":[0,1,2],"NEW_MIN":{"range":[0.0,1.0,4.0]},"NEW_RAN |
| 16 | OneHotEncoder | Transformer | {"MINIMUM_FRACTION":[0.05,0.1,0.15,0.2,0.25],"IGNORE_UNKNOWN":[1]} |
| 17 | LabelEncoder | Transformer | {"IGNORE_UNKNOWN":[1]} |

## 5.7.2 Tutorial Dataset

This is a dataset for classification, which contains an ID column, 6 FEATURES columns and a TARGET column. Details about data table are as follows:

### Training Data

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_AUTOML_TRAIN_TAB;
CREATE COLUMN TABLE PAL_AUTOML_TRAIN_TAB(
"ID" INTEGER,
"X0" INTEGER,
"X1" INTEGER,
"X2" INTEGER,
"X3" DOUBLE,
"X4" DOUBLE,
"X5" INTEGER,
"TARGET" INTEGER
);
insert into PAL_AUTOML_TRAIN_TAB values(0,72,35,0,33.6,0.627,50,1);
insert into PAL_AUTOML_TRAIN_TAB values(1,66,29,0,26.6,0.35103,31,0);
insert into PAL_AUTOML_TRAIN_TAB values(2,64,0,0,23.3,0.672,32,1);
insert into PAL_AUTOML_TRAIN_TAB values(3,66,23,94,28.1,0.166998,21,0);
insert into PAL_AUTOML_TRAIN_TAB values(4,40,35,168,43.1,2.2883,33,1);
insert into PAL_AUTOML_TRAIN_TAB values(5,74,0,0,25.6,0.201,30,0);
```

```
insert into PAL_AUTOML_TRAIN_TAB values(6,50,32,88,31.0,0.248,26,1);
insert into PAL_AUTOML_TRAIN_TAB values(8,70,45,543,30.5,0.158,53,1);
insert into PAL_AUTOML_TRAIN_TAB values(9,96,0,0,0.0,0.231998,54,1);
insert into PAL_AUTOML_TRAIN_TAB values(10,92,0,0,37.6,0.191,30,0);
insert into PAL_AUTOML_TRAIN_TAB values(11,74,0,0,38.0,0.537,34,1);
insert into PAL_AUTOML_TRAIN_TAB values(12,80,0,0,27.1,1.44098,57,0);
insert into PAL_AUTOML_TRAIN_TAB values(13,60,23,846,30.1,0.397996,59,1);
insert into PAL_AUTOML_TRAIN_TAB values(14,72,19,175,25.8,0.5871,51,1);
insert into PAL_AUTOML_TRAIN_TAB values(15,0,0,0,30.0,0.484,32,1);
insert into PAL_AUTOML_TRAIN_TAB values(18,30,38,83,43.3,0.183,33,0);
insert into PAL_AUTOML_TRAIN_TAB values(19,70,30,96,34.6,0.529,32,1);
insert into PAL_AUTOML_TRAIN_TAB values(20,88,41,235,39.3,0.7041,27,0);
insert into PAL_AUTOML_TRAIN_TAB values(21,84,0,0,35.4,0.387996,50,0);
insert into PAL_AUTOML_TRAIN_TAB values(22,90,0,0,39.8,0.451,41,1);
insert into PAL_AUTOML_TRAIN_TAB values(23,80,35,0,29.0,0.263,29,1);
insert into PAL_AUTOML_TRAIN_TAB values(24,94,33,146,36.6,0.254,51,1);
insert into PAL_AUTOML_TRAIN_TAB values(25,70,26,115,31.1,0.205,41,1);
insert into PAL_AUTOML_TRAIN_TAB values(26,76,0,0,39.4,0.257,43,1);
insert into PAL_AUTOML_TRAIN_TAB values(28,82,19,110,22.2,0.245,57,0);
insert into PAL_AUTOML_TRAIN_TAB values(29,92,0,0,34.1,0.337,38,0);
insert into PAL_AUTOML_TRAIN_TAB values(30,75,26,0,36.0,0.546,60,0);
insert into PAL_AUTOML_TRAIN_TAB values(31,76,36,245,31.6,0.851,28,1);
insert into PAL_AUTOML_TRAIN_TAB values(33,92,0,0,19.9,0.188,28,0);
insert into PAL_AUTOML_TRAIN_TAB values(34,78,31,0,27.6,0.512,45,0);
insert into PAL_AUTOML_TRAIN_TAB values(36,76,0,0,33.2,0.42,35,0);
insert into PAL_AUTOML_TRAIN_TAB values(37,76,37,0,32.9,0.665,46,1);
insert into PAL_AUTOML_TRAIN_TAB values(38,68,42,0,38.2,0.503,27,1);
insert into PAL_AUTOML_TRAIN_TAB values(39,72,47,207,37.1,1.39,56,1);
insert into PAL_AUTOML_TRAIN_TAB values(40,64,25,70,34.0,0.271,26,0);
insert into PAL_AUTOML_TRAIN_TAB values(41,84,0,0,40.2,0.696,37,0);
insert into PAL_AUTOML_TRAIN_TAB values(42,92,18,0,22.7,0.235,48,0);
insert into PAL_AUTOML_TRAIN_TAB values(44,64,0,0,27.4,0.294,40,0);
insert into PAL_AUTOML_TRAIN_TAB values(45,66,39,0,42.0,1.893,25,1);
insert into PAL_AUTOML_TRAIN_TAB values(46,56,0,0,29.7,0.564,29,0);
insert into PAL_AUTOML_TRAIN_TAB values(47,70,27,0,28.0,0.586,22,0);
insert into PAL_AUTOML_TRAIN_TAB values(48,66,32,0,39.1,0.344,31,1);
insert into PAL_AUTOML_TRAIN_TAB values(49,0,0,0,0.0,0.305,24,0);
insert into PAL_AUTOML_TRAIN_TAB values(50,80,11,82,19.4,0.491,22,0);
insert into PAL_AUTOML_TRAIN_TAB values(51,50,15,36,24.2,0.526,26,0);
insert into PAL_AUTOML_TRAIN_TAB values(52,66,21,23,24.4,0.342,30,0);
insert into PAL_AUTOML_TRAIN_TAB values(53,90,34,300,33.7,0.467,58,1);
insert into PAL_AUTOML_TRAIN_TAB values(55,50,10,0,23.0,0.248,21,0);
insert into PAL_AUTOML_TRAIN_TAB values(57,88,60,110,46.8,0.9621,31,0);
insert into PAL_AUTOML_TRAIN_TAB values(58,82,0,0,40.5,1.781,44,0);
insert into PAL_AUTOML_TRAIN_TAB values(59,64,41,142,41.5,0.17301,22,0);
insert into PAL_AUTOML_TRAIN_TAB values(60,0,0,0,0.0,0.304,21,0);
insert into PAL_AUTOML_TRAIN_TAB values(61,72,0,0,32.9,0.27,39,1);
insert into PAL_AUTOML_TRAIN_TAB values(62,62,0,0,25.0,0.5871,36,0);
insert into PAL_AUTOML_TRAIN_TAB values(63,58,34,128,25.4,0.6991,24,0);
insert into PAL_AUTOML_TRAIN_TAB values(64,66,0,0,32.8,0.258,42,1);
insert into PAL_AUTOML_TRAIN_TAB values(66,88,30,0,32.5,0.855,38,1);
insert into PAL_AUTOML_TRAIN_TAB values(67,92,0,0,42.7,0.845,54,0);
insert into PAL_AUTOML_TRAIN_TAB values(68,66,13,38,19.6,0.333996,25,0);
insert into PAL_AUTOML_TRAIN_TAB values(69,85,27,100,28.9,0.188997,27,0);
insert into PAL_AUTOML_TRAIN_TAB values(70,66,20,90,32.9,0.867,28,1);
insert into PAL_AUTOML_TRAIN_TAB values(71,64,35,140,28.6,0.41103,26,0);
insert into PAL_AUTOML_TRAIN_TAB values(72,90,0,0,43.4,0.583,42,1);
insert into PAL_AUTOML_TRAIN_TAB values(74,75,30,0,32.0,0.396,22,0);
insert into PAL_AUTOML_TRAIN_TAB values(75,48,20,0,24.7,0.14,22,0);
insert into PAL_AUTOML_TRAIN_TAB values(77,72,33,0,37.7,0.37,27,0);
insert into PAL_AUTOML_TRAIN_TAB values(78,0,0,0,43.2,0.27,26,1);
insert into PAL_AUTOML_TRAIN_TAB values(80,44,13,0,22.4,0.14,22,0);
insert into PAL_AUTOML_TRAIN_TAB values(81,0,0,0,0.0,0.102,22,0);
insert into PAL_AUTOML_TRAIN_TAB values(82,78,26,71,29.3,0.767,36,0);
insert into PAL_AUTOML_TRAIN_TAB values(83,65,28,0,24.6,0.237,22,0);
insert into PAL_AUTOML_TRAIN_TAB values(84,108,0,0,48.8,0.226998,37,1);
insert into PAL_AUTOML_TRAIN_TAB values(87,68,25,71,38.5,0.324,26,0);
insert into PAL_AUTOML_TRAIN_TAB values(88,70,32,110,37.1,0.153,43,1);
```

```
insert into PAL_AUTOML_TRAIN_TAB values(89,68,19,0,26.5,0.165,24,0);
insert into PAL_AUTOML_TRAIN_TAB values(90,55,0,0,19.1,0.258,21,0);
insert into PAL_AUTOML_TRAIN_TAB values(91,80,15,176,32.0,0.442995,34,0);
insert into PAL_AUTOML_TRAIN_TAB values(92,78,40,48,46.7,0.261,42,0);
insert into PAL_AUTOML_TRAIN_TAB values(93,72,0,0,23.8,0.276997,60,1);
insert into PAL_AUTOML_TRAIN_TAB values(95,72,27,228,33.9,0.255,40,0);
insert into PAL_AUTOML_TRAIN_TAB values(97,48,18,76,20.4,0.322995,22,0);
insert into PAL_AUTOML_TRAIN_TAB values(98,50,30,64,28.7,0.35604,23,0);
insert into PAL_AUTOML_TRAIN_TAB values(100,72,0,0,39.0,1.222,33,1);
insert into PAL_AUTOML_TRAIN_TAB values(102,96,0,0,22.5,0.262,21,0);
insert into PAL_AUTOML_TRAIN_TAB values(104,65,0,0,39.6,0.93,27,0);
insert into PAL_AUTOML_TRAIN_TAB values(105,56,29,152,28.7,0.80099,21,0);
insert into PAL_AUTOML_TRAIN_TAB values(107,58,28,140,29.5,0.287,37,0);
insert into PAL_AUTOML_TRAIN_TAB values(108,58,31,18,34.3,0.336,25,0);
insert into PAL_AUTOML_TRAIN_TAB values(109,85,25,36,37.4,0.247,24,1);
insert into PAL_AUTOML_TRAIN_TAB values(110,72,33,135,33.3,0.198998,24,1);
insert into PAL_AUTOML_TRAIN_TAB values(113,62,0,0,34.0,0.391,25,0);
insert into PAL_AUTOML_TRAIN_TAB values(115,92,0,0,31.2,0.539,61,1);
insert into PAL_AUTOML_TRAIN_TAB values(116,74,0,0,34.0,0.22,38,1);
insert into PAL_AUTOML_TRAIN_TAB values(117,48,0,0,33.7,0.654,25,0);
insert into PAL_AUTOML_TRAIN_TAB values(118,60,23,0,28.2,0.442995,22,0);
insert into PAL_AUTOML_TRAIN_TAB values(119,76,15,51,23.2,0.223,21,0);
insert into PAL_AUTOML_TRAIN_TAB values(122,74,30,100,33.6,0.403997,23,0);
insert into PAL_AUTOML_TRAIN_TAB values(123,80,0,0,26.8,0.18603,69,0);
insert into PAL_AUTOML_TRAIN_TAB values(124,76,0,0,33.3,0.278,23,1);
insert into PAL_AUTOML_TRAIN_TAB values(125,30,42,99,55.0,0.496,26,1);
insert into PAL_AUTOML_TRAIN_TAB values(126,70,30,135,42.9,0.452,30,0);
insert into PAL_AUTOML_TRAIN_TAB values(127,58,36,94,33.3,0.261,23,0);
insert into PAL_AUTOML_TRAIN_TAB values(128,88,24,145,34.5,0.402997,40,1);
insert into PAL_AUTOML_TRAIN_TAB values(129,84,0,0,27.9,0.741,62,1);
insert into PAL_AUTOML_TRAIN_TAB values(130,70,14,168,29.7,0.361,33,1);
insert into PAL_AUTOML_TRAIN_TAB values(131,56,0,0,33.3,1.114,33,1);
insert into PAL_AUTOML_TRAIN_TAB values(132,64,37,225,34.5,0.35604,30,1);
insert into PAL_AUTOML_TRAIN_TAB values(133,74,31,0,38.3,0.457,39,0);
insert into PAL_AUTOML_TRAIN_TAB values(134,68,13,49,21.1,0.647,26,0);
insert into PAL_AUTOML_TRAIN_TAB values(135,60,20,140,33.8,0.08801,31,0);
insert into PAL_AUTOML_TRAIN_TAB values(136,70,26,50,30.8,0.597,21,0);
insert into PAL_AUTOML_TRAIN_TAB values(137,60,25,92,28.7,0.532,22,0);
insert into PAL_AUTOML_TRAIN_TAB values(138,80,0,0,31.2,0.703,29,0);
insert into PAL_AUTOML_TRAIN_TAB values(139,72,29,325,36.9,0.159,28,0);
insert into PAL_AUTOML_TRAIN_TAB values(143,66,0,0,32.4,0.272,42,1);
insert into PAL_AUTOML_TRAIN_TAB values(144,62,31,284,32.8,0.237,23,0);
insert into PAL_AUTOML_TRAIN_TAB values(145,75,23,0,0.0,0.5721,21,0);
insert into PAL_AUTOML_TRAIN_TAB values(146,80,37,0,32.8,0.096,41,0);
insert into PAL_AUTOML_TRAIN_TAB values(147,64,35,119,30.5,1.4,34,0);
insert into PAL_AUTOML_TRAIN_TAB values(149,70,17,0,27.3,0.085,22,0);
insert into PAL_AUTOML_TRAIN_TAB values(150,74,50,204,37.4,0.398997,24,0);
insert into PAL_AUTOML_TRAIN_TAB values(151,65,0,0,21.9,0.43205,37,0);
insert into PAL_AUTOML_TRAIN_TAB values(154,78,0,0,47.9,0.136998,43,1);
insert into PAL_AUTOML_TRAIN_TAB values(155,88,44,0,50.0,0.337,36,1);
insert into PAL_AUTOML_TRAIN_TAB values(156,52,15,94,24.6,0.637,21,0);
insert into PAL_AUTOML_TRAIN_TAB values(157,56,21,135,25.2,0.833,23,0);
insert into PAL_AUTOML_TRAIN_TAB values(158,74,19,53,29.0,0.228998,22,0);
insert into PAL_AUTOML_TRAIN_TAB values(159,72,41,114,40.9,0.8171,47,1);
insert into PAL_AUTOML_TRAIN_TAB values(160,90,38,0,29.7,0.294,36,0);
insert into PAL_AUTOML_TRAIN_TAB values(161,74,40,105,37.2,0.204,45,0);
insert into PAL_AUTOML_TRAIN_TAB values(162,80,34,285,44.2,0.166998,27,0);
insert into PAL_AUTOML_TRAIN_TAB values(163,64,23,0,29.7,0.368,21,0);
insert into PAL_AUTOML_TRAIN_TAB values(164,88,0,0,31.6,0.743,32,1);
insert into PAL_AUTOML_TRAIN_TAB values(167,68,0,0,29.6,0.7091,34,0);
insert into PAL_AUTOML_TRAIN_TAB values(168,66,0,0,31.9,0.47103,29,0);
insert into PAL_AUTOML_TRAIN_TAB values(169,90,12,78,28.4,0.495,29,0);
insert into PAL_AUTOML_TRAIN_TAB values(170,82,0,0,30.8,0.18,36,1);
insert into PAL_AUTOML_TRAIN_TAB values(171,70,23,130,35.4,0.542,29,1);
insert into PAL_AUTOML_TRAIN_TAB values(172,0,23,0,28.9,0.773,25,0);
insert into PAL_AUTOML_TRAIN_TAB values(175,72,42,130,32.7,0.7191,36,1);
insert into PAL_AUTOML_TRAIN_TAB values(176,78,0,0,31.2,0.382,42,0);
insert into PAL_AUTOML_TRAIN_TAB values(177,110,46,130,67.1,0.319,26,1);
```

```
insert into PAL_AUTOML_TRAIN_TAB values(178,78,0,0,45.0,0.19,47,0);
insert into PAL_AUTOML_TRAIN_TAB values(179,82,0,0,39.1,0.956,37,1);
insert into PAL_AUTOML_TRAIN_TAB values(180,80,0,0,23.2,0.084,32,0);
insert into PAL_AUTOML_TRAIN_TAB values(181,64,18,92,34.9,0.725,23,0);
insert into PAL_AUTOML_TRAIN_TAB values(182,74,20,23,27.7,0.299,21,0);
insert into PAL_AUTOML_TRAIN_TAB values(183,60,0,0,26.8,0.268,27,0);
insert into PAL_AUTOML_TRAIN_TAB values(185,68,28,0,35.9,0.745,41,1);
insert into PAL_AUTOML_TRAIN_TAB values(186,68,36,495,30.1,0.615,60,1);
insert into PAL_AUTOML_TRAIN_TAB values(188,76,39,114,27.9,0.64,31,1);
insert into PAL_AUTOML_TRAIN_TAB values(189,80,35,160,31.6,0.361,25,1);
insert into PAL_AUTOML_TRAIN_TAB values(190,62,0,0,22.6,0.142,21,0);
insert into PAL_AUTOML_TRAIN_TAB values(191,70,44,94,33.1,0.374,40,0);
insert into PAL_AUTOML_TRAIN_TAB values(192,66,0,0,30.4,0.382995,36,1);
insert into PAL_AUTOML_TRAIN_TAB values(193,0,0,0,52.3,0.578,40,1);
insert into PAL_AUTOML_TRAIN_TAB values(194,55,20,0,24.4,0.136,42,0);
insert into PAL_AUTOML_TRAIN_TAB values(195,84,41,210,39.4,0.395,29,1);
insert into PAL_AUTOML_TRAIN_TAB values(196,58,0,0,24.3,0.187,21,0);
insert into PAL_AUTOML_TRAIN_TAB values(198,64,44,99,34.8,0.905,26,1);
insert into PAL_AUTOML_TRAIN_TAB values(199,60,27,318,30.9,0.15,29,1);
insert into PAL_AUTOML_TRAIN_TAB values(200,80,16,0,31.0,0.8741,21,0);
insert into PAL_AUTOML_TRAIN_TAB values(201,82,0,0,40.1,0.23602,28,0);
insert into PAL_AUTOML_TRAIN_TAB values(202,68,20,0,27.3,0.787,32,0);
insert into PAL_AUTOML_TRAIN_TAB values(203,70,16,44,20.4,0.235,27,0);
insert into PAL_AUTOML_TRAIN_TAB values(204,72,32,190,37.7,0.324,55,0);
insert into PAL_AUTOML_TRAIN_TAB values(205,72,28,0,23.9,0.40703,27,0);
insert into PAL_AUTOML_TRAIN_TAB values(207,104,0,0,37.7,0.151,52,1);
insert into PAL_AUTOML_TRAIN_TAB values(210,60,22,0,27.7,0.29,25,0);
insert into PAL_AUTOML_TRAIN_TAB values(211,85,54,0,42.8,0.375,24,0);
insert into PAL_AUTOML_TRAIN_TAB values(212,95,31,0,34.2,0.163998,60,0);
insert into PAL_AUTOML_TRAIN_TAB values(213,65,26,130,42.6,0.431,24,1);
insert into PAL_AUTOML_TRAIN_TAB values(214,82,32,175,34.2,0.26,36,1);
insert into PAL_AUTOML_TRAIN_TAB values(215,70,40,271,41.8,0.742,38,1);
insert into PAL_AUTOML_TRAIN_TAB values(216,62,41,129,35.8,0.514,25,1);
insert into PAL_AUTOML_TRAIN_TAB values(217,68,30,120,30.0,0.463997,32,0);
insert into PAL_AUTOML_TRAIN_TAB values(218,74,22,0,29.0,1.224,32,1);
insert into PAL_AUTOML_TRAIN_TAB values(220,60,29,478,34.6,1.072,21,1);
insert into PAL_AUTOML_TRAIN_TAB values(221,90,0,0,31.6,0.805,66,1);
insert into PAL_AUTOML_TRAIN_TAB values(222,0,0,0,25.2,0.209,37,0);
insert into PAL_AUTOML_TRAIN_TAB values(223,60,33,190,28.8,0.687,61,0);
insert into PAL_AUTOML_TRAIN_TAB values(225,78,27,32,34.6,0.100999,22,0);
insert into PAL_AUTOML_TRAIN_TAB values(226,76,0,0,35.7,0.198,26,0);
insert into PAL_AUTOML_TRAIN_TAB values(228,70,39,744,36.7,2.329,31,0);
insert into PAL_AUTOML_TRAIN_TAB values(229,80,31,53,45.2,0.08901,24,0);
insert into PAL_AUTOML_TRAIN_TAB values(230,86,0,0,44.0,0.645,22,1);
insert into PAL_AUTOML_TRAIN_TAB values(231,80,37,370,46.2,0.23802,46,1);
insert into PAL_AUTOML_TRAIN_TAB values(232,80,25,37,25.4,0.583,22,0);
insert into PAL_AUTOML_TRAIN_TAB values(234,68,28,45,29.7,0.293,23,0);
insert into PAL_AUTOML_TRAIN_TAB values(235,72,0,0,43.6,0.479,26,1);
insert into PAL_AUTOML_TRAIN_TAB values(239,76,0,0,18.4,0.5821,27,0);
insert into PAL_AUTOML_TRAIN_TAB values(241,70,32,88,33.1,0.446,22,0);
insert into PAL_AUTOML_TRAIN_TAB values(242,54,0,0,25.6,0.402,22,1);
insert into PAL_AUTOML_TRAIN_TAB values(243,50,22,176,27.1,1.318,33,1);
insert into PAL_AUTOML_TRAIN_TAB values(244,76,35,194,38.2,0.328996,29,0);
insert into PAL_AUTOML_TRAIN_TAB values(245,85,15,0,30.0,1.213,49,1);
insert into PAL_AUTOML_TRAIN_TAB values(246,68,0,0,31.2,0.258,41,0);
insert into PAL_AUTOML_TRAIN_TAB values(247,90,33,680,52.3,0.42705,23,0);
insert into PAL_AUTOML_TRAIN_TAB values(248,70,33,402,35.4,0.282,34,0);
insert into PAL_AUTOML_TRAIN_TAB values(249,86,19,0,30.1,0.14302,23,0);
insert into PAL_AUTOML_TRAIN_TAB values(250,52,0,0,31.2,0.38,42,0);
insert into PAL_AUTOML_TRAIN_TAB values(251,84,0,0,28.0,0.284,27,0);
insert into PAL_AUTOML_TRAIN_TAB values(252,80,14,55,24.4,0.249,24,0);
insert into PAL_AUTOML_TRAIN_TAB values(254,62,7,258,27.6,0.92599,44,1);
insert into PAL_AUTOML_TRAIN_TAB values(255,64,35,0,33.6,0.54299,21,1);
insert into PAL_AUTOML_TRAIN_TAB values(257,68,22,0,28.7,0.092,25,0);
insert into PAL_AUTOML_TRAIN_TAB values(258,50,16,375,25.9,0.655,24,0);
insert into PAL_AUTOML_TRAIN_TAB values(259,76,28,150,33.3,1.3532,51,1);
insert into PAL_AUTOML_TRAIN_TAB values(260,68,15,130,30.9,0.299,34,0);
insert into PAL_AUTOML_TRAIN_TAB values(261,0,0,0,30.0,0.76099,27,1);
```

```
insert into PAL_AUTOML_TRAIN_TAB values(262,70,32,0,32.1,0.612,24,0);
insert into PAL_AUTOML_TRAIN_TAB values(263,80,15,0,32.4,0.2,63,0);
insert into PAL_AUTOML_TRAIN_TAB values(264,62,0,0,32.0,0.226,35,1);
insert into PAL_AUTOML_TRAIN_TAB values(265,74,18,67,33.6,0.997,43,0);
insert into PAL_AUTOML_TRAIN_TAB values(266,0,0,0,36.3,0.93299,25,1);
insert into PAL_AUTOML_TRAIN_TAB values(268,52,0,0,25.1,0.078,21,0);
insert into PAL_AUTOML_TRAIN_TAB values(271,62,32,56,25.2,0.128,21,0);
insert into PAL_AUTOML_TRAIN_TAB values(272,78,0,0,23.0,0.254,40,0);
insert into PAL_AUTOML_TRAIN_TAB values(273,78,50,45,33.2,0.42204,21,0);
insert into PAL_AUTOML_TRAIN_TAB values(274,70,0,0,34.2,0.251,52,0);
insert into PAL_AUTOML_TRAIN_TAB values(275,70,52,57,40.5,0.677,25,0);
insert into PAL_AUTOML_TRAIN_TAB values(276,60,24,0,26.5,0.29604,29,1);
insert into PAL_AUTOML_TRAIN_TAB values(278,74,0,0,24.9,0.7441,57,0);
insert into PAL_AUTOML_TRAIN_TAB values(280,70,0,0,37.9,0.333996,28,1);
insert into PAL_AUTOML_TRAIN_TAB values(281,76,28,122,35.9,0.28,39,0);
insert into PAL_AUTOML_TRAIN_TAB values(283,86,0,0,30.4,0.165,47,1);
insert into PAL_AUTOML_TRAIN_TAB values(284,80,0,0,27.0,0.259,52,1);
insert into PAL_AUTOML_TRAIN_TAB values(285,74,26,135,26.0,0.647,51,0);
insert into PAL_AUTOML_TRAIN_TAB values(286,84,44,545,38.7,0.619,34,0);
insert into PAL_AUTOML_TRAIN_TAB values(287,86,39,220,45.6,0.80799,29,1);
insert into PAL_AUTOML_TRAIN_TAB values(288,56,17,49,20.8,0.34,26,0);
insert into PAL_AUTOML_TRAIN_TAB values(289,72,43,75,36.1,0.263,33,0);
insert into PAL_AUTOML_TRAIN_TAB values(290,88,29,40,36.9,0.434,21,0);
insert into PAL_AUTOML_TRAIN_TAB values(291,62,30,74,36.6,0.757,25,1);
insert into PAL_AUTOML_TRAIN_TAB values(292,78,37,182,43.3,1.224,31,1);
insert into PAL_AUTOML_TRAIN_TAB values(293,48,45,194,40.5,0.613,24,1);
insert into PAL_AUTOML_TRAIN_TAB values(295,62,31,120,35.5,0.6921,28,0);
insert into PAL_AUTOML_TRAIN_TAB values(296,70,38,360,28.0,0.337,29,1);
insert into PAL_AUTOML_TRAIN_TAB values(297,84,29,215,30.7,0.52,24,0);
insert into PAL_AUTOML_TRAIN_TAB values(298,78,25,184,36.6,0.41203,46,1);
insert into PAL_AUTOML_TRAIN_TAB values(300,0,0,0,32.3,0.8391,30,1);
insert into PAL_AUTOML_TRAIN_TAB values(302,82,41,42,35.8,0.156,35,0);
insert into PAL_AUTOML_TRAIN_TAB values(303,98,0,0,52.9,0.209,28,1);
insert into PAL_AUTOML_TRAIN_TAB values(304,76,0,0,21.0,0.207,37,0);
insert into PAL_AUTOML_TRAIN_TAB values(305,76,37,105,39.7,0.215,29,0);
insert into PAL_AUTOML_TRAIN_TAB values(306,68,23,132,25.5,0.326,47,1);
insert into PAL_AUTOML_TRAIN_TAB values(307,68,14,148,24.8,0.14302,21,0);
insert into PAL_AUTOML_TRAIN_TAB values(308,68,19,180,30.5,1.391,25,1);
insert into PAL_AUTOML_TRAIN_TAB values(311,70,37,148,39.4,0.605,22,0);
insert into PAL_AUTOML_TRAIN_TAB values(314,80,31,0,35.9,1.127,43,1);
insert into PAL_AUTOML_TRAIN_TAB values(315,68,22,94,34.1,0.315,26,0);
insert into PAL_AUTOML_TRAIN_TAB values(317,74,0,0,30.5,0.345,29,1);
insert into PAL_AUTOML_TRAIN_TAB values(318,66,39,140,38.1,0.15,28,0);
insert into PAL_AUTOML_TRAIN_TAB values(321,74,30,0,31.6,0.196998,25,1);
insert into PAL_AUTOML_TRAIN_TAB values(322,70,20,0,27.4,0.254,36,1);
insert into PAL_AUTOML_TRAIN_TAB values(324,75,32,0,35.7,0.14802,21,0);
insert into PAL_AUTOML_TRAIN_TAB values(325,72,21,168,25.6,0.12301,24,0);
insert into PAL_AUTOML_TRAIN_TAB values(326,64,32,156,35.1,0.6921,30,1);
insert into PAL_AUTOML_TRAIN_TAB values(328,86,36,120,45.5,0.127,23,1);
insert into PAL_AUTOML_TRAIN_TAB values(329,70,32,68,30.8,0.122,37,0);
insert into PAL_AUTOML_TRAIN_TAB values(330,72,19,0,23.1,1.476,46,0);
insert into PAL_AUTOML_TRAIN_TAB values(331,58,16,52,32.7,0.166,25,0);
insert into PAL_AUTOML_TRAIN_TAB values(332,0,0,0,43.3,0.282,41,1);
insert into PAL_AUTOML_TRAIN_TAB values(333,80,0,0,23.6,0.136998,44,0);
insert into PAL_AUTOML_TRAIN_TAB values(334,60,18,58,23.9,0.26,22,0);
insert into PAL_AUTOML_TRAIN_TAB values(335,76,43,255,47.9,0.259,26,0);
insert into PAL_AUTOML_TRAIN_TAB values(336,0,0,0,33.8,0.932,44,0);
insert into PAL_AUTOML_TRAIN_TAB values(338,78,34,171,34.2,0.893,33,1);
insert into PAL_AUTOML_TRAIN_TAB values(339,84,0,0,39.9,0.331,41,1);
insert into PAL_AUTOML_TRAIN_TAB values(340,70,13,105,25.9,0.47203,22,0);
insert into PAL_AUTOML_TRAIN_TAB values(341,74,21,73,25.9,0.67299,36,0);
insert into PAL_AUTOML_TRAIN_TAB values(342,68,35,0,32.0,0.389,22,0);
insert into PAL_AUTOML_TRAIN_TAB values(343,86,0,0,34.7,0.29,33,0);
insert into PAL_AUTOML_TRAIN_TAB values(344,72,0,0,36.8,0.485,57,0);
insert into PAL_AUTOML_TRAIN_TAB values(346,46,19,83,28.7,0.654,22,0);
insert into PAL_AUTOML_TRAIN_TAB values(347,0,0,0,23.5,0.187,23,0);
insert into PAL_AUTOML_TRAIN_TAB values(348,62,19,74,21.8,0.278997,26,0);
insert into PAL_AUTOML_TRAIN_TAB values(349,80,32,0,41.0,0.34603,37,1);
```

```
insert into PAL_AUTOML_TRAIN_TAB values(350,80,0,0,42.2,0.237,29,0);
insert into PAL_AUTOML_TRAIN_TAB values(351,84,0,0,31.2,0.252,30,0);
insert into PAL_AUTOML_TRAIN_TAB values(352,82,28,0,34.4,0.243,46,0);
insert into PAL_AUTOML_TRAIN_TAB values(353,62,12,43,27.2,0.58,24,0);
insert into PAL_AUTOML_TRAIN_TAB values(354,78,0,0,42.7,0.55899,21,0);
insert into PAL_AUTOML_TRAIN_TAB values(356,50,40,167,33.3,0.9621,28,1);
insert into PAL_AUTOML_TRAIN_TAB values(360,64,33,325,31.2,0.583,29,1);
insert into PAL_AUTOML_TRAIN_TAB values(361,70,0,0,29.8,0.207,63,0);
insert into PAL_AUTOML_TRAIN_TAB values(362,108,37,0,39.2,0.305,65,0);
insert into PAL_AUTOML_TRAIN_TAB values(363,78,0,0,38.5,0.52,67,1);
insert into PAL_AUTOML_TRAIN_TAB values(364,74,25,293,34.9,0.385,30,0);
insert into PAL_AUTOML_TRAIN_TAB values(366,72,0,0,27.6,0.368,29,1);
insert into PAL_AUTOML_TRAIN_TAB values(368,86,16,66,27.5,0.306,22,0);
insert into PAL_AUTOML_TRAIN_TAB values(370,82,48,465,38.4,2.137,25,1);
insert into PAL_AUTOML_TRAIN_TAB values(371,64,23,89,0.0,1.73099,21,0);
insert into PAL_AUTOML_TRAIN_TAB values(373,58,40,94,34.9,0.225,25,0);
insert into PAL_AUTOML_TRAIN_TAB values(374,52,43,158,36.2,0.816,28,0);
insert into PAL_AUTOML_TRAIN_TAB values(375,82,43,325,39.2,0.528,58,1);
insert into PAL_AUTOML_TRAIN_TAB values(376,82,15,84,25.2,0.299,22,0);
insert into PAL_AUTOML_TRAIN_TAB values(377,60,37,75,37.2,0.509,22,0);
insert into PAL_AUTOML_TRAIN_TAB values(378,75,0,0,48.3,0.23802,32,1);
insert into PAL_AUTOML_TRAIN_TAB values(379,100,39,72,43.4,1.021,35,0);
insert into PAL_AUTOML_TRAIN_TAB values(380,72,30,82,30.8,0.821,24,0);
insert into PAL_AUTOML_TRAIN_TAB values(381,68,22,0,20.0,0.23602,22,0);
insert into PAL_AUTOML_TRAIN_TAB values(382,60,8,182,25.4,0.9471,21,0);
insert into PAL_AUTOML_TRAIN_TAB values(383,62,18,59,25.1,1.268,25,0);
insert into PAL_AUTOML_TRAIN_TAB values(384,70,24,110,24.3,0.221,25,0);
insert into PAL_AUTOML_TRAIN_TAB values(385,54,13,50,22.3,0.205,24,0);
insert into PAL_AUTOML_TRAIN_TAB values(386,74,29,0,32.3,0.66,35,1);
insert into PAL_AUTOML_TRAIN_TAB values(387,100,36,0,43.3,0.239,45,1);
insert into PAL_AUTOML_TRAIN_TAB values(389,68,23,81,31.6,0.9491,28,0);
insert into PAL_AUTOML_TRAIN_TAB values(391,76,0,0,45.7,0.34,27,1);
insert into PAL_AUTOML_TRAIN_TAB values(392,64,14,415,23.7,0.389,21,0);
insert into PAL_AUTOML_TRAIN_TAB values(393,72,12,87,22.1,0.462997,37,0);
insert into PAL_AUTOML_TRAIN_TAB values(394,78,0,0,32.9,0.80299,31,1);
insert into PAL_AUTOML_TRAIN_TAB values(395,58,24,275,27.7,1.6,25,0);
insert into PAL_AUTOML_TRAIN_TAB values(398,70,0,0,21.1,0.389,25,0);
insert into PAL_AUTOML_TRAIN_TAB values(400,64,0,0,32.0,0.161,31,1);
insert into PAL_AUTOML_TRAIN_TAB values(401,61,0,0,24.2,0.151,55,0);
insert into PAL_AUTOML_TRAIN_TAB values(403,78,25,0,31.6,0.28,38,0);
insert into PAL_AUTOML_TRAIN_TAB values(404,64,0,0,32.9,0.135,41,1);
insert into PAL_AUTOML_TRAIN_TAB values(405,48,32,165,42.1,0.52,26,0);
insert into PAL_AUTOML_TRAIN_TAB values(406,72,0,0,28.9,0.376,46,1);
insert into PAL_AUTOML_TRAIN_TAB values(407,62,0,0,21.9,0.336,25,0);
insert into PAL_AUTOML_TRAIN_TAB values(408,74,0,0,25.9,1.19098,39,1);
insert into PAL_AUTOML_TRAIN_TAB values(409,68,49,579,42.4,0.7021,28,1);
insert into PAL_AUTOML_TRAIN_TAB values(410,90,39,0,35.7,0.674,28,0);
insert into PAL_AUTOML_TRAIN_TAB values(411,72,30,176,34.4,0.528,25,0);
insert into PAL_AUTOML_TRAIN_TAB values(415,84,33,474,35.7,0.258,22,1);
insert into PAL_AUTOML_TRAIN_TAB values(416,68,21,0,27.2,1.095,22,0);
insert into PAL_AUTOML_TRAIN_TAB values(417,82,32,0,38.5,0.55399,37,1);
insert into PAL_AUTOML_TRAIN_TAB values(418,68,0,0,18.2,0.624,27,0);
insert into PAL_AUTOML_TRAIN_TAB values(419,64,29,115,26.4,0.218997,28,1);
insert into PAL_AUTOML_TRAIN_TAB values(420,88,41,170,45.3,0.507,26,0);
insert into PAL_AUTOML_TRAIN_TAB values(421,68,18,76,26.0,0.561,21,0);
insert into PAL_AUTOML_TRAIN_TAB values(422,64,46,78,40.6,0.496,21,0);
insert into PAL_AUTOML_TRAIN_TAB values(423,64,22,0,30.8,0.42104,21,0);
insert into PAL_AUTOML_TRAIN_TAB values(424,78,32,210,42.9,0.516,36,1);
insert into PAL_AUTOML_TRAIN_TAB values(425,78,39,277,37.0,0.264,31,1);
insert into PAL_AUTOML_TRAIN_TAB values(426,0,0,0,0.0,0.256,25,0);
insert into PAL_AUTOML_TRAIN_TAB values(428,94,46,145,40.6,0.284,26,0);
insert into PAL_AUTOML_TRAIN_TAB values(429,82,25,180,35.0,0.233,43,1);
insert into PAL_AUTOML_TRAIN_TAB values(430,0,0,0,22.2,0.10801,23,0);
insert into PAL_AUTOML_TRAIN_TAB values(432,74,11,60,30.0,0.527,22,0);
insert into PAL_AUTOML_TRAIN_TAB values(433,75,0,0,25.6,0.166998,29,0);
insert into PAL_AUTOML_TRAIN_TAB values(434,68,8,0,24.5,1.138,36,0);
insert into PAL_AUTOML_TRAIN_TAB values(435,0,0,0,42.4,0.205,29,1);
insert into PAL_AUTOML_TRAIN_TAB values(436,85,33,0,37.4,0.244,41,0);
```

```
insert into PAL_AUTOML_TRAIN_TAB values(437,75,0,0,29.9,0.434,28,0);
insert into PAL_AUTOML_TRAIN_TAB values(438,70,15,0,18.2,0.147,21,0);
insert into PAL_AUTOML_TRAIN_TAB values(440,104,25,0,34.3,0.435,41,1);
insert into PAL_AUTOML_TRAIN_TAB values(441,66,23,50,32.2,0.49705,22,0);
insert into PAL_AUTOML_TRAIN_TAB values(442,64,27,120,33.2,0.23,24,0);
insert into PAL_AUTOML_TRAIN_TAB values(443,70,0,0,30.5,0.955,33,1);
insert into PAL_AUTOML_TRAIN_TAB values(444,62,12,0,29.7,0.38,30,1);
insert into PAL_AUTOML_TRAIN_TAB values(445,78,63,14,59.4,2.42,25,1);
insert into PAL_AUTOML_TRAIN_TAB values(446,72,12,70,25.3,0.65799,28,0);
insert into PAL_AUTOML_TRAIN_TAB values(447,80,45,92,36.5,0.33,26,0);
insert into PAL_AUTOML_TRAIN_TAB values(448,64,37,64,33.6,0.51,22,1);
insert into PAL_AUTOML_TRAIN_TAB values(449,74,18,63,30.5,0.285,26,0);
insert into PAL_AUTOML_TRAIN_TAB values(451,70,0,0,28.9,0.542,23,1);
insert into PAL_AUTOML_TRAIN_TAB values(453,0,0,0,19.6,0.8321,72,0);
insert into PAL_AUTOML_TRAIN_TAB values(454,54,28,105,37.8,0.498,24,0);
insert into PAL_AUTOML_TRAIN_TAB values(456,54,0,0,26.7,0.687,62,0);
insert into PAL_AUTOML_TRAIN_TAB values(458,84,48,237,37.6,1.001,51,1);
insert into PAL_AUTOML_TRAIN_TAB values(459,74,33,60,25.9,0.46,81,0);
insert into PAL_AUTOML_TRAIN_TAB values(460,72,22,56,20.8,0.733,48,0);
insert into PAL_AUTOML_TRAIN_TAB values(462,70,40,49,35.3,0.705,39,0);
insert into PAL_AUTOML_TRAIN_TAB values(463,78,30,0,27.6,0.258,37,0);
insert into PAL_AUTOML_TRAIN_TAB values(464,98,0,0,24.0,1.022,34,0);
insert into PAL_AUTOML_TRAIN_TAB values(466,52,10,36,27.8,0.268996,22,0);
insert into PAL_AUTOML_TRAIN_TAB values(467,64,36,100,36.8,0.6,25,0);
insert into PAL_AUTOML_TRAIN_TAB values(468,0,0,0,30.0,0.183,38,1);
insert into PAL_AUTOML_TRAIN_TAB values(469,78,41,140,46.1,0.5711,27,0);
insert into PAL_AUTOML_TRAIN_TAB values(470,82,40,0,41.3,0.607,28,0);
insert into PAL_AUTOML_TRAIN_TAB values(471,70,38,0,33.2,0.17,22,0);
insert into PAL_AUTOML_TRAIN_TAB values(472,66,27,0,38.8,0.259,22,0);
insert into PAL_AUTOML_TRAIN_TAB values(473,90,0,0,29.9,0.21,50,0);
insert into PAL_AUTOML_TRAIN_TAB values(474,64,0,0,28.9,0.126,24,0);
insert into PAL_AUTOML_TRAIN_TAB values(475,84,27,0,27.3,0.231,59,0);
insert into PAL_AUTOML_TRAIN_TAB values(476,80,45,191,33.7,0.711,29,1);
insert into PAL_AUTOML_TRAIN_TAB values(477,76,17,110,23.8,0.466,31,0);
insert into PAL_AUTOML_TRAIN_TAB values(478,74,38,75,25.9,0.162,39,0);
insert into PAL_AUTOML_TRAIN_TAB values(479,86,31,0,28.0,0.419,63,0);
insert into PAL_AUTOML_TRAIN_TAB values(480,70,30,328,35.5,0.344,35,1);
insert into PAL_AUTOML_TRAIN_TAB values(481,88,37,0,35.2,0.196998,29,0);
insert into PAL_AUTOML_TRAIN_TAB values(482,58,22,49,27.8,0.306,28,0);
insert into PAL_AUTOML_TRAIN_TAB values(483,82,31,125,38.2,0.233,23,0);
insert into PAL_AUTOML_TRAIN_TAB values(484,0,0,0,44.2,0.63,31,1);
insert into PAL_AUTOML_TRAIN_TAB values(485,68,42,250,42.3,0.365,24,1);
insert into PAL_AUTOML_TRAIN_TAB values(487,78,32,265,46.5,1.159,58,0);
insert into PAL_AUTOML_TRAIN_TAB values(488,72,17,0,25.6,0.294,28,0);
insert into PAL_AUTOML_TRAIN_TAB values(490,65,28,66,36.8,0.629,24,0);
insert into PAL_AUTOML_TRAIN_TAB values(493,70,18,122,28.9,1.1441,45,1);
insert into PAL_AUTOML_TRAIN_TAB values(494,0,0,0,0.0,0.174,22,0);
insert into PAL_AUTOML_TRAIN_TAB values(495,74,0,0,26.6,0.304,66,0);
insert into PAL_AUTOML_TRAIN_TAB values(496,68,0,0,26.0,0.292,30,0);
insert into PAL_AUTOML_TRAIN_TAB values(497,72,15,76,30.1,0.547,25,0);
insert into PAL_AUTOML_TRAIN_TAB values(498,70,33,145,25.1,0.163,55,1);
insert into PAL_AUTOML_TRAIN_TAB values(499,74,32,193,29.3,0.8391,39,0);
insert into PAL_AUTOML_TRAIN_TAB values(500,90,19,71,25.2,0.313,21,0);
insert into PAL_AUTOML_TRAIN_TAB values(501,72,32,0,37.2,0.267,28,0);
insert into PAL_AUTOML_TRAIN_TAB values(504,78,39,0,37.3,0.23802,40,0);
insert into PAL_AUTOML_TRAIN_TAB values(507,60,23,170,28.6,0.6921,21,0);
insert into PAL_AUTOML_TRAIN_TAB values(508,50,23,76,30.4,0.968,21,0);
insert into PAL_AUTOML_TRAIN_TAB values(509,78,0,0,25.0,0.409,64,0);
insert into PAL_AUTOML_TRAIN_TAB values(512,68,0,0,24.2,0.2,58,0);
insert into PAL_AUTOML_TRAIN_TAB values(514,54,19,86,25.6,0.154,24,0);
insert into PAL_AUTOML_TRAIN_TAB values(515,70,18,105,31.6,0.268,28,1);
insert into PAL_AUTOML_TRAIN_TAB values(516,88,34,165,30.3,0.77099,53,1);
insert into PAL_AUTOML_TRAIN_TAB values(517,86,0,0,37.6,0.304,51,0);
insert into PAL_AUTOML_TRAIN_TAB values(518,60,0,0,32.8,0.18,41,0);
insert into PAL_AUTOML_TRAIN_TAB values(519,90,7,326,19.6,0.5821,60,0);
insert into PAL_AUTOML_TRAIN_TAB values(520,70,32,66,25.0,0.187,25,0);
insert into PAL_AUTOML_TRAIN_TAB values(521,80,33,130,33.2,0.305,26,0);
insert into PAL_AUTOML_TRAIN_TAB values(522,0,0,0,0.0,0.188997,26,0);
```

```
insert into PAL_AUTOML_TRAIN_TAB values(523,70,0,0,34.2,0.652,45,1);
insert into PAL_AUTOML_TRAIN_TAB values(524,58,0,0,31.6,0.151,24,0);
insert into PAL_AUTOML_TRAIN_TAB values(526,64,19,82,18.2,0.299,21,0);
insert into PAL_AUTOML_TRAIN_TAB values(527,74,15,105,26.3,0.107,24,0);
insert into PAL_AUTOML_TRAIN_TAB values(528,66,31,188,30.8,0.493,22,0);
insert into PAL_AUTOML_TRAIN_TAB values(530,60,18,106,29.8,0.7171,22,0);
insert into PAL_AUTOML_TRAIN_TAB values(531,76,0,0,45.3,0.68599,24,0);
insert into PAL_AUTOML_TRAIN_TAB values(532,66,52,65,41.3,0.917,29,0);
insert into PAL_AUTOML_TRAIN_TAB values(533,0,0,0,29.8,0.501,31,0);
insert into PAL_AUTOML_TRAIN_TAB values(534,56,30,56,33.3,1.251,24,0);
insert into PAL_AUTOML_TRAIN_TAB values(535,0,0,0,32.9,0.302,23,1);
insert into PAL_AUTOML_TRAIN_TAB values(536,90,0,0,29.6,0.196998,46,0);
insert into PAL_AUTOML_TRAIN_TAB values(538,80,37,210,36.3,0.804,23,0);
insert into PAL_AUTOML_TRAIN_TAB values(539,92,49,155,36.4,0.968,32,1);
insert into PAL_AUTOML_TRAIN_TAB values(540,74,40,215,39.4,0.66099,43,1);
insert into PAL_AUTOML_TRAIN_TAB values(541,72,25,190,32.4,0.54899,27,1);
insert into PAL_AUTOML_TRAIN_TAB values(542,85,32,0,34.9,0.825,56,1);
insert into PAL_AUTOML_TRAIN_TAB values(544,78,29,76,32.0,0.365,29,0);
insert into PAL_AUTOML_TRAIN_TAB values(545,90,35,225,34.5,0.423,37,1);
insert into PAL_AUTOML_TRAIN_TAB values(546,76,27,207,43.6,1.034,53,1);
insert into PAL_AUTOML_TRAIN_TAB values(547,68,21,166,33.1,0.16,28,0);
insert into PAL_AUTOML_TRAIN_TAB values(548,82,43,67,32.8,0.341,50,0);
insert into PAL_AUTOML_TRAIN_TAB values(549,110,31,0,28.5,0.68,37,0);
insert into PAL_AUTOML_TRAIN_TAB values(551,68,30,106,31.9,0.591,25,0);
insert into PAL_AUTOML_TRAIN_TAB values(552,88,0,0,27.8,0.247,66,0);
insert into PAL_AUTOML_TRAIN_TAB values(553,62,24,44,29.9,0.42204,23,0);
insert into PAL_AUTOML_TRAIN_TAB values(554,64,23,115,36.9,0.47103,28,0);
insert into PAL_AUTOML_TRAIN_TAB values(555,70,33,215,25.5,0.161,37,0);
insert into PAL_AUTOML_TRAIN_TAB values(556,70,40,0,38.1,0.218,30,0);
insert into PAL_AUTOML_TRAIN_TAB values(559,74,0,0,30.1,0.3,35,0);
insert into PAL_AUTOML_TRAIN_TAB values(560,76,0,0,33.8,0.121,54,1);
insert into PAL_AUTOML_TRAIN_TAB values(561,66,32,274,41.3,0.502,28,1);
insert into PAL_AUTOML_TRAIN_TAB values(562,68,34,77,37.6,0.401,24,0);
insert into PAL_AUTOML_TRAIN_TAB values(563,60,19,54,26.9,0.49705,32,0);
insert into PAL_AUTOML_TRAIN_TAB values(564,80,0,0,32.4,0.601,27,0);
insert into PAL_AUTOML_TRAIN_TAB values(565,54,14,88,26.1,0.748,22,0);
insert into PAL_AUTOML_TRAIN_TAB values(566,72,30,18,38.6,0.41203,21,0);
insert into PAL_AUTOML_TRAIN_TAB values(567,62,32,126,32.0,0.085,46,0);
insert into PAL_AUTOML_TRAIN_TAB values(568,72,29,126,31.3,0.337997,37,0);
insert into PAL_AUTOML_TRAIN_TAB values(569,66,30,165,34.3,0.203,33,1);
insert into PAL_AUTOML_TRAIN_TAB values(570,70,0,0,32.5,0.27,39,0);
insert into PAL_AUTOML_TRAIN_TAB values(571,96,0,0,22.6,0.268,21,0);
insert into PAL_AUTOML_TRAIN_TAB values(572,58,31,44,29.5,0.43,22,0);
insert into PAL_AUTOML_TRAIN_TAB values(573,60,17,120,34.7,0.198,22,0);
insert into PAL_AUTOML_TRAIN_TAB values(574,86,30,330,30.1,0.892,23,0);
insert into PAL_AUTOML_TRAIN_TAB values(576,44,20,130,24.0,0.813,35,0);
insert into PAL_AUTOML_TRAIN_TAB values(577,80,0,0,42.9,0.693,21,1);
insert into PAL_AUTOML_TRAIN_TAB values(578,68,0,0,27.0,0.245,36,0);
insert into PAL_AUTOML_TRAIN_TAB values(579,70,99,0,34.7,0.575,62,1);
insert into PAL_AUTOML_TRAIN_TAB values(580,90,46,0,42.1,0.371,21,1);
insert into PAL_AUTOML_TRAIN_TAB values(582,78,17,0,26.5,0.259,62,0);
insert into PAL_AUTOML_TRAIN_TAB values(583,76,0,0,38.7,0.19,42,0);
insert into PAL_AUTOML_TRAIN_TAB values(584,76,24,600,28.7,0.687,52,1);
insert into PAL_AUTOML_TRAIN_TAB values(585,56,11,0,22.5,0.41704,22,0);
insert into PAL_AUTOML_TRAIN_TAB values(586,66,0,0,34.9,0.129,41,1);
insert into PAL_AUTOML_TRAIN_TAB values(587,66,0,0,24.3,0.249,29,0);
insert into PAL_AUTOML_TRAIN_TAB values(589,0,0,0,21.1,0.342,25,0);
insert into PAL_AUTOML_TRAIN_TAB values(590,84,40,0,46.8,0.925,45,1);
insert into PAL_AUTOML_TRAIN_TAB values(592,80,0,0,34.4,0.402,44,1);
insert into PAL_AUTOML_TRAIN_TAB values(593,52,22,115,28.5,1.699,25,0);
insert into PAL_AUTOML_TRAIN_TAB values(594,72,45,230,33.6,0.733,34,0);
insert into PAL_AUTOML_TRAIN_TAB values(595,82,14,185,32.0,0.682,22,1);
insert into PAL_AUTOML_TRAIN_TAB values(596,76,0,0,45.3,0.193998,46,0);
insert into PAL_AUTOML_TRAIN_TAB values(597,24,19,25,27.8,0.55899,21,0);
insert into PAL_AUTOML_TRAIN_TAB values(598,74,0,0,36.8,0.08801,38,1);
insert into PAL_AUTOML_TRAIN_TAB values(599,38,18,120,23.1,0.40703,26,0);
insert into PAL_AUTOML_TRAIN_TAB values(600,88,19,0,27.1,0.4,24,0);
insert into PAL_AUTOML_TRAIN_TAB values(601,0,0,0,23.7,0.19,28,0);
```

```
insert into PAL_AUTOML_TRAIN_TAB values(602,74,36,0,27.8,0.1,30,0);
insert into PAL_AUTOML_TRAIN_TAB values(603,78,29,126,35.2,0.6921,54,1);
insert into PAL_AUTOML_TRAIN_TAB values(605,60,32,0,35.8,0.514,21,0);
insert into PAL_AUTOML_TRAIN_TAB values(606,78,42,293,40.0,1.258,22,1);
insert into PAL_AUTOML_TRAIN_TAB values(607,62,25,41,19.5,0.48204,25,0);
insert into PAL_AUTOML_TRAIN_TAB values(608,82,39,272,41.5,0.27,27,0);
insert into PAL_AUTOML_TRAIN_TAB values(610,54,21,158,30.9,0.292,24,0);
insert into PAL_AUTOML_TRAIN_TAB values(612,88,42,321,38.2,0.787,40,1);
insert into PAL_AUTOML_TRAIN_TAB values(613,80,28,0,32.5,0.878,26,0);
insert into PAL_AUTOML_TRAIN_TAB values(614,74,26,144,36.1,0.557,50,1);
insert into PAL_AUTOML_TRAIN_TAB values(615,72,0,0,25.8,0.207,27,0);
insert into PAL_AUTOML_TRAIN_TAB values(616,96,0,0,28.7,0.157,30,0);
insert into PAL_AUTOML_TRAIN_TAB values(617,62,13,15,20.1,0.257,23,0);
insert into PAL_AUTOML_TRAIN_TAB values(619,0,0,0,32.4,0.141,24,1);
insert into PAL_AUTOML_TRAIN_TAB values(620,86,42,160,38.4,0.24602,28,0);
insert into PAL_AUTOML_TRAIN_TAB values(621,76,20,0,24.2,1.6982,28,0);
insert into PAL_AUTOML_TRAIN_TAB values(622,94,0,0,40.8,1.46099,45,0);
insert into PAL_AUTOML_TRAIN_TAB values(623,70,27,115,43.5,0.34703,21,0);
insert into PAL_AUTOML_TRAIN_TAB values(624,64,0,0,30.8,0.158,21,0);
insert into PAL_AUTOML_TRAIN_TAB values(625,88,47,54,37.7,0.36204,29,0);
insert into PAL_AUTOML_TRAIN_TAB values(626,68,0,0,24.7,0.20602,21,0);
insert into PAL_AUTOML_TRAIN_TAB values(627,78,0,0,32.4,0.392996,21,0);
insert into PAL_AUTOML_TRAIN_TAB values(630,64,0,0,27.4,0.732,34,1);
insert into PAL_AUTOML_TRAIN_TAB values(633,82,17,183,27.5,0.115,22,0);
insert into PAL_AUTOML_TRAIN_TAB values(635,72,0,0,31.2,0.465,38,1);
insert into PAL_AUTOML_TRAIN_TAB values(637,76,18,66,31.6,0.649,23,0);
insert into PAL_AUTOML_TRAIN_TAB values(638,76,32,91,40.9,0.871,32,1);
insert into PAL_AUTOML_TRAIN_TAB values(639,74,12,46,19.5,0.149,28,0);
insert into PAL_AUTOML_TRAIN_TAB values(640,86,17,105,29.3,0.695,27,0);
insert into PAL_AUTOML_TRAIN_TAB values(641,70,0,0,34.3,0.303,24,0);
insert into PAL_AUTOML_TRAIN_TAB values(642,80,0,0,29.5,0.17802,50,1);
insert into PAL_AUTOML_TRAIN_TAB values(643,0,0,0,28.0,0.61,31,0);
insert into PAL_AUTOML_TRAIN_TAB values(644,72,30,152,27.6,0.73,27,0);
insert into PAL_AUTOML_TRAIN_TAB values(645,74,35,440,39.4,0.134,30,0);
insert into PAL_AUTOML_TRAIN_TAB values(646,74,17,144,23.4,0.447,33,1);
insert into PAL_AUTOML_TRAIN_TAB values(647,50,36,159,37.8,0.455,22,1);
insert into PAL_AUTOML_TRAIN_TAB values(648,84,35,130,28.3,0.26,42,1);
insert into PAL_AUTOML_TRAIN_TAB values(649,60,25,0,26.4,0.133,23,0);
insert into PAL_AUTOML_TRAIN_TAB values(650,54,25,100,25.2,0.233999,23,0);
insert into PAL_AUTOML_TRAIN_TAB values(651,60,23,106,33.8,0.466,27,0);
insert into PAL_AUTOML_TRAIN_TAB values(653,54,0,0,26.8,0.455,27,0);
insert into PAL_AUTOML_TRAIN_TAB values(654,70,28,135,34.2,0.142,22,0);
insert into PAL_AUTOML_TRAIN_TAB values(657,80,48,200,38.9,1.162,41,0);
insert into PAL_AUTOML_TRAIN_TAB values(658,106,0,0,39.0,0.19,51,0);
insert into PAL_AUTOML_TRAIN_TAB values(659,82,31,70,34.2,1.29198,27,1);
insert into PAL_AUTOML_TRAIN_TAB values(661,76,43,0,42.9,1.3941,22,1);
insert into PAL_AUTOML_TRAIN_TAB values(662,106,46,231,37.6,0.165,43,1);
insert into PAL_AUTOML_TRAIN_TAB values(663,80,46,130,37.9,0.637,40,1);
insert into PAL_AUTOML_TRAIN_TAB values(664,60,39,0,33.7,0.245,40,1);
insert into PAL_AUTOML_TRAIN_TAB values(665,80,45,132,34.8,0.217,24,0);
insert into PAL_AUTOML_TRAIN_TAB values(666,82,18,0,32.5,0.235,70,1);
insert into PAL_AUTOML_TRAIN_TAB values(667,70,27,0,27.5,0.141,40,1);
insert into PAL_AUTOML_TRAIN_TAB values(668,58,33,190,34.0,0.43,43,0);
insert into PAL_AUTOML_TRAIN_TAB values(669,78,30,100,30.9,0.163998,45,0);
insert into PAL_AUTOML_TRAIN_TAB values(670,68,26,168,33.6,0.631,49,0);
insert into PAL_AUTOML_TRAIN_TAB values(671,58,10,0,25.4,0.551,21,0);
insert into PAL_AUTOML_TRAIN_TAB values(672,106,23,49,35.5,0.285,47,0);
insert into PAL_AUTOML_TRAIN_TAB values(673,100,35,240,57.3,0.88,22,0);
insert into PAL_AUTOML_TRAIN_TAB values(674,82,0,0,35.6,0.5871,68,0);
insert into PAL_AUTOML_TRAIN_TAB values(675,70,0,0,30.9,0.327996,31,1);
insert into PAL_AUTOML_TRAIN_TAB values(676,86,0,0,24.8,0.23,53,1);
insert into PAL_AUTOML_TRAIN_TAB values(681,76,36,0,49.6,0.364,26,1);
insert into PAL_AUTOML_TRAIN_TAB values(683,80,0,0,32.3,0.536,27,1);
insert into PAL_AUTOML_TRAIN_TAB values(684,82,0,0,0.0,0.64,69,0);
insert into PAL_AUTOML_TRAIN_TAB values(685,74,26,205,33.2,0.591,25,0);
insert into PAL_AUTOML_TRAIN_TAB values(686,64,0,0,23.1,0.314,22,0);
insert into PAL_AUTOML_TRAIN_TAB values(687,50,19,0,28.3,0.18102,29,0);
insert into PAL_AUTOML_TRAIN_TAB values(689,82,46,180,46.1,0.335,46,1);
```

```
insert into PAL_AUTOML_TRAIN_TAB values(690,80,0,0,24.6,0.856,34,0);
insert into PAL_AUTOML_TRAIN_TAB values(691,114,0,0,42.3,0.257,44,1);
insert into PAL_AUTOML_TRAIN_TAB values(692,70,32,95,39.1,0.88599,23,0);
insert into PAL_AUTOML_TRAIN_TAB values(693,68,49,125,38.5,0.439,43,1);
insert into PAL_AUTOML_TRAIN_TAB values(694,60,0,0,23.5,0.191,25,0);
insert into PAL_AUTOML_TRAIN_TAB values(695,90,24,480,30.4,0.128,43,1);
insert into PAL_AUTOML_TRAIN_TAB values(696,74,19,125,29.9,0.268,31,1);
insert into PAL_AUTOML_TRAIN_TAB values(698,88,11,155,34.5,0.598,28,0);
insert into PAL_AUTOML_TRAIN_TAB values(699,70,0,0,44.5,0.904,26,0);
insert into PAL_AUTOML_TRAIN_TAB values(700,76,27,200,35.9,0.483,26,0);
insert into PAL_AUTOML_TRAIN_TAB values(701,78,31,0,27.6,0.565,49,1);
insert into PAL_AUTOML_TRAIN_TAB values(703,0,0,0,38.5,0.304,41,0);
insert into PAL_AUTOML_TRAIN_TAB values(704,76,20,100,28.4,0.11801,27,0);
insert into PAL_AUTOML_TRAIN_TAB values(705,80,36,0,39.8,0.177,28,0);
insert into PAL_AUTOML_TRAIN_TAB values(707,46,21,335,34.4,0.17602,22,0);
insert into PAL_AUTOML_TRAIN_TAB values(708,78,0,0,32.8,0.14802,45,1);
insert into PAL_AUTOML_TRAIN_TAB values(709,64,32,160,38.0,0.674,23,1);
insert into PAL_AUTOML_TRAIN_TAB values(710,64,13,387,31.2,0.295,24,0);
insert into PAL_AUTOML_TRAIN_TAB values(711,78,27,22,29.6,0.439,40,0);
insert into PAL_AUTOML_TRAIN_TAB values(712,62,36,0,41.2,0.441,38,1);
insert into PAL_AUTOML_TRAIN_TAB values(713,58,20,291,26.4,0.35204,21,0);
insert into PAL_AUTOML_TRAIN_TAB values(714,74,0,0,29.5,0.121,32,0);
insert into PAL_AUTOML_TRAIN_TAB values(715,50,33,392,33.9,0.826,34,1);
insert into PAL_AUTOML_TRAIN_TAB values(717,72,18,0,23.1,0.595,56,0);
insert into PAL_AUTOML_TRAIN_TAB values(719,76,27,0,35.6,0.377995,52,1);
insert into PAL_AUTOML_TRAIN_TAB values(720,86,19,0,29.3,0.317,34,0);
insert into PAL_AUTOML_TRAIN_TAB values(722,68,29,127,29.3,0.349,42,1);
insert into PAL_AUTOML_TRAIN_TAB values(723,86,30,105,39.1,0.251,42,0);
insert into PAL_AUTOML_TRAIN_TAB values(724,94,0,0,32.8,0.265,45,0);
insert into PAL_AUTOML_TRAIN_TAB values(726,78,29,180,36.1,0.496,25,0);
insert into PAL_AUTOML_TRAIN_TAB values(727,84,26,0,32.4,0.433,22,0);
insert into PAL_AUTOML_TRAIN_TAB values(728,88,0,0,22.9,0.326,22,0);
insert into PAL_AUTOML_TRAIN_TAB values(729,52,0,0,30.1,0.141,22,0);
insert into PAL_AUTOML_TRAIN_TAB values(730,78,23,79,28.4,0.322995,34,1);
insert into PAL_AUTOML_TRAIN_TAB values(731,86,0,0,28.4,0.259,22,1);
insert into PAL_AUTOML_TRAIN_TAB values(732,88,37,120,44.5,0.64599,24,1);
insert into PAL_AUTOML_TRAIN_TAB values(733,56,27,165,29.0,0.426,22,0);
insert into PAL_AUTOML_TRAIN_TAB values(734,75,0,0,23.3,0.56,53,0);
insert into PAL_AUTOML_TRAIN_TAB values(735,60,32,0,35.4,0.284,28,0);
insert into PAL_AUTOML_TRAIN_TAB values(736,86,27,120,27.4,0.515,21,0);
insert into PAL_AUTOML_TRAIN_TAB values(737,72,23,0,32.0,0.6,42,0);
insert into PAL_AUTOML_TRAIN_TAB values(738,60,17,160,36.6,0.452996,21,0);
insert into PAL_AUTOML_TRAIN_TAB values(739,74,0,0,39.5,0.293,42,1);
insert into PAL_AUTOML_TRAIN_TAB values(740,80,37,150,42.3,0.785,48,1);
insert into PAL_AUTOML_TRAIN_TAB values(742,58,18,116,28.5,0.218997,22,0);
insert into PAL_AUTOML_TRAIN_TAB values(743,94,0,0,32.7,0.7341,45,1);
insert into PAL_AUTOML_TRAIN_TAB values(744,88,37,140,40.6,1.1742,39,0);
insert into PAL_AUTOML_TRAIN_TAB values(745,84,33,105,30.0,0.488,46,0);
insert into PAL_AUTOML_TRAIN_TAB values(746,94,41,0,49.3,0.358,27,1);
insert into PAL_AUTOML_TRAIN_TAB values(748,70,22,200,36.4,0.408,36,1);
insert into PAL_AUTOML_TRAIN_TAB values(750,70,0,0,31.2,1.182,22,1);
insert into PAL_AUTOML_TRAIN_TAB values(751,78,39,74,39.0,0.261,28,0);
insert into PAL_AUTOML_TRAIN_TAB values(752,62,24,0,26.0,0.223,25,0);
insert into PAL_AUTOML_TRAIN_TAB values(753,88,44,510,43.3,0.222,26,1);
insert into PAL_AUTOML_TRAIN_TAB values(754,78,32,0,32.4,0.442995,45,1);
insert into PAL_AUTOML_TRAIN_TAB values(756,90,41,0,32.0,0.391,39,0);
insert into PAL_AUTOML_TRAIN_TAB values(757,72,0,0,36.3,0.258,52,1);
insert into PAL_AUTOML_TRAIN_TAB values(758,76,0,0,37.5,0.196998,26,0);
insert into PAL_AUTOML_TRAIN_TAB values(760,58,26,16,28.4,0.76599,22,0);
insert into PAL_AUTOML_TRAIN_TAB values(762,62,0,0,22.5,0.142,33,0);
insert into PAL_AUTOML_TRAIN_TAB values(763,76,48,180,32.9,0.171,63,0);
insert into PAL_AUTOML_TRAIN_TAB values(764,70,27,0,36.8,0.34,27,0);
insert into PAL_AUTOML_TRAIN_TAB values(765,72,23,112,26.2,0.245,30,0);
insert into PAL_AUTOML_TRAIN_TAB values(766,60,0,0,30.1,0.349,47,1);
```

## Test Data

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_AUTOML_TEST_TAB;
CREATE COLUMN TABLE PAL_AUTOML_TEST_TAB(
"ID" INTEGER,
"X0" INTEGER,
"X1" INTEGER,
"X2" INTEGER,
"X3" DOUBLE,
"X4" DOUBLE,
"X5" INTEGER
);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(7,0,0,0,35.3,0.134,29);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(16,84,47,230,45.8,0.551,31);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(17,74,0,0,29.6,0.254,31);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(27,66,15,140,23.2,0.48704,22);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(32,58,11,54,24.8,0.267,22);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(35,60,33,192,24.0,0.966,33);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(43,110,24,240,45.4,0.721,54);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(54,66,42,342,34.7,0.718,42);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(56,68,39,304,37.7,0.254,41);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(65,74,27,0,29.0,0.203,32);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(73,86,20,270,35.1,0.231,23);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(76,78,0,0,32.6,0.391,41);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(79,66,22,0,25.0,0.307,24);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(85,74,29,125,32.4,0.698,27);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(86,72,54,0,36.6,0.17802,45);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(94,82,18,64,24.7,0.76099,21);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(96,62,28,0,31.6,0.13,24);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(99,90,51,220,49.7,0.325,31);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(101,60,0,0,26.1,0.179,22);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(103,72,18,40,26.6,0.28303,24);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(106,122,0,0,22.4,0.207,27);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(111,62,26,495,34.0,0.54299,46);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(112,76,34,37,31.2,0.192,23);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(114,54,32,175,30.5,0.588,39);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(120,76,56,100,53.2,0.759,25);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(121,64,39,0,34.2,0.26,24);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(140,78,0,0,21.1,0.268,55);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(141,82,30,0,39.5,0.28603,38);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(142,52,26,63,32.5,0.318,22);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(148,78,0,0,33.7,0.218,65);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(152,86,28,155,34.3,1.189,42);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(153,82,42,485,40.6,0.687,23);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(165,74,18,156,29.9,0.722,41);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(166,66,25,0,32.5,0.256,22);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(173,60,42,48,43.5,0.67799,23);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(174,64,24,55,29.7,0.37,33);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(184,74,0,0,27.6,0.244,40);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(187,98,41,58,32.0,1.321,33);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(197,62,13,48,22.9,0.67799,23);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(206,76,29,280,37.5,0.605,57);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(208,64,27,87,33.2,0.289,21);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(209,84,33,0,35.5,0.355,41);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(219,66,0,0,37.8,0.261,41);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(224,66,15,56,23.6,0.66599,26);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(227,52,38,0,37.2,0.652,24);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(233,68,0,0,35.0,0.393996,29);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(236,84,21,192,35.9,0.586,51);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(237,90,27,0,44.1,0.68599,23);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(238,84,21,0,30.8,0.831,32);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(240,64,24,0,29.2,0.192,21);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(253,68,32,0,35.8,0.23802,25);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(256,56,39,0,30.1,0.557,30);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(267,64,42,0,40.0,1.101,24);
```

```
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(269,0,0,0,27.5,0.24,28);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(270,86,37,0,45.6,1.136,38);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(277,64,23,116,27.8,0.453996,23);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(279,62,10,278,25.3,0.88099,22);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(282,88,15,155,32.4,0.262,37);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(294,50,0,0,21.9,0.254,65);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(299,72,0,0,23.6,0.84,58);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(301,58,33,135,31.6,0.42204,25);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(309,68,28,205,32.9,0.875,30);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(310,66,30,0,26.2,0.313,41);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(312,74,17,96,26.6,0.433,27);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(313,50,10,85,29.5,0.626,25);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(316,80,11,64,19.3,0.284,30);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(319,78,0,0,23.5,0.129,59);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(320,60,12,231,27.5,0.527,31);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(323,90,33,29,26.8,0.731,43);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(327,70,0,0,35.1,0.2,37);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(337,76,0,0,31.2,0.342997,44);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(345,88,36,108,38.5,0.349,49);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(355,88,0,0,30.4,0.302,49);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(357,0,30,0,39.9,0.569,44);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(358,74,40,54,35.3,0.377995,48);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(359,76,36,249,36.5,0.875,29);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(365,54,28,83,34.0,0.499,30);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(367,64,17,0,21.0,0.252,21);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(369,102,28,140,32.8,0.233999,45);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(372,64,22,66,35.8,0.545,21);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(388,82,26,285,32.0,0.452,58);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(390,66,29,196,32.0,0.444,42);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(396,56,34,115,24.7,0.9441,39);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(397,66,40,0,34.3,0.196,22);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(399,70,31,0,34.9,0.24102,25);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(402,84,41,88,35.0,0.28603,35);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(412,84,23,310,42.4,1.07598,22);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(413,74,22,61,26.2,0.256,21);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(414,60,35,167,34.6,0.534,21);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(427,64,30,180,34.1,0.327996,38);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(431,74,16,85,30.4,0.551,38);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(439,88,0,0,36.8,0.727,31);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(450,64,13,95,21.2,0.415,23);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(452,68,32,210,39.9,0.381,25);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(455,62,30,0,33.6,0.212,38);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(457,68,28,71,30.2,0.364,24);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(461,62,0,0,21.8,0.41604,26);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(465,56,13,105,21.8,0.452,21);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(486,62,41,480,40.7,0.536,21);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(489,80,0,0,26.1,0.551,67);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(491,90,30,0,33.5,0.292,42);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(492,68,38,0,32.8,0.145,33);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(502,68,41,0,39.0,0.727,41);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(503,64,25,79,33.3,0.738,41);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(505,82,0,0,33.3,0.263,38);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(506,90,26,90,36.5,0.314,35);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(510,72,31,0,29.7,0.297,46);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(511,62,17,210,22.1,0.207,21);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(513,62,0,0,27.3,0.525,22);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(525,60,18,0,21.8,0.444,21);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(529,65,0,0,24.6,0.66,31);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(537,60,0,0,21.7,0.735,67);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(543,90,23,56,39.5,0.159,25);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(550,70,28,0,27.4,0.204,21);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(557,76,0,0,27.8,0.237,58);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(558,68,40,0,46.2,0.126,42);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(575,44,47,63,35.5,0.28,25);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(581,60,27,0,25.0,0.20602,27);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(588,86,27,156,33.3,1.1541,52);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(591,78,50,140,39.4,0.175,24);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(604,0,0,0,28.4,0.212,36);
```

```
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(609,62,13,182,24.0,0.138,23);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(611,58,22,194,32.9,0.593,36);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(618,82,24,0,28.2,1.28198,50);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(628,80,0,0,34.6,0.14402,45);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(629,65,22,0,24.7,0.14802,21);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(631,78,40,90,34.5,0.23802,24);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(632,60,0,0,26.2,0.342997,23);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(634,62,0,0,25.9,0.166998,31);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(636,74,0,0,28.8,0.153,48);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(652,74,40,77,34.1,0.268996,28);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(655,52,27,540,38.7,0.24,25);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(656,58,35,90,21.8,0.155,22);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(660,84,0,0,27.7,0.182,54);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(677,60,0,0,35.3,0.263,25);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(678,52,0,0,36.0,0.127,25);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(679,58,17,265,24.2,0.614,23);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(680,56,28,45,24.2,0.332,22);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(682,64,39,105,44.6,0.366,22);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(688,74,26,180,24.1,0.828,23);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(697,0,0,0,25.0,0.253,22);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(702,88,29,0,35.0,0.905,52);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(706,0,0,0,0.0,0.261,30);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(716,78,39,185,33.8,0.97,31);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(718,60,46,178,35.5,0.415,24);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(721,66,36,200,38.1,0.289,21);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(725,78,40,0,39.4,0.23602,38);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(741,44,20,94,30.8,0.4,26);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(747,74,41,57,46.3,1.09599,32);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(749,62,0,0,24.3,0.17802,50);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(755,88,39,110,36.5,1.057,37);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(759,92,0,0,35.5,0.278,66);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(761,74,31,0,44.0,0.402997,43);
INSERT INTO PAL_AUTOML_TEST_TAB VALUES(767,70,31,0,30.4,0.315,23);
```

## 5.7.3 Optimization

AutoML uses genetic programming algorithm (GA) as the optimization algorithm, in which NSGA-II is used as selection algorithm to support multiple objectives.

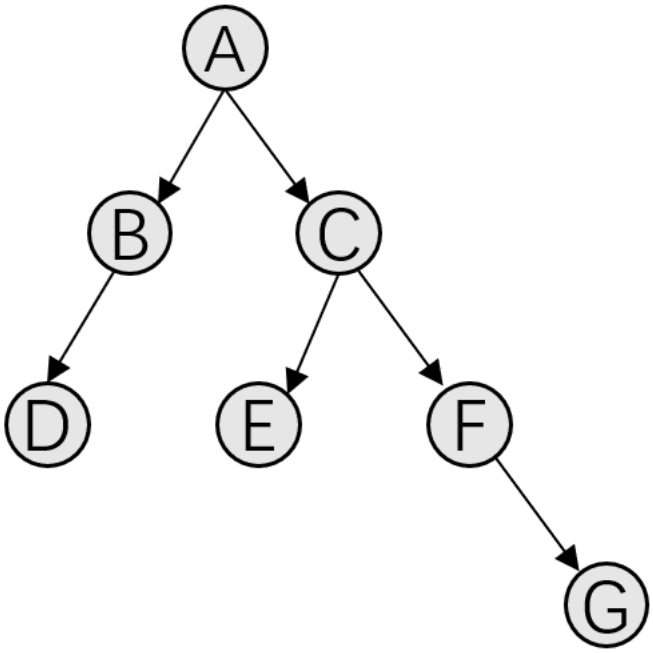
In general, the evolution usually starts from a population of randomly generated individuals, and is an iterative process, with the population in each iteration called a generation. In each generation, the fitness of every individual in the population is evaluated; the fitness is usually the value of the objective function in the optimization problem being solved. The more fit individuals are stochastically selected from the current population, and each individual's genome is modified (recombined and possibly randomly mutated) to form a new generation. The new generation of candidate solutions is then used in the next iteration of the algorithm. Commonly, the algorithm terminates when either a maximum number of generations has been produced, or a satisfactory fitness level has been reached for the population.

### Genetic Programming Algorithm

#### Individual Representation

In AutoML, individual in GA is represented in as a tree structure. By specifying the maximum and minimum number of layers of the tree structure, GA can randomly generate individuals that satisfy the condition at a given `config_dict`.

- Tree Structure

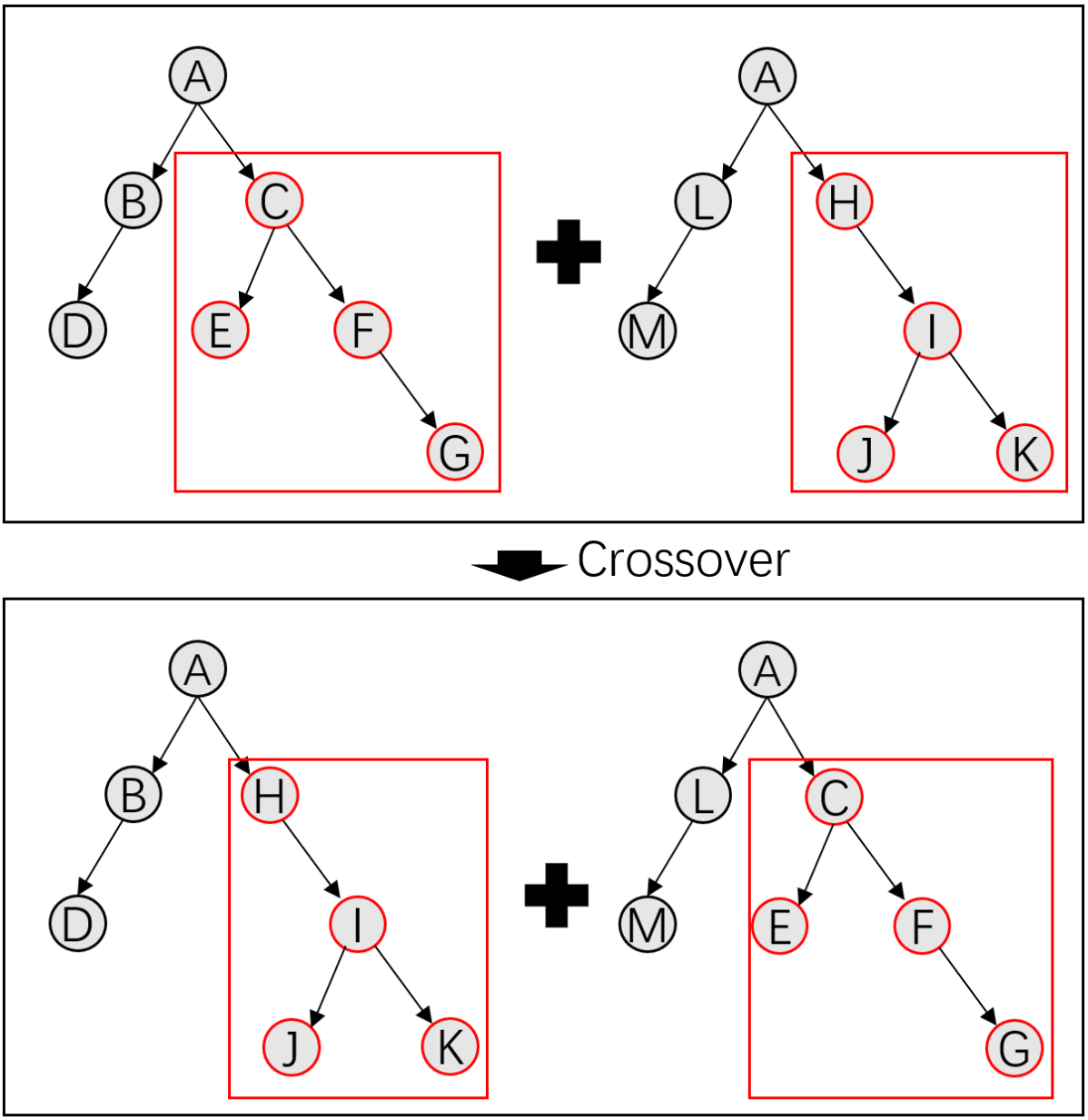



**Selection**

Selection is a process whereby certain individuals from the current generation are selected as parents of the next generation. The algorithm currently used in AutoML is NSGA-II.

**Crossover**

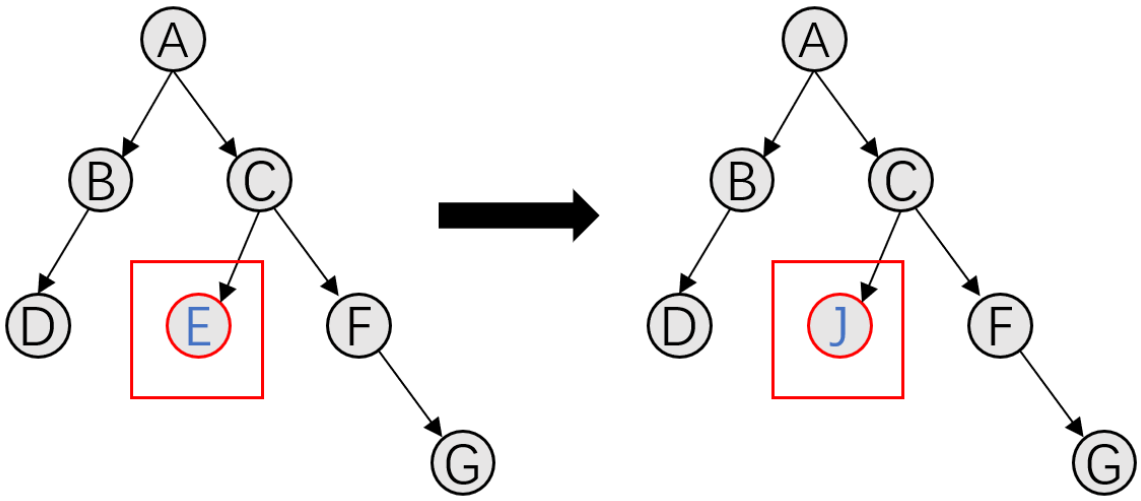
Select two individuals in the population that can be crossover, and randomly exchange a pair of leaf nodes of equal type but different values or child nodes with different structures.



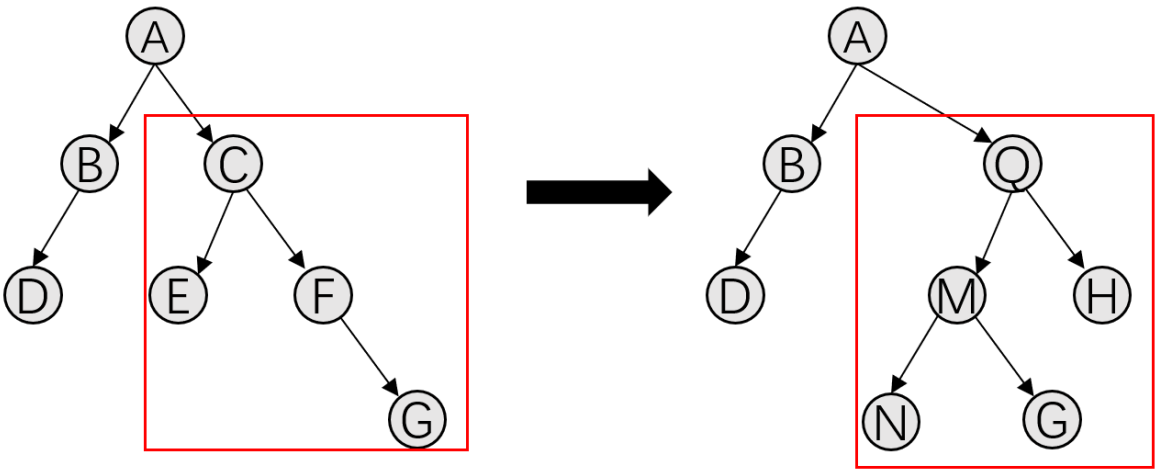

**Mutation**

There are two types of mutation in AutoML, leaf node mutation and child node mutation.

- Leaf Node Mutation:
  Randomly replaces one of the leaf nodes with other leaf nodes of the same type.




- Child Node Mutation:
  Replaces the entire child node with a newly generated child node that meets the requirements.




### Iteration

In each iteration, AutoML randomly selects an operation from crossover, mutation, and regeneration with user input probability to generate a new individual and evaluate the score with the giving SCORES. Then uses the selection algorithm to select population_size individuals from the population and offspring as the next generation population.

### Control Parameters

- The following are the main control parameters of the GA:

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| GENERATIONS | INTEGER | 5 | The number of iterations of the pipeline optimization. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| POPULATION_SIZE | INTEGER | 20 | The number of individuals in each generation in genetic programming algorithm. | |
| OFFSPRING_SIZE | INTEGER | POPULATION_SIZE | The number of offsprings to produce in each generation. | Only valid when SUCCESSIVE_HALVING is set to 0. |
| MUTATION_RATE | DOUBLE | 0.9 | The mutation rate for the genetic programming algorithm. | |
| CROSSOVER_RATE | DOUBLE | 0.1 | The crossover rate for genetic programming algorithm. | |
| MIN_LAYER | INTEGER | 1 | The minimum number of operators in pipeline. | |
| MAX_LAYER | INTEGER | 5 | The maximum number of operators in pipeline | |

- **GENERATIONS**
  Generally, GENERATIONS increases the total training time, but in most cases, the iteration will terminate when the EARLY_STOP condition is reached. And the user can also stop the training process manually and get the optimal solution from logging information.
- **POPULATION_SIZE**
  If there are too few individuals, GA has a few possibilities to perform crossover and only a small part of search space is explored. On the other hand, if there are too many individuals, GA slows down.
- **OFFSPRING_SIZE**
  It controls the number of new individuals generated in each iteration by genetic operations, from population.
- **MUTATION_RATE**
  Represents the random search ability.
  A suitable value can prevent the GA from falling into a local optimum.
- **CROSSOVER_RATE**
  Represents the local search ability.
  A larger crossover rate will cause GA to converge to a local optimum faster.
- **REPRODUCTION_RATE**
  (implicit)=1-MUTATION_RATE+CROSSOVER_RATE
  The sum of MUTATION_RATE and CROSSOVER_RATE cannot be greater than 1.0.
  When the sum is less than 1.0, the remaining probability will be used to regenerate.

## Successive Halving

AutoML can be time consuming when it runs on large datasets. To reduce the time consumption, we introduce successive halving strategy during the iteration process of the genetic algorithm. Successive

halving dynamically decreases the population size and increases the budget, the size of subset data used for evaluation.

Denote $p_i$ and $b_i$ the population size and the budget at the $i^th$ generation, $p_i$ and $b_i$ will be changed during the iteration as follows:

$$p_i = \lfloor P/(2^{\square i \square(\log_2(p/m)+1\square/(G+1)\square}) \rfloor$$

$$b_i = b2^{\square i \square(\log_2(B/b)+1\square/(G+1)\square}$$

Where `m` is the minimal population size to keep in the parent population; `b` is the initial budget and `B` the maximum budget; `G` and `P` are respectively the number of generations and the initial population size specified in GA.

**Control Parameters**

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| SUCCESSIVE_HALVING | INTEGER | 1 | Indicates whether to use successive halving.<br>• 0: do not use successive halving<br>• 1: use successive halving | |
| MIN_BUDGET | INTEGER | (data size)/5 | Minimum data size used in successive halving iteration. | |
| MAX_BUDGET | INTEGER | data size | Maximum data size used in successive halving iteration | |
| MIN_INDIVICUALS | DOUBLE | 3 | Minimum population size in successive halving iteration. | |

## 5.7.4 AutoML Tutorial

A PAL AutoML Tutorial for classification problems.

See Tutorial Dataset [page 1026] for details about data table used in this section.

### Verify and Customize Config Dict

Users can modify config_dict based on experience and data characteristics and use PAL_AUTOML_CONFIG to verify and save modified result into table.

```
set schema DM_PAL;
drop table PAL_PARAMETER_TAB;
create column table PAL_PARAMETER_TAB (
    "PARAM_NAME" nvarchar(256),
```

```
    "INT_VALUE" integer,
    "DOUBLE_VALUE" double,
    "STRING_VALUE" nvarchar(5000)
);
INSERT INTO PAL_PARAMETER_TAB VALUES ('PIPELINE_TYPE', null, null, 'classifier');
INSERT INTO PAL_PARAMETER_TAB VALUES ('CONFIG_DICT', null, null,   'default');
INSERT INTO PAL_PARAMETER_TAB VALUES ('CONFIG_REMOVE', null, null,
'SVM_Classifier');
INSERT INTO PAL_PARAMETER_TAB VALUES ('CONFIG_REMOVE', null, null,
'OneHotEncoder'); -- no categorical feature
INSERT INTO PAL_PARAMETER_TAB VALUES ('CONFIG_REMOVE', null, null,
'LabelEncoder');
INSERT INTO PAL_PARAMETER_TAB VALUES ('CONFIG_MODIFY', null, null,
'{"RDT_Classifier":{"TREES_NUM":[100, 200]}}');
INSERT INTO PAL_PARAMETER_TAB VALUES ('CONFIG_MODIFY',
null, null, '{"HGBT_Classifier":{"MAX_DEPTH":{"range":[1,1,4]},"ETA":
[0.01,0.1,0.5],"NODE_SIZE":{"range":[1,2,10]}}}');
drop table PAL_AUTOML_CONFIG_TAB;
create column table PAL_AUTOML_CONFIG_TAB (
    "ROW_INDEX" integer,
    "CONTENT" nvarchar(5000)
);
drop table PAL_AUTOML_CONFIGINFO_TAB;
create column table PAL_AUTOML_CONFIGINFO_TAB (
    "NAME" nvarchar(256),
    "TYPE" nvarchar(256),
    "CONFIG" nvarchar(1000)
);
DO BEGIN
lt_para = select * from PAL_PARAMETER_TAB;
call _SYS_AFL.PAL_AUTOML_CONFIG(:lt_para, lt_config, lt_info);
insert into PAL_AUTOML_CONFIG_TAB select * from :lt_config;
insert into PAL_AUTOML_CONFIGINFO_TAB select * from :lt_info;
END;
select * from PAL_AUTOML_CONFIG_TAB;
select * from PAL_AUTOML_CONFIGINFO_TAB;
```

**Result**

## Execute AUTOML

Uses customized config_dict and data to generate optimized pipelines, and output elite pipeline model directly if necessary.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_PARAMETER_TAB;
CREATE COLUMN TABLE PAL_PARAMETER_TAB (
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR(1000)
);
-- use the same parameters in PAL_AUTOML_CONFIG to get the same config_dict used
in PAL_AUTOML_FIT
INSERT INTO PAL_PARAMETER_TAB VALUES ('PIPELINE_TYPE', null, null, 'classifier');
INSERT INTO PAL_PARAMETER_TAB VALUES ('CONFIG_DICT', null, null, 'default');
INSERT INTO PAL_PARAMETER_TAB VALUES ('CONFIG_REMOVE', null, null,
'SVM_Classifier');
INSERT INTO PAL_PARAMETER_TAB VALUES ('CONFIG_REMOVE', null, null,
'OneHotEncoder');
INSERT INTO PAL_PARAMETER_TAB VALUES ('CONFIG_REMOVE', null, null,
'LabelEncoder');
```

```
INSERT INTO PAL_PARAMETER_TAB VALUES ('CONFIG_MODIFY', null, null,
'{"RDT_Classifier":{"TREES_NUM":[100, 200]}}');
INSERT INTO PAL_PARAMETER_TAB VALUES ('CONFIG_MODIFY',
null, null, '{"HGBT_Classifier":{"MAX_DEPTH":{"range":[1,1,4]},"ETA":
[0.01,0.1,0.5],"NODE_SIZE":{"range":[1,2,10]}}}');
-- or use the saved CONFIG_DICT directly
-- INSERT INTO PAL_PARAMETER_TAB SELECT 'CONFIG_DICT', NULL, NULL, CONTENT FROM
PAL_AUTOML_CONFIG_TAB WHERE ID = 0;
INSERT INTO PAL_PARAMETER_TAB VALUES ('GENERATIONS', 5, null, null);
INSERT INTO PAL_PARAMETER_TAB VALUES ('POPULATION_SIZE', 10, null, null);
INSERT INTO PAL_PARAMETER_TAB VALUES ('OFFSPRING_SIZE', 10, null, null);
INSERT INTO PAL_PARAMETER_TAB VALUES ('MUTATION_RATE', null, 0.7, null);
INSERT INTO PAL_PARAMETER_TAB VALUES ('CROSSOVER_RATE', null, 0.1, null); --
regeneration rate is now 1-0.7-0.1 = 0.2
INSERT INTO PAL_PARAMETER_TAB VALUES ('MIN_LAYER', 1, null, null);
INSERT INTO PAL_PARAMETER_TAB VALUES ('MAX_LAYER', 5, null, null);
INSERT INTO PAL_PARAMETER_TAB VALUES ('OUTPUT_MODEL', 1, null, null); -- set to
0 if user want to select pipeine from elite pipelines by themselves
INSERT INTO PAL_PARAMETER_TAB VALUES ('ELITE_NUMBER', 3, null, null);
INSERT INTO PAL_PARAMETER_TAB VALUES ('RANDOM_SEED', 1, null, null);
INSERT INTO PAL_PARAMETER_TAB VALUES ('PROGRESS_INDICATOR_ID', null, null,
'AUTOML_TEST');
DROP TABLE PAL_AUTOML_PIPELINES_TAB;
CREATE COLUMN TABLE PAL_AUTOML_PIPELINES_TAB (
    "ID" INTEGER,
    "PIPELINE" NVARCHAR(1000),
    "SCORES" NVARCHAR(1000)
);
DROP TABLE PAL_AUTOML_MODEL_TAB;
CREATE COLUMN TABLE PAL_AUTOML_MODEL_TAB (
    "ROW_INDEX" INTEGER,
    "CONTEXT" NVARCHAR(5000)
);
DO BEGIN
lt_data = select * from PAL_AUTOML_TRAIN_TAB;
lt_para = select * from PAL_PARAMETER_TAB;
CALL _SYS_AFL.PAL_AUTOML_FIT(:lt_data, :lt_para, lt_pipelines, lt_model,
lt_info);
insert into PAL_AUTOML_PIPELINES_TAB select * from :lt_pipelines;
insert into PAL_AUTOML_MODEL_TAB select * from :lt_model;
END;
select * from PAL_AUTOML_PIPELINES_TAB;
select * from PAL_AUTOML_MODEL_TAB;
```

### Result

- PAL_AUTOML_PIPELINES_TAB

Rows (3) SQL

| | ID | PIPELINE | SCORES |
|---|---|---|---|
| 1 | 0 | {"RDT_Classifier":{"args":{"TREES_NUM":20 | {"ACCURACY":0.7117686258829802,"AUC":0.7945441 |
| 2 | 1 | {"RDT_Classifier":{"args":{"TREES_NUM":20 | {"ACCURACY":0.7199120351859257,"AUC":0.7944738 |
| 3 | 2 | {"RDT_Classifier":{"args":{"TREES_NUM":20 | {"ACCURACY":0.7133946421431429,"AUC":0.7945019 |

- PAL_AUTOML_MODEL_TAB

Rows (198)

| | ROW_INDEX | CONTEXT |
|---|---|---|
| 1 | 0 | {"pipeline":{"RDT_Classifier":{"args":{"TREES_NUM":200,"NODE_SIZE":8,"CALCULATE_OOB":0,"SAMPLE_F |
| 2 | 1 | 1","0"],"num_every_class":[39,218]},{"node_id":3,"record":202,"sc":"0","sp":{"field_id":1,"cond":"31.5","op":"lt |
| 3 | 2 | _every_class":[22,36]},{"node_id":41,"record":46,"sc":"0","sp":{"field_id":3,"cond":"36.9","op":"lt"},"class_nam |
| 4 | 3 | ode_id":18,"record":337,"sc":"0","sp":{"field_id":5,"cond":"27.5","op":"ge"},"class_names":["1","0"],"num_every |
| 5 | 4 | "field_id":5,"cond":"29.5","op":"lt"},"class_names":["1","0"],"num_every_class":[11,0]},{"node_id":57,"record": |
| 6 | 5 | 28.35","op":"lt"},"class_names":["1","0"],"num_every_class":[4,12]},{"node_id":34,"record":56,"sc":"1","sp":{"fi |
| 7 | 6 | "lt"},"class_names":["1","0"],"num_every_class":[1,113]},{"node_id":7,"record":89,"sc":"0","sp":{"field_id":3,"c |
| 8 | 7 | d_id":3,"cond":"27.4","op":"ge"},"class_names":["1","0"],"num_every_class":[11,12]},{"node_id":46,"record":1 |
| 9 | 8 | "],"num_every_class":[26,9]},{"node_id":19,"record":101,"sc":"0","sp":{"field_id":2,"cond":"9","op":"ge"},"class |
| 10 | 9 | sp":{"field_id":5,"cond":"27.5","op":"ge"},"class_names":["1","0"],"num_every_class":[40,20]}]},{"h":11,"nnode |
| 11 | 10 | ":"139.5","op":"ge"},"class_names":["1","0"],"num_every_class":[23,10]},{"node_id":38,"record":20,"sc":"1","sp |
| 12 | 11 | _every_class":[0,35]},{"node_id":17,"record":72,"sc":"0","sp":{"field_id":5,"cond":"27.5","op":"ge"},"class_nam |
| 13 | 12 | .5","op":"ge"},"class_names":["1","0"],"num_every_class":[22,9]},{"node_id":56,"record":14,"sc":"1","sp":{"field |
| 14 | 13 | "num_every_class":[71,258]},{"node_id":2,"record":277,"sc":"0","sp":{"field_id":1,"cond":"36.5","op":"lt"},"clas |
| 15 | 14 | ":23,"sc":"0","sp":{"field_id":3,"cond":"25.4","op":"ge"},"class_names":["1","0"],"num_every_class":[1,22]},{"no |
| 16 | 15 | ry_class":[27,11]},{"node_id":46,"record":61,"sc":"1","sp":{"field_id":2,"cond":"153.5","op":"ge"},"class_names |
| 17 | 16 | "record":112,"sc":"0","sp":{"field_id":2,"cond":"153.5","op":"lt"},"class_names":["1","0"],"num_every_class":[54 |
| 18 | 17 | "0.323999","op":"ge"},"class_names":["1","0"],"num_every_class":[9,0]},{"node_id":25,"record":87,"sc":"0","sp |

## Progress Status Monitoring

When PAL_AUTOML_FIT is running, use the PROGRESS_INDICATOR_ID entered in the parameters table to capture log information.

- Ensure the uniqueness of the PROGRESS_INDICATOR_ID among all the PAL_AUTOML_FIT sessions running in the same instance to avoid confusing log information.

Execute in another SQL connection with the same user.

```
SET SCHEMA DM_PAL;
SELECT * FROM _SYS_AFL.FUNCTION_PROGRESS_IN_AFLPAL WHERE EXECUTION_ID
='AUTOML_TEST';
```

## Train Pipeline Model with Self-Selected Elite Pipeline

Users can choose other elite pipelines generated in PAL_AUTOML_FIT to train pipeline model.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_PARAMETER_TAB;
CREATE COLUMN TABLE PAL_PARAMETER_TAB (
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR(1000)
);
DROP TABLE PAL_AUTOML_MODEL_REFIT_TAB;
CREATE COLUMN TABLE PAL_AUTOML_MODEL_REFIT_TAB (
    "ROW_INDEX" INTEGER,
```

```
    "CONTEXT" NVARCHAR(5000)
);
insert into PAL_PARAMETER_TAB select 'PIPELINE', NULL, NULL, PIPELINE from
PAL_AUTOML_PIPELINES_TAB where ID = 1;
DO BEGIN
lt_data = select * from PAL_AUTOML_TRAIN_TAB;
lt_para = select * from PAL_PARAMETER_TAB;
CALL _SYS_AFL.PAL_PIPELINE_FIT(:lt_data, :lt_para, lt_model, lt_info);
insert into PAL_AUTOML_MODEL_REFIT_TAB select * from :lt_model;
END;
select * from PAL_AUTOML_MODEL_REFIT_TAB;
```

## Predict with Optimized Pipeline Model

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_PARAMETER_TAB;
CREATE COLUMN TABLE PAL_PARAMETER_TAB (
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR(1000)
);
CALL _SYS_AFL.PAL_PIPELINE_PREDICT(PAL_AUTOML_TEST_TAB, PAL_AUTOML_MODEL_TAB,
PAL_PARAMETER_TAB, ?, ?);
```

### Result

- Predict result

Rows (154) SQL

| | ID | SCORES |
|---|---|---|
| 1 | 7 | 1 |
| 2 | 16 | 1 |
| 3 | 17 | 0 |
| 4 | 27 | 0 |
| 5 | 32 | 0 |
| 6 | 35 | 0 |
| 7 | 43 | 1 |
| 8 | 54 | 1 |
| 9 | 56 | 1 |
| 10 | 65 | 0 |
| 11 | 73 | 0 |
| 12 | 76 | 1 |
| 13 | 79 | 0 |
| 14 | 85 | 0 |
| 15 | 86 | 0 |
| 16 | 94 | 0 |
| 17 | 96 | 0 |
| 18 | 99 | 1 |
| 19 | 101 | 0 |
| 20 | 103 | 0 |
| 21 | 106 | 0 |
| 22 | 111 | 1 |

## Further Optimize Pipeline

Based on the result of the first run of PAL_AUTOML_FIT, users can adjust config_dict to narrow the searching space, and execute PAL_AUTOML_FIT again to obtain a further optimized pipeline.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_PARAMETER_TAB;
CREATE COLUMN TABLE PAL_PARAMETER_TAB (
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR(1000)
);
INSERT INTO PAL_PARAMETER_TAB VALUES ('PIPELINE_TYPE', null, null, 'classifier');
INSERT INTO PAL_PARAMETER_TAB VALUES ('CONFIG_DICT', null, null, 'empty');
```

```
INSERT INTO PAL_PARAMETER_TAB VALUES ('CONFIG_ADD', null,
null, '{"HGBT_Classifier":{"ITER_NUM":[100],"MAX_DEPTH":{"range":[1,1,11]},"ETA":
[0.001,0.01,0.1,0.5,1.0],"NODE_SIZE":{"range":[1,1,21]}}}');
INSERT INTO PAL_PARAMETER_TAB VALUES ('CONFIG_ADD', null, null, '{"SMOTE":
{"K_NEAREST_NEIGHBOURS":[1,3],"METHOD":[1]}}');
INSERT INTO PAL_PARAMETER_TAB VALUES ('CONFIG_ADD', null, null,
'{"FS_unsupervised":{"FS_METHOD":[9],"HAS_ID":[1],"TOP_K_BEST":[10]}}');
INSERT INTO PAL_PARAMETER_TAB VALUES ('GENERATIONS', 5, null, null);
INSERT INTO PAL_PARAMETER_TAB VALUES ('POPULATION_SIZE', 10, null, null);
INSERT INTO PAL_PARAMETER_TAB VALUES ('OFFSPRING_SIZE', 10, null, null);
INSERT INTO PAL_PARAMETER_TAB VALUES ('MUTATION_RATE', null, 0.7, null);
INSERT INTO PAL_PARAMETER_TAB VALUES ('CROSSOVER_RATE', null, 0.1, null);
INSERT INTO PAL_PARAMETER_TAB VALUES ('MIN_LAYER', 1, null, null);
INSERT INTO PAL_PARAMETER_TAB VALUES ('MAX_LAYER', 5, null, null);
INSERT INTO PAL_PARAMETER_TAB VALUES ('OUTPUT_MODEL', 1, null, null);
INSERT INTO PAL_PARAMETER_TAB VALUES ('ELITE_NUMBER', 3, null, null);
DROP TABLE PAL_AUTOML_PIPELINES_TAB;
CREATE COLUMN TABLE PAL_AUTOML_PIPELINES_TAB (
    "ID" INTEGER,
    "PIPELINE" NVARCHAR(1000),
    "SCORES" NVARCHAR(1000)
);
DROP TABLE PAL_AUTOML_MODEL_TAB;
CREATE COLUMN TABLE PAL_AUTOML_MODEL_TAB (
    "ROW_INDEX" INTEGER,
    "CONTEXT" NVARCHAR(5000)
);
DO BEGIN
lt_data = select * from PAL_AUTOML_TRAIN_TAB;
lt_para = select * from PAL_PARAMETER_TAB;
CALL _SYS_AFL.PAL_AUTOML_FIT(:lt_data, :lt_para, lt_pipelines, lt_model,
lt_info);
insert into PAL_AUTOML_PIPELINES_TAB select * from :lt_pipelines;
insert into PAL_AUTOML_MODEL_TAB select * from :lt_model;
END;
select * from PAL_AUTOML_PIPELINES_TAB;
select * from PAL_AUTOML_MODEL_TAB;
```

## 5.8 Preprocessing Algorithms

The records in business database are usually not directly ready for predictive analysis due to the following reasons:

- Some data come in large amount, which may exceed the capacity of an algorithm.
- Some data contains noisy observations which may hurt the accuracy of an algorithm.
- Some attributes are badly scaled, which can make an algorithm unstable.

To address the above challenges, PAL provides several convenient algorithms for data preprocessing.

## 5.8.1 Categorical Principal Component Analysis

Principal component analysis (PCA) aims at reducing the dimensionality of multivariate data while accounting for as much of the variation in the original data set as possible.

The application of this dimension reduction technique assumes three assumptions:

1. Linear relationship between different variables
2. Large Variances of variable imply important or interesting structure
3. Principal Components are orthogonal

With these assumptions, the problem can be simplified and transformed to diagonalize the covariance / correlation matrix with diagonal entry listed in descending order. The eigen value in transformed diagonal entry represents the variance for each component, while the off-diagonal entry is all 0, which represents the covariances between different components are all 0s.

This implies principal components are orthogonal. The eigen vectors which form the transformation matrix change basis from the original vector space into this new basis spanned by the principal components and it is also named component loading. After transformation, each data instance will be represented in the linear space spanned by the principal components, which are called component scores.

The method can be formalized as an optimization problem to minimize following loss function:

$$\sigma(Z,A) = \frac{1}{p}tr(X-ZA^T)^T(X-ZA^T)=\frac{1}{p}\sum_{j=1}^{p}(x_j-Za_j^T)^T(x_j-Za_j^T)$$

Here $X$ is $n \times p$ data matrix. $Z$ is $n \times r$ component scores with component dimension $1 \leq r \leq p$. $A$ is component loading with dimension $p \times r$.

This method has the advantage that it is non-parametric, highly efficient in run-time performance and applicable to any dataset. Since PCA is basically a change of basis operation, thus the total variance will not change. The value of total variance after dimension reduction divided by the original total variance is the percentage of information retained.

Current implementation also supports dataset with categorical variables. To be compatible with other algorithms, we assume all DOUBLE data type of variables are numeric data and all VARCHAR/NVARCHAR data type of variables are categorical data. Variable with INTEGER data type can be treated either as numeric data type or categorical data type. By default it will be deemed as numeric data, unless the variable is listed in CATEGORICAL_VARIABLE parameter.

Category Quantification for each categorical variable tries to assign a numeric value to each category in the principal component vector space. For each principal dimension, category quantification can assign different values to each category, this is called multiple nominal quantification. Current implementation impose rank-one restriction, which is called single nominal quantification.

Suppose the quantification matrix for some variable is $W_j$, which has dimension $K_j \times r$, where $K_j$ is the number of distinct category value, $r$ is principal component dimension. Single Quantification assumes that $W_j = q_j \times a_j$. Here $q_j$ is the quantification for each category in original vector space, $a_j$ is component loading.

When coping with single nominal quantification, the optimization problem can be formalized as below:

$$\sigma(Z,A,X^*) = \frac{1}{p}tr(X^*-ZA^T)^T(X^*-ZA^T)=\frac{1}{p}\sum_{j=1}^{p}(x_j^* -Za_j^T)^T(X_j^*-Za_j^T)$$

Here $X^*$ represents the data matrix after quantification with constraints $X^{*T}1_n = 0_p$ and $\mathrm{diag}\left(\frac{X^{*T}X^*}{n-1}\right)=I_p$ .

Current implementation use Alternating Least Square algorithm to find the optimal scaling quantification for categorical data.

## Prerequisites

- No missing or null data in the inputs
- Each column must have at least two different values

## _SYS_AFL.PAL_CATPCA

This procedure is used to perform principal component analysis.

It uses SVD algorithm to compute eigen value and eigen vector. Currently it supports two SVD algorithms, one is LANCZOS algorithm, the other is Jacobi Algorithm. Both algorithms have their pros and cons, so it depends on the user to choose either for different cases.

When performing dimension reduction with PCA, it is required to provide the reduced dimension N_COMPONENTS. The smaller parameter N_COMPONENTS is, the faster the procedure will finish. This means user should input smallest dimension that meets requirement.

If the input dimension is far smaller than the full dimension, it is recommended to use LANCZOS algorithm. However, LANCZOS algorithm is a convergence based approximation algorithm and user needs to set a maximum iteration count to prevent the algorithm running too long. Otherwise there can be cases where this algorithm fails to converge and leads to failure. When confronting this error, user should increase the maximum iteration count or increase the convergence tolerance. Another disadvantage is that LANCZOS algorithm can not be used to compute full dimension SVD. In this case, current implementation will always switch to Jacobi Algorithm even user has specified LANCZOS Algorithm.

When N_COMPONENTS is closer to full dimension, it is recommended to use Jacobi Algorithm to compute SVD because in this case this algorithm runs faster. The Jacobi Algorithm will always succeed. However, it will always compute full dimension of SVD and then retrieve the first nth dimension, so it will only outperform LANCZOS Algorithm when the required dimension is closer to full dimension.

**Overview of all Tables**

| Position | Table Name | Type |
|---|---|---|
| 1 | Data Table | In |
| 2 | Parameter Table | In |
| 3 | Loadings Table | Out |
| 4 | Loadings Information Table | Out |
| 5 | Scores Table | Out |

| Position | Table Name | Type |
|---|---|---|
| 6 | Scaling Information Table | Out |
| 7 | Category Quantification Table | Out |
| 8 | Statistic Table | Out |

### Input Table

### Data Table

| Column | Data Type | Description |
|---|---|---|
| First Column | INTEGER, BIGINT, VARCHAR, NVARCHAR | Data ID |
| Variable Columns | INTEGER, DOUBLE, DECIMAL(p, s), VARCHAR, NVARCHAR | Data variables |

### Parameter Table

Mandatory Parameters

| Name | Data Type | Description |
|---|---|---|
| N_COMPONENTS | INTEGER | Number of Components to keep |

Optional Parameters

The following parameter is optional. If it is not specified. PAL will use its default value.

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| SCALING | INTEGER | 0 | Specifies whether variable should be scaled to unit variance before analysis.<br>**Note**<br>when data instance contains category variable, this parameter will always be treated as 1.<br>Valid value:<br>• 0: No<br>• 1: Yes |
| SCORES | INTEGER | 0 | Specifies whether to output component scores.<br>Valid value<br>• 0: No<br>• 1: Yes |

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| THREAD_RATIO | DOUBLE | 1.0 | Specifies the ratio of available total thread number. The value range is [0,1]. 0 means 1 thread; 1 means all available threads. |
| CATEGORICAL_VARIABLE | VARCHAR, NVARCHAR | | This is used to specify categorical variable name. The parameter can be specified multiple times, one time per variable. Only INTEGER or VARCHAR/NVARCHAR type variable should be specified. |
| COMPONENT_TOL | DOUBLE | | Specifies the threshold of ratio between current singular value and largest singular value whose component will be dropped in final output. If not specified, no component will be dropped. Valid range is [0, 1). |
| SEED | INTEGER | 0 | Seed of random generator to generate initial category quantification. When value is 0, the initial value will be generated differently for each run. Valid range is [0,∞). |
| MAX_ITERATION | INTEGER | 100 | Specifies the maximum iteration count for optimal scaling quantification. Valid range is [1,∞). |
| CONVERGE_TOL | DOUBLE | 1E-5 | Specifies the convergence criteria to detect convergence for optimal scaling quantification. Valid range is (0,1). |
| LANCZOS_ITERATION | INTEGER | 1000 | Specifies maximum iteration when using LANCZOS algorithm to compute SVD. Valid only when SVD_CALCULATOR is 0. Valid range is [1,∞). |

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| SVD_CALCULATOR | INTEGER | 0 | Specifies the algorithm to calculate SVD.<br>**Note**<br>When N_COMPONENTS equals to data dimension, this parameter will always be 1.<br>Valid Values:<br>• 0: LANCZOS Algorithm<br>• 1: Jacobi Algorithm |

### Output Table

**Table** *Loadings*

| Column | Data Type | Column Name | Description |
|---|---|---|---|
| First Column | NVARCHAR(128) | VARIABLE_NAME | Variable Name |
| Second Column | INTEGER | COMPONENT_ID | Component IDs are marked from 1 ordered by decreasing variance |
| Third Column | DOUBLE | COMPONENT_LOADING | Component Loadings form the change of basis to transform data from original vector space to vector space spanned by principal components |

**Table** *Loadings Information*

Current implementation will only output three kinds of metrics: Standard Deviation, Variance Proportion, Cumulative Variance Proportion.

| Column | Data Type | Column Name | Description |
|---|---|---|---|
| First Column | INTEGER | COMPONENT_ID | Component ID are marked from 1 ordered by decreasing variance |
| Second Column | NVARCHAR(128) | METRIC_NAME | Metric Name |
| Third Column | DOUBLE | METRIC_VALUE | Metric Value |

**Table** *Scores*

This output table is only relevant when SCORES parameter is 1.

| Column | Data Type | Column Name | Description |
|---|---|---|---|
| First Column | Refer to the ID Column in Data Table | Data ID Column Name in Data Table | Data ID |
| Second Column | INTEGER | COMPONENT_ID | Component IDs are marked from 1 ordered by decreasing variance |
| Third Column | DOUBLE | COMPONENT_SCORE | Component Score |

**Table** *Scaling Information*

| Column | Data Type | Column Name | Description |
|---|---|---|---|
| First Column | NVARCHAR(128) | VARIABLE_NAME | Variable Name |
| Second Column | DOUBLE | MEAN | Mean of Variable |
| Third Column | DOUBLE | SCALE | Standard Deviation of Variable |

**Table** *Category Quantification*

This output table is only relevant when data instance contains categorical variable.

| Column | Data Type | Column Name | Description |
|---|---|---|---|
| First Column | NVARCHAR(128) | VARIABLE_NAME | Variable Name |
| Second Column | NVARCHAR(5000) | CATEGORY_VALUE | Discrete Value |
| Third Column | INTEGER | COMPONENT_ID | Component IDs are set 0 for single nominal variable quantification |
| Fourth Column | DOUBLE | QUANTIFICATION | Category Quantification |

**Table** *Statistic*

This output table is only relevant when data instance contains categorical variable.

| Column | Data Type | Column Name | Description |
|---|---|---|---|
| First Column | NVARCHAR(128) | STAT_NAME | Statistical Name |
| Second Column | NVARCHAR(1024) | STAT_VALUE | Statistical Value |

### Example: Perform Categorical Principal Component Analysis

### Example 1: Perform PCA with LANCZOS SVD Calculator

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_PCA_DATA_TBL;
CREATE COLUMN TABLE PAL_PCA_DATA_TBL ("ID" INTEGER, "X1" DOUBLE, "X2"
NVARCHAR(5), "X3" DOUBLE, "X4" INTEGER, "X5" DOUBLE, "X6" DOUBLE);
INSERT INTO PAL_PCA_DATA_TBL VALUES (1, 12, 'A', 20, 44, 48, 16);
INSERT INTO PAL_PCA_DATA_TBL VALUES (2, 12, 'B', 25, 45, 50, 16);
INSERT INTO PAL_PCA_DATA_TBL VALUES (3, 12, 'C', 21, 45, 50, 16);
INSERT INTO PAL_PCA_DATA_TBL VALUES (4, 13, 'A', 21, 46, 51, 17);
INSERT INTO PAL_PCA_DATA_TBL VALUES (5, 14, 'C', 24, 46, 51, 17);
INSERT INTO PAL_PCA_DATA_TBL VALUES (6, 22, 'A', 25, 54, 58, 26);
```

```
INSERT INTO PAL_PCA_DATA_TBL VALUES (7, 22, 'D', 26, 55, 58, 27);
INSERT INTO PAL_PCA_DATA_TBL VALUES (8, 17, 'A', 21, 45, 52, 17);
INSERT INTO PAL_PCA_DATA_TBL VALUES (9, 15, 'D', 24, 45, 53, 18);
INSERT INTO PAL_PCA_DATA_TBL VALUES (10, 23, 'C', 23, 53, 57, 24);
INSERT INTO PAL_PCA_DATA_TBL VALUES (11, 25, 'B', 23, 55, 58, 25);
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL ("PARAM_NAME" VARCHAR(256), "INT_VALUE"
INTEGER, "DOUBLE_VALUE" DOUBLE, "STRING_VALUE" VARCHAR(1000));
--SCALING: 0 means use covariance matrix to compute PCA, 1 means use correlation
matrix to compute PCA. Default is 0.
--When coping with categorical data, it is forced to set SCALING to 1.
INSERT INTO PAL_PARAMETER_TBL VALUES ('SCALING', 1, NULL, NULL);
-- Whether output componnet score data. 0 means no output score data; 1 means
output score data. Default is 0.
INSERT INTO PAL_PARAMETER_TBL VALUES ('SCORES', 1, NULL, NULL);
-- Proportion of CPU resources to use in current computation. Value should be
between 0 and 1. Default is 1
INSERT INTO PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 0.0, NULL);
-- This is a mandatory parameter. It indicates number of components to extract.
Value should be between 1 and the minimum of data num and feature dimensions
INSERT INTO PAL_PARAMETER_TBL VALUES ('N_COMPONENTS', 2, NULL, NULL);
-- Determine the threshold to neglect component whose singular value is too
small. The valid value is [0, 1)
-- If this parameter is not specified, then all required components are output.
-- When the singular value of some component divide with largest singular value
is less than this threshold, the component will be dropped
INSERT INTO PAL_PARAMETER_TBL VALUES ('COMPONENT_TOL', NULL, 1E-5, NULL);
-- Specify the list of categorical variable names. It can be specified multiple
times, once for each categorical variable
-- Only Integer Variable needs to be specified. VARCHAR/NVARCHAR variable will
always be treated as categorical variable
INSERT INTO PAL_PARAMETER_TBL VALUES ('CATEGORICAL_VARIABLE', NULL, NULL, 'X4');
-- SEED is used to generate initial quantification for categorical variable;
Valid value is [0, Infinity).
-- Default value is 0, which means the initial quantification will be different
for each run; Otherwise the initial quantification will be deterministic
INSERT INTO PAL_PARAMETER_TBL VALUES ('SEED', 2021, NULL, NULL);
-- Used to compute quantification for categorical variable. The algorithm is an
iterative method. This parameter specify the maximum allowed iterations
-- Default value is 100. Valid value range is (0, Infinity)
INSERT INTO PAL_PARAMETER_TBL VALUES ('MAX_ITERATION', 550, NULL, NULL);
-- Used as threshold to determine when the iterative quantification process can
be stopped
-- When the improvement of loss value is less than this threshold, the iterative
process will terminate and regard as converged
-- Default value is 1E-5. Valid range is (0, 1)
INSERT INTO PAL_PARAMETER_TBL VALUES ('CONVERGE_TOL', NULL, 1E-5, NULL);
-- Only valid when SVD_CALCULATOR is 0. This parameter specify the maximum
allowed iteration when computing SVD using the LANCZOS algorithm
-- Default value is 1000. Valid range is (0, Infinity). Note if this value is
too small, algorithm will raised error as the computation result doesn't converge
INSERT INTO PAL_PARAMETER_TBL VALUES ('LANCZOS_ITERATION', 100, NULL, NULL);
-- Choose the SVD Algorithm to compute SVD: 0 means LANCZOS algorithm; 1 means
divide and conquer with Jacobi Algorithm
-- Note when extracting full components(minimum of data num and feature
dimension), algorithm will force as if this parameter is 1, since LANCZOS
algorithm can't cope with this case
INSERT INTO PAL_PARAMETER_TBL VALUES ('SVD_CALCULATOR', 0, NULL, NULL);
DROP TABLE PAL_PCA_LOADINGS_TBL;
-- Specify component loading for each variable and component; component is
represent as integer starting from 1 with decreasing order of singular values
CREATE COLUMN TABLE PAL_PCA_LOADINGS_TBL ("VARIABLE_NAME" NVARCHAR(100),
"COMPONENT_ID" INTEGER, "COMPONENT_LOADING" DOUBLE);
DROP TABLE PAL_PCA_LOADINGS_INFORMATION;
-- Specify some metrics for each component, currently supported are SD:
standard deviation; VAR_PROP: proportion of variance; CUM_VAR_PROP: proportion
of cumulative variance
```

```
CREATE COLUMN TABLE PAL_PCA_LOADINGS_INFORMATION ("COMPONENT_ID" INTEGER,
"METRIC_NAME" NVARCHAR(128), "METRIC_VALUE" DOUBLE);
DROP TABLE PAL_PCA_SCORES;
-- Specify component score for each data instance
CREATE COLUMN TABLE PAL_PCA_SCORES ("ID" INTEGER, "COMPONENT_ID" INTEGER,
"COMPONENT_SCORE" DOUBLE);
DROP TABLE PAL_PCA_SCALING_INFORMATION_TBL;
-- Specify the mean and scale info for each variable
CREATE COLUMN TABLE PAL_PCA_SCALING_INFORMATION_TBL ("VARIABLE_NAME"
NVARCHAR(100), "MEAN" DOUBLE, "SCALE" DOUBLE);
DROP TABLE PAL_PCA_QUANTIFICATION_TBL;
-- Specify the quantification info for each variable. For each category value
there is one quantification
-- Note current algorithm only support single nominal variable quantification,
which impose rank-one restriction, so COMPONENT_ID is set 0 to indicate this case
CREATE COLUMN TABLE PAL_PCA_QUANTIFICATION_TBL ("VARIABLE_NAME" NVARCHAR(100),
"CATEGORY_VALUE" NVARCHAR(1000), "COMPONENT_ID" INTEGER, "QUANTIFICATION"
DOUBLE);
DROP TABLE PAL_PCA_STAT_TBL;
-- Statistics info about current run result. Thhis is only meaningful when
categorical variable exists.
CREATE COLUMN TABLE PAL_PCA_STAT_TBL ("STAT_NAME" NVARCHAR(100), "STAT_VALUE"
NVARCHAR(1000));
CALL _SYS_AFL.PAL_CATPCA
(PAL_PCA_DATA_TBL,PAL_PARAMETER_TBL,PAL_PCA_LOADINGS_TBL,PAL_PCA_LOADINGS_INFORMA
TION,PAL_PCA_SCORES,PAL_PCA_SCALING_INFORMATION_TBL, PAL_PCA_QUANTIFICATION_TBL,
PAL_PCA_STAT_TBL) WITH OVERVIEW;
```

## Expected Result

### Loadings

| | VARIABLE_NAME | COMPONENT_ID | COMPONENT_LOADING |
|---|---|---|---|
| 1 | X1 | 1 | -0.4444619685563741 |
| 2 | X1 | 2 | -0.26654271070698154 |
| 3 | X3 | 1 | -0.3313307127301887 |
| 4 | X3 | 2 | 0.5321248854087529 |
| 5 | X5 | 1 | -0.46741091979170735 |
| 6 | X5 | 2 | -0.13200579260364523 |
| 7 | X6 | 1 | -0.46348994938251437 |
| 8 | X6 | 2 | -0.1582530181988565 |
| 9 | X2 | 1 | 0.21311242882105036 |
| 10 | X2 | 2 | -0.755338277038107 |
| 11 | X4 | 1 | -0.46255887070896795 |
| 12 | X4 | 2 | -0.181087354304041 |

### Loading Information

| | COMPONENT_ID | METRIC_NAME | METRIC_VALUE |
|---|---|---|---|
| 1 | 1 | SD | 2.097760451845607 |
| 2 | 1 | VAR_PROP | 0.7334331522212476 |
| 3 | 1 | CUM_VAR_PROP | 0.7334331522212476 |
| 4 | 2 | SD | 1.0918379582314002 |
| 5 | 2 | VAR_PROP | 0.19868502117248546 |
| 6 | 2 | CUM_VAR_PROP | 0.932118173393733 |

### Scores

| | ID | COMPONENT_ID | COMPONENT_SCORE |
|---|---|---|---|
| 1 | 1 | 1 | 2.734518253371139 |
| 2 | 2 | 1 | 1.0556373876562994 |
| 3 | 3 | 1 | 1.9187114824211804 |
| 4 | 4 | 1 | 1.7688406242829946 |
| 5 | 5 | 1 | 1.0199875124476907 |
| 6 | 6 | 1 | -2.3861262894490896 |
| 7 | 7 | 1 | -3.1862611058245927 |
| 8 | 8 | 1 | 1.29112266740513 |
| 9 | 9 | 1 | 0.1996064290931538 |
| 10 | 10 | 1 | -1.8474542465888582 |
| 11 | 11 | 1 | -2.5685827148150446 |
| 12 | 1 | 2 | -0.7476313063145041 |
| 13 | 2 | 2 | 1.709687165972214 |
| 14 | 3 | 2 | -0.06488889208489766 |
| 15 | 4 | 2 | -0.7674418196517394 |
| 16 | 5 | 2 | 0.5579476239394725 |
| 17 | 6 | 2 | -1.094885111358763 |
| 18 | 7 | 2 | 1.0711595872237887 |
| 19 | 8 | 2 | -1.014390339057052 |
| 20 | 9 | 2 | 1.7496627590037872 |
| 21 | 10 | 2 | -0.9438062618990525 |
| 22 | 11 | 2 | -0.4554134057732535 |

### Scaling Information

| | VARIABLE_NAME | MEAN | SCALE |
|---|---|---|---|
| 1 | X1 | 17 | 5.039841267341661 |
| 2 | X3 | 23 | 2 |
| 3 | X5 | 53.27272727272727 | 3.7707004413214555 |
| 4 | X6 | 19.90909090909091 | 4.5266885347800425 |
| 5 | X2 | -8.074349270001139e-17 | 0.8799963317181186 |
| 6 | X4 | -1.2111523905001707e-16 | 0.9806364730554222 |

### Category Quantification

| | VARIABLE_NAME | CATEGORY_VALUE | COMPONENT_ID | QUANTIFICATION |
|---|---|---|---|---|
| 1 | X2 | A | 0 | 0.866492224214148 |
| 2 | X2 | B | 0 | -0.6368883994607137 |
| 3 | X2 | C | 0 | 0.19066734092217208 |
| 4 | X2 | D | 0 | -1.3820970603508407 |
| 5 | X4 | 44 | 0 | -1.1297143287199223 |
| 6 | X4 | 45 | 0 | -0.6239679875964266 |
| 7 | X4 | 46 | 0 | -0.6260893000301522 |
| 8 | X4 | 53 | 0 | 1.0252292663814258 |
| 9 | X4 | 54 | 0 | 1.3018050147058182 |
| 10 | X4 | 55 | 0 | 1.2753652990393438 |

### Statistic

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | iteration | 22 |
| 2 | converge_status | Converged |
| 3 | objective_value | 0.6788182660626786 |

## _SYS_AFL.PAL_CATPCA_PROJECT

This procedure is used to transform given data to vector space spanned by principal components. Besides parameter table and data table, there are three input tables which form the model of Principal Component Analysis.

Loading Table contains the Component Loadings. Scale Information Table contain the mean and standard deviation per variable. Quantification Table contain the optimal scaling quantification for each category in each variable. These three tables are output by `_SYS_PAL.PAL_CATPCA` procedure and form the model of PCA.

### Overview

| Position | Table Name | Type |
|---|---|---|
| 1 | Data Table | In |
| 2 | Loadings Table | In |
| 3 | Scale Information Table | In |
| 4 | Quantification Table | In |
| 5 | Parameter Table | In |
| 6 | Score Table | Out |

### Input Table

**Table** *Data*

| Column | Data Type | Description |
|---|---|---|
| First Column | INTEGER, BIGINT, VARCHAR, NVARCHAR | Data ID |
| Variable Columns | INTEGER, DOUBLE, DECIMAL(p, s), VARCHAR, NVARCHAR | Data variables |

**Table** *Loadings*

| Column | Data Type | Column Name | Description |
|---|---|---|---|
| First Column | VARCHAR, NVARCHAR | VARIABLE_NAME | Variable Name |
| Second Column | INTEGER | COMPONENT_ID | Component IDs are marked from 1 ordered by decreasing variance |
| Third Column | DOUBLE | COMPONENT_LOADING | Component Loadings form the change of basis to transform data from original vector space to vector space spanned by principal components |

**Table** *Scaling Information*

| Column | Data Type | Column Name | Description |
|---|---|---|---|
| First Column | VARCHAR, NVARCHAR | VARIABLE_NAME | Variable Name |
| Second Column | DOUBLE | MEAN | Mean of Variable |
| Third Column | DOUBLE | SCALE | Standard Deviation of Variable |

**Table** *Category Quantification*

Current implementation only support single nominal variable, the COMPONENT_ID will always output 0 to represent this case.

| Column | Data Type | Column Name | Description |
|---|---|---|---|
| First Column | VARCHAR, NVARCHAR | VARIABLE_NAME | Variable Name |
| Second Column | VARCHAR, NVARCHAR | CATEGORY_VALUE | Discrete Value |
| Third Column | INTEGER | COMPONENT_ID | Component IDs are set 0 for single nominal variable quantification |
| Fourth Column | DOUBLE | QUANTIFICATION | Category Quantification |

### Parameter Table

Mandatory Parameters

None

Optional Parameters

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| THREAD_RATIO | DOUBLE | 1.0 | Specifies the ratio of available total thread number. The value range is [0,1]. 0 means 1 thread; 1 means all available threads. |
| MAX_COMPONENTS | INTEGER | Component Dimension of given loading | Specify the component dimension to keep in output. The value range should be between 1 and default value. |
| IGNORE_UNKNOWN_CATEGORY | INTEGER | 0 | Specify the behavior to cope with unknown category.<br>Valid Value:<br>• 0: Raise Error Message<br>• 1: Ignore the unknown category and quantify 0 to the category feature |

### Output Table

**Table** *Scores*

| Column | Data Type | Column Name | Description |
|---|---|---|---|
| First Column | Refer to the ID Column in Data Table | Data ID Column Name in Data Table | Data ID |
| Second Column | INTEGER | COMPONENT_ID | Component IDs are marked from 1 ordered by decreasing variance |
| Third Column | DOUBLE | COMPONENT_SCORE | Component Score |

### Example: Perform PCA Projection based on given data and model

The data in table `PAL_PCA_LOADINGS_TBL`, `PAL_PCA_SCALING_INFORMATION_TBL`, `PAL_PCA_QUANTIFICATION_TBL` is obtained in the first example for procedure `_SYS_PAL.PAL_CATPCA`.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_PCA_PROJ_DATA_TBL;
CREATE COLUMN TABLE PAL_PCA_PROJ_DATA_TBL ("ID" INTEGER, "X1" DOUBLE, "X2"
NVARCHAR(5), "X3" DOUBLE, "X4" INTEGER, "X5" DOUBLE, "X6" DOUBLE);
INSERT INTO PAL_PCA_PROJ_DATA_TBL VALUES (1, 12, 'A', 20, 44, 48, 16);
INSERT INTO PAL_PCA_PROJ_DATA_TBL VALUES (2, 12, 'B', 25, 45, 50, 16);
INSERT INTO PAL_PCA_PROJ_DATA_TBL VALUES (3, 12, 'C', 21, 45, 50, 16);
INSERT INTO PAL_PCA_PROJ_DATA_TBL VALUES (4, 13, 'A', 21, 46, 51, 17);
INSERT INTO PAL_PCA_PROJ_DATA_TBL VALUES (5, 14, 'C', 24, 46, 51, 17);
INSERT INTO PAL_PCA_PROJ_DATA_TBL VALUES (6, 22, 'A', 25, 54, 58, 26);
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL ("PARAM_NAME" VARCHAR(256), "INT_VALUE"
INTEGER, "DOUBLE_VALUE" DOUBLE, "STRING_VALUE" VARCHAR(1000));
-- Proportion of CPU resources to use in current computation. Value should be
between 0 and 1. Default is 1
INSERT INTO PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 0.5, NULL);
-- Specify number of components to output; Default value is the number of
components in the given model
INSERT INTO PAL_PARAMETER_TBL VALUES ('MAX_COMPONENTS', 2, NULL, NULL);
-- This parameter specify the behavior when encountering unknown category value
comparing to the given model.
-- Default value is 0. When it is 0, algorithm will raise error; Otherwise
algorithm will bypass the error and quantify 0 to the category feature
INSERT INTO PAL_PARAMETER_TBL VALUES ('IGNORE_UNKNOWN_CATEGORY', 0, NULL, NULL);
DROP TABLE PAL_PCA_PROJ_SCORES;
CREATE COLUMN TABLE PAL_PCA_PROJ_SCORES ("ID" INTEGER, "COMPONENT_ID" INTEGER,
"COMPONENT_SCORE" DOUBLE);
CALL _SYS_AFL.PAL_CATPCA_PROJECT
(PAL_PCA_PROJ_DATA_TBL,PAL_PCA_LOADINGS_TBL,PAL_PCA_SCALING_INFORMATION_TBL,
PAL_PCA_QUANTIFICATION_TBL, PAL_PARAMETER_TBL, PAL_PCA_PROJ_SCORES) WITH
OVERVIEW;
```

#### Expected Result

#### Scores

| | ID | COMPONENT_ID | COMPONENT_SCORE |
|---|---|---|---|
| 1 | 1 | 1 | 2.734518253371139 |
| 2 | 2 | 1 | 1.0556373876562994 |
| 3 | 3 | 1 | 1.9187114824211804 |
| 4 | 4 | 1 | 1.7688406242829946 |
| 5 | 5 | 1 | 1.0199875124476907 |
| 6 | 6 | 1 | -2.3861262894490896 |
| 7 | 1 | 2 | -0.7476313063145041 |
| 8 | 2 | 2 | 1.709687165972214 |
| 9 | 3 | 2 | -0.06488889208489766 |
| 10 | 4 | 2 | -0.7674418196517394 |
| 11 | 5 | 2 | 0.5579476239394725 |
| 12 | 6 | 2 | -1.094885111358763 |

## 5.8.2 Discretize

It is an enhanced version of binning function which can be applied to table with multiple columns.

This function partitions table rows into multiple segments called bins, then applies smoothing methods in each bin of each column respectively.

## Prerequisites

- The input data does not contain null value.

## _SYS_AFL.PAL_DISCRETIZE

This function preprocesses the data.

### Overview of all Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<RESULT table>` | OUT |
| 4 | `<ASSIGNMENT table>` | OUT |
| 5 | `<MODEL table>` | OUT |
| 6 | `<STATISTIC table>` | OUT |

### Input Table

| Table | Column | Reference | Data Type | Description |
|---|---|---|---|---|
| DATA | 1st column | ID | INTEGER, BIGINT, VARCHAR, or NVARCHAR | Data ID<br>It must be the first column. |
| | Other columns | FEATURES | INTEGER, DOUBLE, DECIMAL(p, s), VARCHAR, or NVARCHAR | Attribute data |

### Parameter Table

Mandatory Parameters

The following parameters are mandatory and must be given a value.

| Name | Data Type | Description | Dependency |
|---|---|---|---|
| BINNING_VARIABLE | VARCHAR or NVARCHAR | Attribute name, to which binning operation is applied. | Variable data type must be numeric. |

| Name | Data Type | Description | Dependency |
|---|---|---|---|
| METHOD | INTEGER | Binning methods:<br>• 0: equal widths based on the number of bins<br>• 1: equal widths based on the bin width<br>• 2: equal number of records per bin<br>• 3: mean/ standard deviation bin boundaries | |

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| BIN_NUMBER | INTEGER | 2 | Number of needed bins. | Only valid when METHOD is 0 or 2. |
| BIN_DISTANCE | DOUBLE | 10 | Specifies the distance for binning. | Only valid when METHOD is 1. |
| SD | INTEGER | 1 | Specifies the number of standard deviation at each side of the mean.<br>For example, if SD equals 2, this function takes mean +/- 2 * standard deviation as the upper/lower bound for binning. | Only valid when METHOD is 3. |
| DEFAULT_SMOOTHING_METHOD | INTEGER | 0 | Default smoothing methods:<br>• 0: no smoothing<br>• 1: smoothing by bin means<br>• 2: smoothing by bin medians<br>• 3: smoothing by bin boundaries | Only applies for none-categorical attributes that do not get specified smoothing method by parameter SMOOTHING_METHOD. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| SMOOTHING_METHOD | VARCHAR or NVARCHAR for column name and INTEGER for smoothing method code | No default value | Specifies column name and its method code for smoothing. The set of code is the same as DEFAULT_SMOOTHING_METHOD. | Only applies for non-categorical attributes. |
| CATEGORICAL_VARIABLE | VARCHAR, or NVARCHAR | No default value | Indicates whether a column data is actually corresponding to a category variable even the data type of this column is INTEGER.<br>By default, VARCHAR or NVARCHAR is category variable, and INTEGER or DOUBLE is continuous variable. | |
| SAVE_MODEL | INTEGER | 1 | Indicates whether the model is saved.<br>• 0: not save<br>• 1: save | |

### Output Tables

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| RESULT | 1st column | ID (in input table) data type | ID (in input table) column name | Data ID |
| | Other columns | FEATURES (in input table) data type with conversion from integer to double | FEATURES (in input table) column name | Data smoothed respectively in each bins |
| ASSIGNMENT | 1st column | ID (in input table) data type | ID (in input table) column name | Data ID |
| | 2nd column | INTEGER | BIN_INDEX | Bin index |
| MODEL | 1st column | INTEGER | ROW_INDEX | Model row index |
| | 2nd column | NVARCHAR(5000) | MODEL_CONTENT | Model contents |

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| STATISTICS | 1st column | NVARCHAR(256) | STAT_NAME | Statistics name |
| | 2nd column | NVARCHAR(1000) | STAT_VALUE | Statistics value |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_DISC_DATA_TBL;
CREATE COLUMN TABLE PAL_DISC_DATA_TBL (
    "ID" INTEGER,
    "ATT1" DOUBLE,
    "ATT2" INTEGER,
    "ATT3" INTEGER,
    "ATT4" NVARCHAR(10)
);
INSERT INTO PAL_DISC_DATA_TBL VALUES(1, 10.0, 100, 1, 'A');
INSERT INTO PAL_DISC_DATA_TBL VALUES(2, 10.1, 101, 1, 'A');
INSERT INTO PAL_DISC_DATA_TBL VALUES(3, 10.2, 100, 1, 'A');
INSERT INTO PAL_DISC_DATA_TBL VALUES(4, 10.4, 103, 1, 'A');
INSERT INTO PAL_DISC_DATA_TBL VALUES(5, 10.3, 100, 1, 'A');
INSERT INTO PAL_DISC_DATA_TBL VALUES(6, 40.0, 400, 4, 'C');
INSERT INTO PAL_DISC_DATA_TBL VALUES(7, 40.1, 402, 4, 'B');
INSERT INTO PAL_DISC_DATA_TBL VALUES(8, 40.2, 400, 4, 'B');
INSERT INTO PAL_DISC_DATA_TBL VALUES(9, 40.4, 402, 4, 'B');
INSERT INTO PAL_DISC_DATA_TBL VALUES(10, 40.3, 400, 4, 'A');
INSERT INTO PAL_DISC_DATA_TBL VALUES(11, 90.0, 900, 2, 'C');
INSERT INTO PAL_DISC_DATA_TBL VALUES(12, 90.1, 903, 1, 'B');
INSERT INTO PAL_DISC_DATA_TBL VALUES(13, 90.2, 901, 2, 'B');
INSERT INTO PAL_DISC_DATA_TBL VALUES(14, 90.4, 900, 1, 'B');
INSERT INTO PAL_DISC_DATA_TBL VALUES(15, 90.3, 900, 1, 'B');
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR(100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR(100)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES('BINNING_VARIABLE', NULL, NULL, 'ATT1');
INSERT INTO #PAL_PARAMETER_TBL VALUES('METHOD', 0, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES('BIN_NUMBER', 3, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES('DEFAULT_SMOOTHING_METHOD', 2, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES('SMOOTHING_METHOD', 1, NULL, 'ATT2');
INSERT INTO #PAL_PARAMETER_TBL VALUES('CATEGORICAL_VARIABLE', NULL, NULL,
'ATT3');
CALL _SYS_AFL.PAL_DISCRETIZE(PAL_DISC_DATA_TBL, #PAL_PARAMETER_TBL, ?, ?, ?, ?);
```

### Expected Results

RESULT:

| | ID | ATT1 | ATT2 | ATT3 | ATT4 |
|---|---|---|---|---|---|
| 1 | 1 | 10.2 | 100.8 | 1 | A |
| 2 | 2 | 10.2 | 100.8 | 1 | A |
| 3 | 3 | 10.2 | 100.8 | 1 | A |
| 4 | 4 | 10.2 | 100.8 | 1 | A |
| 5 | 5 | 10.2 | 100.8 | 1 | A |
| 6 | 6 | 40.2 | 400.8 | 4 | C |
| 7 | 7 | 40.2 | 400.8 | 4 | B |
| 8 | 8 | 40.2 | 400.8 | 4 | B |
| 9 | 9 | 40.2 | 400.8 | 4 | B |
| 10 | 10 | 40.2 | 400.8 | 4 | A |
| 11 | 11 | 90.2 | 900.8 | 2 | C |
| 12 | 12 | 90.2 | 900.8 | 1 | B |
| 13 | 13 | 90.2 | 900.8 | 2 | B |
| 14 | 14 | 90.2 | 900.8 | 1 | B |
| 15 | 15 | 90.2 | 900.8 | 1 | B |

ASSIGNMENT:

| | ID | BIN_INDEX |
|---|---|---|
| 1 | 1 | 1 |
| 2 | 2 | 1 |
| 3 | 3 | 1 |
| 4 | 4 | 1 |
| 5 | 5 | 1 |
| 6 | 6 | 2 |
| 7 | 7 | 2 |
| 8 | 8 | 2 |
| 9 | 9 | 2 |
| 10 | 10 | 2 |
| 11 | 11 | 3 |
| 12 | 12 | 3 |
| 13 | 13 | 3 |
| 14 | 14 | 3 |
| 15 | 15 | 3 |

MODEL:

| | ROW_INDEX | MODEL_CONTENT |
|---|---|---|
| 1 | 1 | {"Binning":{"Barriers":[10.0,36.80,63.60,90.40000000000001]... |

## 5.8.3 Discretize Apply

This function applies models generated from the DISCRETIZE function to new data.

It partitions new data according to bin boundaries contained in the model and performs smoothing in each bin of each column of the new data. The table structure of the new data should be the same as the one that generates this model. The new data does not update the binning model.

## Prerequisites

- Input table and model table are not empty and do not contain null values.
- Input table has the same structure as the one that generate the model.

## _SYS_AFL.PAL_DISCRETIZE_APPLY

This function directly assigns data to bins based on the model, and performs smoothing on them.

### Overview of all Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<MODEL table>` | IN |
| 3 | `<PARAMETER table>` | IN |
| 4 | `<RESULT table>` | OUT |
| 5 | `<ASSIGNMENT table>` | OUT |
| 6 | `<STATISTIC table>` | OUT |

### Input Table

| Table | Column | Reference | Data Type | Description |
|---|---|---|---|---|
| DATA | 1st column | ID | INTEGER, BIGINT, VARCHAR, or NVARCHAR | Data ID<br>It must be the first column. |
| | Other columns | FEATURES | INTEGER, DOUBLE, DECIMAL(p, s), VARCHAR, or NVARCHAR | Attribute data |
| MODEL | 1st column | | INTEGER | Model row index |
| | 2nd column | | NVARCHAR | Model content |

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

None.

### Output Tables

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| RESULT | 1st column | ID (in input table) data type | ID (in input table) column name | Data ID |
| | Other columns | FEATURES (in input table) data type with conversion from integer to double | FEATURES (in input table) column name | Data smoothed respectively in each bins |
| ASSIGNMENT | 1st column | ID (in input table) data type | ID (in input table) column name | Data ID |
| | 2nd column | INTEGER | BIN_INDEX | Bin index |
| STATISTICS | 1st column | NVARCHAR(256) | STAT_NAME | Statistics name |
| | 2nd column | NVARCHAR(1000) | STAT_VALUE | Statistics value |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_DISC_DATA_TBL;
CREATE COLUMN TABLE PAL_DISC_DATA_TBL (
    "ID" INTEGER,
    "ATT1" DOUBLE,
    "ATT2" INTEGER,
    "ATT3" INTEGER,
    "ATT4" NVARCHAR(10)
);
INSERT INTO PAL_DISC_DATA_TBL VALUES(1, 10.0, 100, 1, 'A');
INSERT INTO PAL_DISC_DATA_TBL VALUES(2, 10.1, 101, 1, 'A');
INSERT INTO PAL_DISC_DATA_TBL VALUES(3, 10.2, 100, 1, 'A');
INSERT INTO PAL_DISC_DATA_TBL VALUES(4, 10.4, 103, 1, 'A');
INSERT INTO PAL_DISC_DATA_TBL VALUES(5, 10.3, 100, 1, 'A');
INSERT INTO PAL_DISC_DATA_TBL VALUES(6, 40.0, 400, 4, 'C');
INSERT INTO PAL_DISC_DATA_TBL VALUES(7, 40.1, 402, 4, 'B');
INSERT INTO PAL_DISC_DATA_TBL VALUES(8, 40.2, 400, 4, 'B');
INSERT INTO PAL_DISC_DATA_TBL VALUES(9, 40.4, 402, 4, 'B');
INSERT INTO PAL_DISC_DATA_TBL VALUES(10, 40.3, 400, 4, 'A');
INSERT INTO PAL_DISC_DATA_TBL VALUES(11, 90.0, 900, 2, 'C');
INSERT INTO PAL_DISC_DATA_TBL VALUES(12, 90.1, 903, 1, 'B');
INSERT INTO PAL_DISC_DATA_TBL VALUES(13, 90.2, 901, 2, 'B');
INSERT INTO PAL_DISC_DATA_TBL VALUES(14, 90.4, 900, 1, 'B');
INSERT INTO PAL_DISC_DATA_TBL VALUES(15, 90.3, 900, 1, 'B');
DROP TABLE PAL_DISC_RESULT_TBL;
CREATE COLUMN TABLE PAL_DISC_RESULT_TBL (
    "ID" INTEGER,
    "ATT1" DOUBLE,
    "ATT2" DOUBLE,
    "ATT3" DOUBLE,
    "ATT4" NVARCHAR(10)
```

```
);
DROP TABLE PAL_DISC_ASSIGN_TBL;
CREATE COLUMN TABLE PAL_DISC_ASSIGN_TBL (
    "ID" INTEGER,
    "BIN_INDX" DOUBLE
);
DROP TABLE PAL_DISC_MODEL_TBL;
CREATE COLUMN TABLE PAL_DISC_MODEL_TBL (
    "ROW_INDEX" INT,
    "MODEL_CONTENT" VARCHAR(5000)
);
DROP TABLE PAL_DISC_STAT_TBL;
CREATE COLUMN TABLE PAL_DISC_STAT_TBL (
    "STAT_NAME" VARCHAR(256),
    "STAT_VALUE" VARCHAR(1000)
);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR(100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR(100)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES('BINNING_VARIABLE', NULL, NULL, 'ATT1');
INSERT INTO #PAL_PARAMETER_TBL VALUES('METHOD', 0, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES('BIN_NUMBER', 3, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES('DEFAULT_SMOOTHING_METHOD', 2, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES('SMOOTHING_METHOD', 1, NULL, 'ATT2');
INSERT INTO #PAL_PARAMETER_TBL VALUES('CATEGORICAL_VARIABLE', NULL, NULL,
'ATT3');
CALL _SYS_AFL.PAL_DISCRETIZE(PAL_DISC_DATA_TBL, #PAL_PARAMETER_TBL,
PAL_DISC_RESULT_TBL, PAL_DISC_ASSIGN_TBL, PAL_DISC_MODEL_TBL, PAL_DISC_STAT_TBL)
WITH OVERVIEW;
------ APPLY ------
DROP TABLE PAL_DISC_APPLY_TBL;
CREATE COLUMN TABLE PAL_DISC_APPLY_TBL (
    "ID" INTEGER,
    "ATT1" DOUBLE,
    "ATT2" INTEGER,
    "ATT3" INTEGER,
    "ATT4" NVARCHAR(10)
);
INSERT INTO PAL_DISC_APPLY_TBL VALUES(1, 20.0, 120, 1, 'A');
INSERT INTO PAL_DISC_APPLY_TBL VALUES(2, 30.0, 110, 2, 'A');
INSERT INTO PAL_DISC_APPLY_TBL VALUES(3, 10.7, 143, 1, 'B');
INSERT INTO PAL_DISC_APPLY_TBL VALUES(4, 15.8, 256, 2, 'C');
INSERT INTO PAL_DISC_APPLY_TBL VALUES(5, 88.9, 100, 1, 'A');
INSERT INTO PAL_DISC_APPLY_TBL VALUES(6, 76.5, 402, 2, 'A');
INSERT INTO PAL_DISC_APPLY_TBL VALUES(7, 55.3, 905, 4, 'B');
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR(100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR(100)
);
CALL _SYS_AFL.PAL_DISCRETIZE_APPLY(PAL_DISC_APPLY_TBL, PAL_DISC_MODEL_TBL,
#PAL_PARAMETER_TBL, ?, ?, ?);
```

### Expected Results

RESULT:

| | ID | ATT1 | ATT2 | ATT3 | ATT4 |
|---|---|---|---|---|---|
| 1 | 1 | 10.2 | 100.8 | 1 | A |
| 2 | 2 | 10.2 | 100.8 | 2 | A |
| 3 | 3 | 10.2 | 100.8 | 1 | B |
| 4 | 4 | 10.2 | 100.8 | 2 | C |
| 5 | 5 | 90.2 | 900.8 | 1 | A |
| 6 | 6 | 90.2 | 900.8 | 2 | A |
| 7 | 7 | 40.2 | 400.8 | 4 | B |

ASSIGNMENT:

| | ID | BIN_INDEX |
|---|---|---|
| 1 | 1 | 1 |
| 2 | 2 | 1 |
| 3 | 3 | 1 |
| 4 | 4 | 1 |
| 5 | 5 | 3 |
| 6 | 6 | 3 |
| 7 | 7 | 2 |

## 5.8.4 Feature Selection

Feature selection (FS) is a dimensionality reduction technique, which selects a subset of relevant features for model construction, thus reducing the memory storage and improving computational efficiency while avoiding significant loss of information.

The covered FS algorithms are listed as follows,

- Statistic based FS methods
  - 0: Anova
  - 1: Chi-squared
  - 2: Gini Index
  - 3: Fisher Score
- Information theory based FS methods
  - 4: Information Gain

    - 5: Minimum Redundancy Maximum Relevance (MRMR)
    - 6: Joint Mutual Information (JMI)
    - 7: Interaction Weight Based Feature Selection (IWFS)
    - 8: Fast Correlation Based Filter (FCBF)
- Similarity based FS methods
    - 9: Laplacian Score
    - 10: Spectral Feature Selection (SPEC)
    - 11: ReliefF
- Sparse Learning Based FS method
    - 12: ADMM
- Wrapper method
    - 13: Competitive Swarm Optimizer (CSO)

According to the calculation efficiency, capability of handling supervision and feature redundancy, these methods could be broadly classified in the following way,

| Method | Supervised/Unsupervised | Tackling Feature Redundancy (No/Yes) | Efficiency (Low/Medium/High) |
|---|---|---|---|
| 0 | Supervised | No | High |
| 1 | Supervised | No | High |
| 2 | Supervised | No | High |
| 3 | Supervised | No | High |
| 4 | Supervised | No | High |
| 5 | Supervised | Yes | Medium |
| 6 | Supervised | Yes | Medium |
| 7 | Supervised | Yes | Medium |
| 8 | Supervised | Yes | Medium |
| 9 | Unsupervised | No | Medium |
| 10 | Supervised/Unsupervised | No | Medium |
| 11 | Supervised | No | Medium |
| 12 | Supervised | Yes | Low |
| 13 | Supervised | Yes | Low |

## Prerequisites

- No null data in the inputs.

# _SYS_AFL.PAL_FEATURE_SELECTION

This function selects the best subset of features for modeling.

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | <DATA table> | IN |
| 2 | <PARAMETER table> | IN |
| 3 | <RESULT table> | OUT |

### Input Table

| Table | Column | Data Type | Description | Dependency |
|---|---|---|---|---|
| DATA | Multiple columns | INTEGER, DOUBLE, DECIMAL, VARCHAR or NVARCHAR | ID, feature columns or target column | |

### Parameter Table

Mandatory Parameters

The following parameters are mandatory and have no default values.

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| FS_METHOD | INTEGER | No default value | Feature selection method | Can only choose between 0 to 13 |
| TOP_K_BEST | INTEGER | No default value | Top k features to be selected | Must be assigned a value except for FCBF and CSO |

Optional Parameters

The following parameters are optional and for all the methods. If a parameter is not specified, its default value will be used.

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| HAS_ID | INTEGER | 0 | Indicates whether the first column is an ID column,<br>• 0: no<br>• 1: yes | If `yes`, the first column will be excluded from the feature candidates. |
| DEPENDENT_VARIABLE | VARCHAR or NVARCHAR | No default value | Specifies a dependent variable. | It performs unsupervised feature selection if no value is specified. |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| CATEGORICAL_VARIABLE | VARCHAR or NVARCHAR | No default value | Indicates categorical variables. The inner logic is,<br>• string: category<br>• integer or double: numeral | Only valid for integer or string column. |
| FIXED_FEATURE | VARCHAR or NVARCHAR | No default value | Will always be selected out as the best subset. | |
| EXCLUDED_FEATURE | VARCHAR or NVARCHAR | No default value | Excludes the indicated columns as feature candidates. | |
| VERBOSE | INTEGER | 0 | Indicates whether to output more specified results,<br>• 0: no<br>• 1: yes | |
| THREAD_RATIO | DOUBLE | -1 | The ratio of available threads.<br>• 0: single thread<br>• 0~1: percentage<br>• others: heuristically determined | |
| SEED | INTEGER | 0 | Random seed. 0 means using system time as seed. | |
| FS_ROWSAMPLING_RATIO | DOUBLE | • FS_METHOD 11: 0.6<br>• FS_METHOD 12, 13: 1.0 | The ratio of random sampling without replacement. | Valid only for method 11, 12 and 13. |

The following parameters are valid just for specified methods.

- FCBF (8)

In FCBF, we choose symmetrical uncertainty(SU) as a measure of correlation between two random variables. Features whose SU value is less than the predefined FS_THRESHOLD will be ignored in the calculation.

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| FS_THRESHOLD | DOUBLE | 0.01 | Predefined threshold for SU values between features and target. | |

- Similarity based FS methods (9, 10 and 11)

In these methods, one way to construct the affinity matrix is to use RBF kernel,

$$S(i,j)=e^{-\frac{\|x_i-x_j\|_2^2}{\sigma}}$$

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| FS_N_NEIGHBOURS | INTEGER | 5 | Number of neighbors considered in the computation of affinity matrix | |
| FS_CATEGORY_WEIGHT | DOUBLE | $[0.5 * [X_{num}.std()]_{avg}$, the average of the standard derivation of numerical columns scaled by 0.5 | The weight of categorical features whilst calculating distance. | |
| FS_SIGMA | DOUBLE | 1.0 | σ in above equation. | |

- SPEC(10)

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| FS_REGULARIZATION_POWER | INTEGER | 0 | The order of the power function that penalizes high frequency components. | |

- ADMM(12)

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| FS_MAX_ITER | INTEGER | 100 | Maximal iterations allowed to run optimization. | |
| FS_ADMM_TOL | DOUBLE | 0.0001 | Convergence threshold. | |
| FS_ADMM_RHO | DOUBLE | 1.0 | Penalty factor. | |
| FS_ADMM_MU | DOUBLE | 1.05 | Gain of FS_ADMM_RHO at each iteration. | |
| FS_ADMM_GAMMA | DOUBLE | 1.0 | Regularization coefficient. | |

- CSO(13)

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| CSO_REPEAT_NUM | INTEGER | 2 | Number of repetitions to run CSO. CSO starts with a different initialization at each time. | |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| CSO_MAXGENERATION_NUM | INTEGER | 100 | Maximal number of generations. | |
| CSO_EARLYSTOP_NUM | INTEGER | 30 | Stop if there's no change in `CSO_EARLYSTOP_NUM` generation. | |
| CSO_POPULATION_SIZE | INTEGER | 30 | Population size of the swarm particles. | |
| CSO_PHI | DOUBLE | 0.1 | Social factor. | |
| CSO_FEATURENUM_PENALTY | DOUBLE | 0.1 | Penalize the number of features. | |
| CSO_TEST_RATIO | DOUBLE | 0.2 | The ratio for the splitting of training data and testing data. | |

### Output Table

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| RESULT | 1st Column | INTEGER | ROWID | Indicates the id of current row. |
| | 2nd Column | NCLOB | OUTPUT | Best set of features |

### Examples

Assume that:

- DM_PAL has been assigned the AFL__SYS_AFL_AFLPAL_EXECUTE or AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION role.

### Example

```
SET SCHEMA DM_PAL;
DROP TYPE PAL_FS_DATA_TT;
CREATE TYPE PAL_FS_DATA_TT AS TABLE (
            "X1" INTEGER,
            "X2" DOUBLE,
            "X3" DOUBLE,
            "X4" DOUBLE,
            "X5" DOUBLE,
            "X6" DOUBLE,
            "X7" DOUBLE,
            "X8" DOUBLE,
            "X9" DOUBLE,
            "X10" DOUBLE,
            "X11" DOUBLE,
            "X12" DOUBLE,
            "X13" DOUBLE,
            "Y" DOUBLE
);
DROP TABLE PAL_FS_DATA_TBL;
CREATE COLUMN TABLE PAL_FS_DATA_TBL like  PAL_FS_DATA_TT;
DROP PROCEDURE FILL_DATA;
CREATE PROCEDURE FILL_DATA AS
BEGIN
```

```
    DECLARE x1,x2,x3,x4,x5,x6,x7,x8,x9,x10,x11,x12,x13,y DOUBLE ARRAY;
    DECLARE len INTEGER = 30;
    declare i integer;
    DECLARE data PAL_FS_DATA_TT;
    x1 = ARRAY(1,0,0,0,1,0,1,0,1,0,1,1,1,1,1,1,1,0,1,0,0,0,1,1,1,1,1,1,0,1);
    x2 =
ARRAY(22.08,22.67,29.58,21.67,20.17,15.83,17.42,58.67,27.83,55.75,33.5,41.42,20.6
7,34.92,58.58,48.08,29.58,18.92,20.0,22.42,28.17,19.17,41.17,41.58,19.5,32.75,22.
5,33.17,30.67,23.08);
    x3 =
ARRAY(11.46,7.0,1.75,11.5,8.17,0.585,6.5,4.46,1.0,7.08,1.75,5.0,1.25,5.0,2.71,6.0
4,4.5,9.0,1.25,5.665,0.585,0.585,1.335,1.75,9.585,1.5,0.125,3.04,12.0,2.5);
    x4 = ARRAY(2,2,1,1,2,2,2,2,1,2,2,2,1,2,2,2,2,2,1,2,2,1,2,2,2,2,1,1,2,2);
    x5 =
ARRAY(4,8,4,5,6,8,3,11,2,4,14,11,8,14,8,4,9,6,4,11,6,6,2,4,6,13,4,8,8,8);
    x6 = ARRAY(4,4,4,3,4,8,4,8,8,8,8,8,8,8,4,4,4,4,4,4,4,4,4,4,4,8,4,8,4,4);
    x7 =
ARRAY(1.585,0.165,1.25,0.0,1.96,1.5,0.125,3.04,3.0,6.75,4.5,5.0,1.375,7.5,2.415,0
.04,7.5,0.75,0.125,2.585,0.04,0.585,0.165,0.21,0.79,5.5,0.125,2.04,2.0,1.085);
    x8 = ARRAY(0,0,0,1,1,1,0,1,0,1,1,1,1,1,0,0,1,1,0,1,0,1,0,1,0,1,1,1,1);
    x9 = ARRAY(0,0,0,1,1,1,0,1,0,1,1,1,1,1,0,0,1,1,0,1,0,0,0,0,0,1,0,1,1,1);
    x10 = ARRAY(0,0,0,11,14,2,0,6,0,3,4,6,3,6,0,0,2,2,0,7,0,0,0,0,0,3,0,1,1,11);
    x11 = ARRAY(1,0,1,1,0,0,0,0,0,1,1,1,1,1,1,0,1,0,0,0,0,1,0,0,0,1,0,1,0,1);
    x12 = ARRAY(2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2);
    x13 =
ARRAY(100,160,280,0,60,100,60,43,176,100,253,470,140,0,320,0,330,88,140,129,260,1
60,168,160,80,0,200,180,220,60);
    y =
ARRAY(1213,1,1,1,159,1,101,561,538,51,858,1,211,1001,1,2691,1,592,5,3258,1005,1,1
,1,351,1,71,18028,20,2185);
    for i in 1..len do
        data =
            SELECT * FROM :data
            UNION ALL
            select :x1[i] as value, :x2[i] as value, :x3[i] as value, :x4[i]
as value, :x5[i] as value, :x6[i] as value, :x7[i] as value, :x8[i] as
value, :x9[i] as value, :x10[i] as value, :x11[i] as value, :x12[i] as
value, :x13[i] as value, :y[i] as value from dummy;
    end for;
    TRUNCATE TABLE PAL_FS_DATA_TBL;
    INSERT INTO PAL_FS_DATA_TBL SELECT * FROM :data;
END;
CALL FILL_DATA;
--select * from PAL_FS_DATA_TBL;
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR (1000)
);
--INSERT INTO #PAL_PARAMETER_TBL VALUES ('HAS_ID', 0, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FS_METHOD', 3, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('TOP_K_BEST', 8, NULL, NULL);
--INSERT INTO #PAL_PARAMETER_TBL VALUES ('VERBOSE', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('DEPENDENT_VARIABLE', NULL, NULL, 'Y');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('CATEGORICAL_VARIABLE', NULL, NULL, 'X1');
CALL _SYS_AFL.PAL_FEATURE_SELECTION (PAL_FS_DATA_TBL, "#PAL_PARAMETER_TBL", ?);
```

### Expected Result

RESULT:

| | ROWID | OUTPUT |
|---|---|---|
| 1 | 0 | {"__method__":"fisher-score","__SelectedFeatures__":["X3","X7","X2","X8","X9","X13","X6","X5"]} |

## 5.8.5 Isolation Forest

This function generates the anomaly score of each sample using the IsolationForest algorithm.

The algorithm takes advantage of two quantitative properties of anomalies:

- they are the minority consisting of few instances.
- they have attribute-values that are very different from those of normal instances. In other words, anomalies are 'few and different', which make them more susceptible to a mechanism called Isolation.

We use a binary tree structure called isolation tree (iTree) to recursively separate instances. Observations are isolated by randomly selecting a feature and then randomly selecting a split value between the maximum and minimum values of the selected feature. The number of splittings required to isolate a sample is equivalent to the path length from the root node to the terminating node. Because of the susceptibility to isolation, anomalies are more likely to be isolated closer to the root of an iTree; whereas normal points are more likely to be isolated at the deeper end of an iTree. Isolation Forest (iForest) builds an ensemble of iTrees for a given data set; anomalies are those instances that have short average path lengths on the iTrees.

Using the anomaly score s, we are able to make the following assessment:

- if instances return s very close to 1, then they are definitely anomalies.
- if instances have s much smaller than 0.5, then they are quite safe to be regarded as normal instances.
- if all the instances return s ≈ 0.5, then the entire sample does not really have any distinct anomaly.

In practice, we assume the proportions of outliers in the training and scoring data sets are the same. Suppose the amount of contamination is 0.25, first we score the training data and compute the 75% quantile of scores. Then we score the scoring data. If the score of a sample is greater than that pre-calculated quantile, the sample can be regarded as an outliner.

### Prerequisites

- No missing or null data in the inputs.
- The number of training samples should be at least two.
- The table used to store the tree model is a column table.
- The scoring data (predict) must have an ID column.
- The scoring data should be the same (name, order, data type) as those used in model training.

### _SYS_AFL.PAL_ISOLATION_FOREST

#### Overview of all tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |

| Position | Table Name | Parameter Type |
|---|---|---|
| 2 | `<PARAMETER table>` | IN |
| 3 | `<MODEL table>` | OUT |

### Input Table

| Table | Column | Data Type | Description |
|---|---|---|---|
| DATA | Columns | INTEGER, DOUBLE, or DECIMAL (p, s) | The input data. |

### Parameter Table

Mandatory Parameters

None.

Optional parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| THREAD_RATIO | DOUBLE | -1 | The ratio of available threads.<br>• 0: single thread<br>• 0~1: percentage<br>• Others: heuristically determined | |
| SEED | INTEGER | 0 | Specifies the seed for random number generator.<br>• 0: Uses the current time (in second) as seed<br>• Others: Uses the specified value as seed | |
| N_ESTIMATORS | INTEGER | 100 | Specifies the number of trees to grow. | |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| MAX_SAMPLES | INTEGER | 256 | Specifies the number of samples to draw from input to train each tree. If MAX_SAMPLES is larger than the number of samples provided, all samples will be used for all trees. | |
| MAX_FEATURES | INTEGER | 0 | Specifies the number of features to draw from input to train each tree. 0 means no sampling. | |
| BOOTSTRAP | INTEGER | 0 | Specifies sampling method.<br>• 0: Sampling without replacement<br>• 1: Sampling with replacement | |
| HAS_ID | INTEGER | 0 | Specifies if the first column of DATA table is used as ID column.<br>• 0: No<br>• 1: Yes | |

### Output Tables

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| MODEL | 1st Column | INTEGER | TREE_INDEX | Indicates the tree number. |
| | 2nd Column | NCLOB | MODEL_CONTENT | Model content. |

### Example

Assume that:

- DM_PAL is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_IF_DATA_TBL;
CREATE COLUMN TABLE PAL_IF_DATA_TBL (
    "X1" DOUBLE,
```

```
    "X2" DOUBLE
);
INSERT INTO PAL_IF_DATA_TBL VALUES (-2, -1);
INSERT INTO PAL_IF_DATA_TBL VALUES (-1, -1);
INSERT INTO PAL_IF_DATA_TBL VALUES (-1, -2);
INSERT INTO PAL_IF_DATA_TBL VALUES (1, 1);
INSERT INTO PAL_IF_DATA_TBL VALUES (1, 2);
INSERT INTO PAL_IF_DATA_TBL VALUES (2, 1);
INSERT INTO PAL_IF_DATA_TBL VALUES (6, 3);
INSERT INTO PAL_IF_DATA_TBL VALUES (-4, 7);
DROP TABLE PAL_IF_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_IF_PARAMETER_TBL (
        "PARAM_NAME" VARCHAR(256),
        "INT_VALUE" INTEGER,
        "DOUBLE_VALUE" DOUBLE,
        "STRING_VALUE" VARCHAR(100)
);
INSERT INTO PAL_IF_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 0, NULL);
INSERT INTO PAL_IF_PARAMETER_TBL VALUES ('SEED', 2, NULL, NULL);
DROP TABLE PAL_IF_MODEL_TBL;
CREATE COLUMN TABLE PAL_IF_MODEL_TBL ("TREE_INDEX" INTEGER, "MODEL_CONTENT"
NCLOB);
DO BEGIN
  lt_data = SELECT * FROM PAL_IF_DATA_TBL;
  lt_param = SELECT * FROM PAL_IF_PARAMETER_TBL;
  CALL _SYS_AFL.PAL_ISOLATION_FOREST (:lt_data, :lt_param, lt_model);
  INSERT INTO PAL_IF_MODEL_TBL SELECT * FROM :lt_model;
END;
SELECT * FROM PAL_IF_MODEL_TBL;
```

### Expected Result

| | TREE_INDEX | MODEL_CONTENT |
|---|---|---|
| 1 | 0 | {"NS":8,"NF":2,"FX":[0,1],"1":{"SF":0,"SV":5.525540351867676,"SS":0},"2":{ |
| 2 | 1 | {"NS":8,"NF":2,"FX":[0,1],"1":{"SF":0,"SV":5.525540351867676,"SS":0},"2":{ |
| 3 | 2 | {"NS":8,"NF":2,"FX":[1,0],"1":{"SF":0,"SV":4.684035301208496,"SS":0},"2":{ |
| 4 | 3 | {"NS":8,"NF":2,"FX":[1,0],"1":{"SF":0,"SV":4.684035301208496,"SS":0},"2":{ |
| 5 | 4 | {"NS":8,"NF":2,"FX":[1,0],"1":{"SF":0,"SV":5.334948539733887,"SS":0},"2":{ |
| 6 | 5 | {"NS":8,"NF":2,"FX":[1,0],"1":{"SF":0,"SV":5.334948539733887,"SS":0},"2":{ |
| 7 | 6 | {"NS":8,"NF":2,"FX":[0,1],"1":{"SF":0,"SV":3.6340508460998537,"SS":0},"2" |
| 8 | 7 | {"NS":8,"NF":2,"FX":[0,1],"1":{"SF":0,"SV":3.6340508460998537,"SS":0},"2" |
| 9 | 8 | {"NS":8,"NF":2,"FX":[1,0],"1":{"SF":0,"SV":-0.6070070266723633,"SS":0},"2 |
| 10 | 9 | {"NS":8,"NF":2,"FX":[1,0],"1":{"SF":0,"SV":-0.6070070266723633,"SS":0},"2 |
| 11 | 10 | {"NS":8,"NF":2,"FX":[0,1],"1":{"SF":1,"SV":5.90471076965332,"SS":0},"2":{" |
| 12 | 11 | {"NS":8,"NF":2,"FX":[0,1],"1":{"SF":1,"SV":5.90471076965332,"SS":0},"2":{" |
| 13 | 12 | {"NS":8,"NF":2,"FX":[1,0],"1":{"SF":1,"SV":5.519649028778076,"SS":0},"2":{ |
| 14 | 13 | {"NS":8,"NF":2,"FX":[1,0],"1":{"SF":1,"SV":5.519649028778076,"SS":0},"2":{ |
| 15 | 14 | {"NS":8,"NF":2,"FX":[0,1],"1":{"SF":0,"SV":-0.4324929714202881,"SS":0},"2 |
| 16 | 15 | {"NS":8,"NF":2,"FX":[0,1],"1":{"SF":0,"SV":-0.4324929714202881,"SS":0},"2 |

## _SYS_AFL.PAL_ISOLATION_FOREST_PREDICT

### Overview of all tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | <DATA table> | IN |
| 2 | <MODEL table> | IN |
| 3 | <PARAMETER table> | IN |
| 4 | <RESULT table> | OUT |

### Input Table

| Table | Column | Reference for Output Table | Data Type | Description | Constraint |
|---|---|---|---|---|---|
| DATA | 1st Column | ID | INTEGER, BIGINT, VARCHAR, or NVARCHAR | Data ID | This must be the first column. |
| | Other Columns | | INTEGER, DOUBLE, or DECIMAL(p, s) | The input features. | |
| MODEL | 1st Column | | INTEGER | Tree index. | |
| | 2nd Column | | NCLOB | Model content. | |

### Parameter Table

Mandatory Parameters

None.

Optional parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| THREAD_RATIO | DOUBLE | -1 | The ratio of available threads.<br>• 0: single thread<br>• 0~1: percentage<br>• Others: heuristically determined | |
| CONTAMINATION | DOUBLE | 0.1 | The proportion of outliers in the data set. CONTAMINATION should be in the range (0, 0.5]. | |

### Output table

| Table | Column | Data Type | Column Name | Description | Constraint |
|---|---|---|---|---|---|
| RESULT | 1st Column | ID column type | ID | ID column name. | Must be the first column. |
| | 2nd Column | DOUBLE | SCORE | Scoring result. | |
| | 3rd Column | INTEGER | LABEL | -1 for outliers and 1 for inliers. | |

### Example

Assume that:

- DM_PAL is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_IF_SCORING_DATA_TBL;
CREATE COLUMN TABLE PAL_IF_SCORING_DATA_TBL (
        "ID" INTEGER,
       "X1" DOUBLE,
       "X2" DOUBLE
);
INSERT INTO PAL_IF_SCORING_DATA_TBL VALUES (0, -2, -1);
INSERT INTO PAL_IF_SCORING_DATA_TBL VALUES (1, -1, -1);
INSERT INTO PAL_IF_SCORING_DATA_TBL VALUES (2, -1, -2);
INSERT INTO PAL_IF_SCORING_DATA_TBL VALUES (3, 1, 1);
INSERT INTO PAL_IF_SCORING_DATA_TBL VALUES (4, 1, 2);
INSERT INTO PAL_IF_SCORING_DATA_TBL VALUES (5, 2, 1);
INSERT INTO PAL_IF_SCORING_DATA_TBL VALUES (6, 6, 3);
INSERT INTO PAL_IF_SCORING_DATA_TBL VALUES (7, -4, 7);
DROP TABLE PAL_IF_SCORING_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_IF_SCORING_PARAMETER_TBL (
        "PARAM_NAME" VARCHAR(256),
        "INT_VALUE" INTEGER,
        "DOUBLE_VALUE" DOUBLE,
        "STRING_VALUE" VARCHAR(100)
);
INSERT INTO PAL_IF_SCORING_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 0, NULL);
INSERT INTO PAL_IF_SCORING_PARAMETER_TBL VALUES ('CONTAMINATION', NULL, 0.25,
NULL);
DROP TABLE PAL_IF_RESULT_TBL;
CREATE COLUMN TABLE PAL_IF_RESULT_TBL ("ID" INTEGER, "SCORE" DOUBLE, "LABEL"
INTEGER);
DO BEGIN
  lt_data = SELECT * FROM PAL_IF_SCORING_DATA_TBL;
  lt_model = SELECT * FROM PAL_IF_MODEL_TBL;
  lt_param = SELECT * FROM PAL_IF_SCORING_PARAMETER_TBL;
  CALL _SYS_AFL.PAL_ISOLATION_FOREST_PREDICT (:lt_data, :lt_model, :lt_param,
lt_result);
  INSERT INTO PAL_IF_RESULT_TBL SELECT * FROM :lt_result;
END;
SELECT * FROM PAL_IF_RESULT_TBL;
```

### Expected Result

| | ID | SCORE | LABEL |
|---|---|---|---|
| 1 | 0 | 0.44689698428888736 | 1 |
| 2 | 1 | 0.411048431708785 | 1 |
| 3 | 2 | 0.4989311315568398 | 1 |
| 4 | 3 | 0.40779644687145056 | 1 |
| 5 | 4 | 0.42326365421687284 | 1 |
| 6 | 5 | 0.4432695217396772 | 1 |
| 7 | 6 | 0.6195125059059307 | -1 |
| 8 | 7 | 0.6388737067483532 | -1 |

## 5.8.6 Inter-Quartile Range

Given a series of numeric data, the inter-quartile range (IQR) is the difference between the third quartile (Q3) and the first quartile (Q1) of the data.

```
IQR = Q3 – Q1
```

Q1 is equal to 25th percentile and Q3 is equal to 75th percentile.

The p-th percentile of a numeric vector is a number, which is greater than or equal to p% of all the values of this numeric vector.

IQR Test is a method to test the outliers of a series of numeric data. The algorithm performs the following tasks:

1. Calculates Q1, Q3, and IQR.
2. Set upper and lower bounds as follows:
   `Upper-bound = Q3 + 1.5 × IQR`
   `Lower-bound = Q1 – 1.5 × IQR`
3. Tests all the values of a numeric vector to determine if it is in the range. The value outside the range is marked as an outlier, meaning it does not pass the IQR test.

### Prerequisites

The input data does not contain null value.

### _SYS_AFL.PAL_IQR

This function performs the inter-quartile range test and outputs the test results.

**Overview of all Tables**

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<RESULT table>` | OUT |
| 4 | `<STATISTICS table>` | OUT |

### Input Table

| Table | Column | Reference for Output Table | Data Type | Description |
|---|---|---|---|---|
| DATA | 1st column | DATA_ID | INTEGER, VARCHAR, or NVARCHAR | ID |
| | 2nd column | TEST_DATA | INTEGER, DOUBLE, or DECIMAL(p, s) | Data that needs to be tested |

### Parameter Table

Mandatory Parameters

None.

Optional Parameter

The following parameter is optional. If it is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| MULTIPLIER | DOUBLE | 1.5 | The multiplier used in the IQR test. |

### Output Tables

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| RESULT | 1st column | DATA_ID (in input table) data type | DATA_ID (in input table) column name | ID of data |

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| | 2nd column | INTEGER | IS_OUT_OF_RANGE | Test result:<br>• 0: a value is in the range<br>• 1: a value is out of range |
| STATISTICS | 1st column | NVARCHAR | STAT_NAME | Statistics name |
| | 2nd column | DOUBLE | STAT_VALUE | Statistics value |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_IQR_DATA_TBL;
CREATE COLUMN TABLE PAL_IQR_DATA_TBL (
    "ID" VARCHAR(10),
    "VAL" DOUBLE
);
INSERT INTO PAL_IQR_DATA_TBL VALUES ('P1', 10);
INSERT INTO PAL_IQR_DATA_TBL VALUES ('P2', 11);
INSERT INTO PAL_IQR_DATA_TBL VALUES ('P3', 10);
INSERT INTO PAL_IQR_DATA_TBL VALUES ('P4', 9);
INSERT INTO PAL_IQR_DATA_TBL VALUES ('P5', 10);
INSERT INTO PAL_IQR_DATA_TBL VALUES ('P6', 24);
INSERT INTO PAL_IQR_DATA_TBL VALUES ('P7', 11);
INSERT INTO PAL_IQR_DATA_TBL VALUES ('P8', 12);
INSERT INTO PAL_IQR_DATA_TBL VALUES ('P9', 10);
INSERT INTO PAL_IQR_DATA_TBL VALUES ('P10', 9);
INSERT INTO PAL_IQR_DATA_TBL VALUES ('P11', 1);
INSERT INTO PAL_IQR_DATA_TBL VALUES ('P12', 11);
INSERT INTO PAL_IQR_DATA_TBL VALUES ('P13', 12);
INSERT INTO PAL_IQR_DATA_TBL VALUES ('P14', 13);
INSERT INTO PAL_IQR_DATA_TBL VALUES ('P15', 12);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MULTIPLIER', null, 1.5, null);
CALL "_SYS_AFL"."PAL_IQR"(PAL_IQR_DATA_TBL, #PAL_PARAMETER_TBL, ?, ?);
```

### Expected Result

RESULT:

| | ID | IS_OUT_OF_RANGE |
|---|---|---|
| 1 | P1 | 0 |
| 2 | P2 | 0 |
| 3 | P3 | 0 |
| 4 | P4 | 0 |
| 5 | P5 | 0 |
| 6 | P6 | 1 |
| 7 | P7 | 0 |
| 8 | P8 | 0 |
| 9 | P9 | 0 |
| 10 | P10 | 0 |
| 11 | P11 | 1 |
| 12 | P12 | 0 |
| 13 | P13 | 0 |
| 14 | P14 | 0 |
| 15 | P15 | 0 |

STATISTICS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | lower quartile | 10 |
| 2 | upper quartile | 12 |

> **Note**
>
> The wrapper procedure generation approach to call PAL functions is still supported. For more information, refer to the older versions (prior to SAP HANA Platform 2.0 SPS 02) of this document.

## 5.8.7 MCMC Sampling

Markov Chain Monte Carlo (MCMC) generates samples from a given distribution.

The probability density functions of supported distributions are defined as follows:

Normal

$$p(y|\mu,\sigma)=\frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{1}{2}\left(\frac{y-\mu}{\sigma}\right)^2\right)$$

Skew Normal

$$p(y|\xi,\omega,\alpha)=\frac{1}{\omega\sqrt{2\pi}}\exp\left(-\frac{1}{2}\left(\frac{y-\xi}{\omega}\right)^2\right)\left(1+\mathrm{erf}\left(\alpha\left(\frac{y-\xi}{\omega\sqrt{2}}\right)\right)\right)$$

Student-T

$$p(y|\nu,\mu,\sigma)=\frac{\Gamma\left((\nu+1)/2\right)}{\Gamma(\nu/2)}\frac{1}{\sqrt{\nu\pi}\sigma}\left(1+\frac{1}{\nu}\left(\frac{y-\mu}{\sigma}\right)^2\right)^{-(\nu+1)/2}$$

Cauchy

$$p(y|\mu,\sigma)=\frac{1}{\pi\sigma}\frac{1}{1+\left((y-\mu)/\sigma\right)^2}$$

Laplace

$$p(y|\mu,\sigma) = \frac{1}{2\sigma}\exp\left(-\frac{|y-\mu|}{\sigma}\right)$$

Logistic

$$p(y|\mu,\sigma) = \frac{1}{\sigma}\exp\left(-\frac{y-\mu}{\sigma}\right)\left(1+\exp\left(-\frac{y-\mu}{\sigma}\right)\right)^{-2}$$

Gumbel

$$p(y|\mu,\beta)=\frac{1}{\beta}\exp\left(-\frac{y-\mu}{\beta}-\exp\left(-\frac{y-\mu}{\beta}\right)\right)$$

Exponential

$$p(y|\beta)=\beta\ \exp(-\beta y)$$

Chi-Square

$$p(y|\nu)=\frac{2^{-\nu/2}}{\Gamma(\nu/2)}y^{\nu/2-1}\exp\left(-\frac{1}{2}y\right)$$

Inverse Chi-Square

$$p(y|\nu)=\frac{2^{-\nu/2}}{\Gamma(\nu/2)}y^{-\nu/2-1}\exp\left(-\frac{1}{2}\frac{1}{y}\right)$$

Gamma

$$p(y|\alpha,\beta)=\frac{\beta^{\alpha}}{\Gamma(\alpha)}y^{\alpha-1}\exp(-\beta y)$$

Weibull

$$p(y|\alpha,\sigma)=\frac{\alpha}{\sigma}\left(\frac{y}{\sigma}\right)^{\alpha-1}\exp\left(-\left(\frac{y}{\sigma}\right)^{\alpha}\right)$$

Frechet

$$p(y|\alpha,\sigma)=\frac{\alpha}{\sigma}\left(\frac{y}{\sigma}\right)^{-\alpha-1}\exp\left(-\left(\frac{y}{\sigma}\right)^{-\alpha}\right)$$

Rayleigh

$$p(y|\sigma)=\frac{y}{\sigma^2}\exp(-y^2/2\sigma^2)$$

Multivariate Normal

$$\mathrm{MultiNormal}(y|\mu,\Sigma)=\frac{1}{(2\pi)^{K/2}}\frac{1}{\sqrt{|\Sigma|}}\exp\left(-\frac{1}{2}(y-\mu)^{\top}\Sigma^{-1}(y-\mu)\right)$$

Multivariate Normal with Precision Parameterization

$$p(y|\mu,\Omega)=\mathrm{MultiNormal}(y|\mu,\Omega^{-1})$$

Multivariate Normal with Cholesky Parameterization

$$p(y|\mu,L)=\mathrm{MultiNormal}(y|\mu,LL^{\square})$$

Multivariate Student-T

$$p(y|\nu,\mu,\Sigma)=\frac{1}{\pi^{K/2}}\frac{1}{\nu^{K/2}}\frac{\Gamma\left((\nu+K)/2\right)}{\Gamma(\nu/2)}\frac{1}{\sqrt{|\Sigma|}}\left(1+\frac{1}{\nu}(y-\mu)^{\top}\Sigma^{-1}(y-\mu)\right)^{-(\nu+K)/2}$$

Dirichlet

$$p(y|\alpha)=\frac{\Gamma(\sum_{k=1}^{K}\alpha_k)}{\prod_{k=1}^{K}\Gamma(\alpha_k)}\prod_{k=1}^{K}y_k^{\alpha_k-1}$$

Beta

$$p(y|\alpha,\beta)=\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}y^{\alpha-1}(1-y)^{\beta-1}$$

Inverse Gamma

$$p(y|\alpha,\beta)=\frac{\beta^{\alpha}}{\Gamma(\alpha)}y^{-(\alpha+1)}\exp\left(-\beta\frac{1}{y}\right)$$

Lognormal

$$p(y|\mu,\sigma)=\frac{1}{\sqrt{2\pi}\sigma}\frac{1}{y}\exp\left(-\frac{1}{2}\left(\frac{\log y-\mu}{\sigma}\right)^2\right)$$

Pareto

$$p\left(y|y_{min},\alpha\right)=\frac{\alpha y_{min}{}^{\alpha}}{y^{\alpha+1}}$$

Lomax

$$p(y|\lambda,\alpha)=\frac{\alpha}{\lambda}\left(1+\frac{y}{\lambda}\right)^{-(\alpha+1)}$$

Distribution Parameters

| Distribution Name | Parameters |
|---|---|
| Normal | MU and SIGMA |
| Skew Normal | XI and OMEGA, and ALPHA |
| Student-T | NU, MU, and SIGMA |
| Cauchy | MU and SIGMA |
| Laplace | MU and SIGMA |
| Logistic | MU and SIGMA |
| Gumbel | MU and BETA |
| Exponential | BETA |
| Chi-Square | NU |
| Inverse Chi-Square | NU |
| Gamma | ALPHA and BETA |
| Weibull | ALPHA and SIGMA |
| Frechet | ALPHA and SIGMA |
| Rayleigh | SIGMA |
| Multivariate Normal | MU and SIGMA |
| Multivariate Normal with Precision Parameterization | MU and OMEGA |
| Multivariate Normal with Cholesky Parameterization | MU and L |
| Multivariate Student-T | NU, MU, and SIGMA |
| Dirichlet | ALPHA |
| Beta | ALPHA and BETA |
| Inverse Gamma | ALPHA and BETA |
| Lognormal | MU and SIGMA |
| Pareto | Y_MIN and ALPHA |
| Lomax | LAMBDA and ALPHA |

# _SYS_AFL.PAL_MCMC

## Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<PARAMETER table>` | IN |
| 2 | `<RESULT table>` | OUT |

## Input Table

Mandatory parameter

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| DISTRIBUTION_NAME | VARCHAR or NVARCHAR | None | Specify the distribution:<br>• normal<br>• skew_normal<br>• student_t<br>• cauchy<br>• laplace<br>• logistic<br>• gumbel<br>• exponential<br>• lognormal<br>• chi_square<br>• invchi_square<br>• gamma<br>• invgamma<br>• weibull<br>• frechet<br>• rayleigh<br>• beta<br>• pareto<br>• lomax<br>• multinormal<br>• multinormalprec<br>• multinormalcholesky<br>• multistudent_t<br>• dirichlet |

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| DISTRIBUTION_PARAM | VARCHAR or NVARCHAR | None | Specifies the distribution parameters in a json format. For example: `'{"NU":[1], "MU":[0,0], "SIGMA":[1,0,0,1]}'`. The length of distribution parameters should be in accord with DIMENSION (default value is 1). Otherwise superfluous values will be discarded and insufficient values will be auto filled with 0 or 1. |

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default | Description | Dependency |
|---|---|---|---|---|
| ITER | INTEGER | 2000 | Number of iterations for each chain including warmup | |
| RANDOM_SEED | INTEGER | 0 | Seed used to initialize the random number generator | |
| INIT_RADIUS | DOUBLE | 2.0 | Radius to initialize | |
| ADAPT_ENGAGED | INTEGER | 1 | Whether to use adaptation | |
| WARMUP | INTEGER | ITER/2 | Number of warmup iterations | |
| THIN | INTEGER | 1 | Period for saving samples | |
| ADAPT_GAMMA | DOUBLE | 0.05 | Adaptation regularization scale, constraint (0, infty) | Only valid when ADAPT_ENGAGED is true. |
| ADAPT_DELTA | DOUBLE | 0.8 | Target Metropolis acceptance rate, constraint [0, 1] | |

| Name | Data Type | Default | Description | Dependency |
|---|---|---|---|---|
| ADAPT_KAPPA | DOUBLE | 0.75 | Adaptation relaxation exponent, constraint (0, infty) | |
| ADAPT_T0 | DOUBLE | 10.0 | Adaptation iteration offset, constraint (0, infty) | |
| ADAPT_INIT_BUFFER | INTEGER | 75 | Width of the initial fast adaptation interval | |
| ADAPT_TERM_BUFFER | INTEGER | 50 | Width of the final fast adaptation interval | |
| ADAPT_WINDOW | INTEGER | 25 | Initial width of slow adaptation interval | |
| STEPSIZE | DOUBLE | 1.0 | Discretization time interval | |
| STEPSIZE_JITTER | DOUBLE | 0.0 | Uniform random jitter of step size | |
| MAX_TREEDEPTH | INTEGER | 10 | Maximum tree depth | |
| DIMENSION | INTEGER | 1 | Dimensions of output | Mandatory only when the value of DISTRIBUTION_NAME is dirichlet. |

### Output Tables

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| RESULT | 1st column | INTEGER | ID | IDs |
| | 2nd column | NCLOB | SAMPLES | Generated samples |

### Examples

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA DM_PAL;

DROP TABLE PAL_MCMC_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_MCMC_PARAMETER_TBL (
        "PARAM_NAME" VARCHAR(256),
```

```
        "INT_VALUE" INTEGER,
        "DOUBLE_VALUE" DOUBLE,
        "STRING_VALUE" VARCHAR(1000)
);
INSERT INTO PAL_MCMC_PARAMETER_TBL VALUES ('DISTRIBUTION_NAME', NULL, NULL,
'multinormal');
INSERT INTO PAL_MCMC_PARAMETER_TBL VALUES ('DIMENSION', 2, NULL, NULL);
--the covariance matrix Sigma stores in column-major order
INSERT INTO PAL_MCMC_PARAMETER_TBL VALUES ('DISTRIBUTION_PARAM', NULL, NULL,
'{"MU":[1,10], "SIGMA":[1,0,0,5]}');
INSERT INTO PAL_MCMC_PARAMETER_TBL VALUES ('ITER', 2000, NULL, NULL);
INSERT INTO PAL_MCMC_PARAMETER_TBL VALUES ('THIN', 10, NULL, NULL);

DROP TABLE PAL_MCMC_RESULT_TBL;
CREATE COLUMN TABLE PAL_MCMC_RESULT_TBL ("ID" INTEGER, "SAMPLES" NCLOB);

DO BEGIN
  lt_param = SELECT * FROM PAL_MCMC_PARAMETER_TBL;
  CALL _SYS_AFL.PAL_MCMC (:lt_param, lt_result);
  INSERT INTO PAL_MCMC_RESULT_TBL SELECT * FROM :lt_result;
END;

SELECT * FROM PAL_MCMC_RESULT_TBL;
```

**Expected Result**

| ID | SAMPLES |
|---|---|
| 0 | [-0.3847637703128550,0.005918072773587912] |
| 1 | [1.719465688529201,8.282673042589110] |
| 2 | [0.4948567878325943,9.333670290297809] |
| 3 | [1.783880136943846,10.51239376078307] |
| 4 | [-0.1929886761797402,6.881991677982647] |
| 5 | [0.5798111307968541,9.509248379614490] |
| 6 | [2.476462367239271,9.981583883644763] |
| 7 | [2.843904542919752,10.98070827061291] |
| 8 | [1.575740025824908,10.02056551215444] |
| 9 | [0.2561491800429305,5.154370551586896] |
| 10 | [2.135361463836434,8.638837443936444] |
| 11 | [1.394928635047403,10.99760259054135] |
| 12 | [0.9119939627863762,9.472894657100834] |
| 13 | [1.020038564775359,9.234200380309382] |
| 14 | [1.758071510338004,10.99249093481913] |
| 15 | [1.02771693801950,8.928249052712717] |

## 5.8.8 Missing Value Handling

This function aims to handle missing values in a table.

Missing values in each column of the table can be handled by one of the following imputation types:

- MODE: Fills all missing values by the mode of the column, i.e., the value that appears most often in the column. This type only suits column with categorical variables.
- MEAN: Fills all missing values by the mean of the column, i.e., the average of all non-missing values in the column. This type only suits column with numerical variables.
- MEDIAN: Fills all missing values by the median of the column, i.e., the value that separates the higher half of the non-missing values from the lower half. If there is an even number of observations, then the median is given by the mean of the two middle values. This type only suits column with numerical variables.
- ALS: Fills each missing value by the value imputed by a matrix completion model trained using alternating least squares method. This type only suits column with numerical variables.
- SPECIFIED_CATEGORICAL_VALUE: Fills all missing values by a specified categorical value. This type only suits column with categorical variables.
- SPECIFIED_NUMERICAL_VALUE: Fills all missing values by a specified numerical value. This type only suits column with numerical variables.
- DELETE: Deletes all rows with missing values. The entire row in table will be deleted.
- NONE: Does nothing. Leave the column untouched.

Imputation type MODE and SPECIFIED_CATEGORICAL_VALUE only suit for columns with categorical variables. Imputation type MEAN, MEDIAN, ALS, and SPECIFIED_NUMERICAL_VALUE only suit for columns with numerical variables. By default, columns of integer type and double type will be treated as numerical column, columns of string type will be treated as categorical column. Columns of integer type can also be indicated as categorical column by explicitly setting as CATEGORICAL_VARIABLE. Besides, if a column of integer type is set with a categorical only imputation type (MODE or SPECIFIED_CATEGORICAL_VALUE), it will be treated as a categorical column automatically.

The output statistics table can serve as a model to handle missing values in new tables with procedure _SYS_AFL.PAL_MISSING_VALUE_HANDLING_WITH_MODEL. In general, the new tables to be handled should have the same column structure (number of columns, column names, and column types) with the table with which the model is trained.

Note about ALS:

- Missing values in columns set with imputation type ALS will be imputed by a matrix completion model trained using the alternating least squares method. The model is trained by solving the problem:

  $$\mathrm{argmin}_{U,V}\frac{1}{|D|}\left(\|P_D(A-UV^T)\|_F^2+\lambda\left(R(U)+R(V)\right)\right),$$

  where $A\in R^{m\times n}$ is the matrix formed by all numerical columns (both columns with imputation type ALS and columns with other imputation types, except for the ID column if HAS_ID = 1) in the table, $m$ is the number of rows, $n$ is the number of numerical columns, $U\in R^{m\times f}$, $V\in R^{n\times f}$ are the two factor matrices to be trained, $f$ is the factor number. $D$ is the set of indices of all non-missing values in $A$, for any matrix $M$, $P_D(M)$ is a matrix defined as

  $$P_D(M)(i,\ j)=\begin{cases}M_{ij}, & \text{if } (i,j)\in D,\\ 0, & \text{else.}\end{cases}$$

$\lambda$ is the regularization parameter, $R(U)=\sum_i n_i\|u_i\|^2$, $R(V)=\sum_j n_j\|v_j\|^2$ are the regularization term of U,V, where $n_i$ is the number of non-missing values in the i-th row of A, $n_j$ is the number of non-missing values in the j-th column of A, and $u_i$,$v_j$ are the i-th row of U and the j-th row of V, respectively, that is, the factor vectors.
After the two factor matrices U,V are trained, the missing value of matrix A at index (i,j) will be imputed as the inner product of the corresponding factor vectors: $\hat{A}_{ij}=u_i v_j^T$.
For more information about ALS, see Alternating Least Squares [page 1310] under the *Recommender System Algorithms* category, where the procedure uses the same matrix completion model to build up recommendation systems.

- The number of factors should be set less than the number of numerical columns to get a meaningful result.
- Imputation type ALS is not suitable when impute new tables with model, do not use type ALS when you need to impute new tables with the model.
- Training of ALS can be time consuming (comparing to other imputation types) when the table is large and the number of factors is big.
- The final value of the cost function and RMSE (root-mean-squared error) of the trained model can be found in the statistics table.

## Prerequisite

None.

## _SYS_AFL.PAL_MISSING_VALUE_HANDLING

This procedure handles missing values in a table.

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<RESULT table>` | OUT |
| 4 | `<STATISTICS/MODEL table>` | OUT |

### Input Table

| Table | Column | Reference for Output Table | Data Type | Description |
|---|---|---|---|---|
| DATA | Columns | DATA_FEATURES | INTEGER, DOUBLE, DECIMAL(p, s), VARCHAR, or NVARCHAR | Attribute data |

### Parameter Table

Mandatory Parameter

None.

Optional Parameter

The following parameter is optional. If it is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| IMPUTATION_TYPE | INTEGER | 1 | Chooses the overall default imputation type for all columns:<br>• 0: NONE,<br>• 1: MODE_MEAN,<br>• 2: MODE_MEDIAN,<br>• 3: MODE_ALLZERO,<br>• 4: MODE_ALS,<br>• 5: DELETE.<br>NONE and DELETE: set all columns to imputation type NONE/DELETE.<br>MODE_MEAN, MODE_MEDIAN, MODE_ALLZERO, and MODE_ALS: set categorical columns to the first imputation type (the type before "_"); set numerical columns to the second imputation type (the type after "_"). ALLZERO means to fill all missing values in numerical columns with 0. |

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| <column name>_IMPUTATION_TYPE | INTEGER | No default value | Sets imputation type of a specified column. This value overwrites the default imputation type deduced by the IMPUTATION_TYPE parameter:<br>• 0: NONE,<br>• 1: DELETE,<br>• 100: MODE,<br>• 101: SPECIFIED_CATEGORICAL_VALUE,<br>• 200: MEAN,<br>• 201: MEDIAN,<br>• 203: SPECIFIED_NUMERICAL_VALUE,<br>• 204: ALS.<br>For type 101 (SPECIFIED_CATEGORICAL_VALUE), use the following syntax to specify a categorical value to be filled in a column V0:<br>`INSERT INTO #PAL_PARAMETER_TBL VALUES('V0_IMPUTATION_TYPE', 101, NULL, '<CATEGORICAL_VALUE>');`<br>For type 203 (SPECIFIED_NUMERICAL_VALUE), use the following syntax to specify a numerical value to be filled in a column V1:<br>`INSERT INTO #PAL_PARAMETER_TBL VALUES('V1_IMPUTATION_TYPE', 203, <NUMERICAL_VALUE>, NULL);` |

| Name | Data Type | Default Value | Description |
| --- | --- | --- | --- |
| | | | **Note**<br>A column of integer type can be set to type 101 (SPECIFIED_CATEGORICAL_VALUE) or type 203 (SPECIFIED_NUMERICAL_VALUE):<br>• when setting to type 101 (SPECIFIED_CATEGORICAL_VALUE), the string must contain a valid integer value, otherwise an error will be thrown;<br>• when setting to type 203 (SPECIFIED_NUMERICAL_VALUE), the double value will be converted to integer. |
| ALS_FACTOR_NUMBER | INTEGER | 3 | Length of factor vectors, that is, the $f$ in ALS model.<br>Note: ALS_FACTOR_NUMBER should be less than the number of numerical columns to get a meaningful result. |
| ALS_REGULARIZATION | DOUBLE | 1e-2 | L2 regularization of the factors, that is, the $\lambda$ in ALS model. |
| ALS_MAX_ITERATION | INTEGER | 20 | Maximum number of iterations when training ALS model. |

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| ALS_SEED | INTEGER | 0 | Specifies the seed of the random number generator used in the training of ALS model:<br>• 0: Uses the current time as the seed,<br>• Others: Uses the specified value as the seed. |
| ALS_EXIT_THRESHOLD | DOUBLE | 0.0 | The training of ALS model exits if the value of cost function is not decreased more than the value of ALS_EXIT_THRESHOLD since the last check. If ALS_EXIT_THRESHOLD is set to 0, there is no check and the training only exits on reaching the maximum number of iterations. Evaluations of cost function require additional calculations. You can set this parameter to 0 to avoid it. |
| ALS_EXIT_INTERVAL | INTEGER | 5 | During the training of ALS model, it will calculate cost function and check the exit criteria every `<ALS_EXIT_INTERVAL>` iterations. Evaluations of cost function require additional calculations. |
| ALS_LINEAR_SYSTEM_SOLVER | INTEGER | 0 | • 0: cholesky solver<br>• 1: cg solver (recommended when ALS_FACTOR_NUMBER is large) |
| ALS_CG_MAX_ITERATION | INTEGER | 3 | Maximum number of iteration of cg solver. |
| ALS_CENTERING | INTEGER | 1 | Whether to center the data by column before training ALS model.<br>• 0: no centering,<br>• 1: centering. |

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| ALS_SCALING | INTEGER | 1 | Whether to scale the data by column before training ALS model.<br>• 0: no scaling,<br>• 1: scaling. |
| CATEGORICAL_VARIABLE | NVARCHAR | No default value | Indicates whether a column of INTEGER type is corresponding to a categorical variable. By default, columns of VARCHAR or NVARCHAR type are treated as categorical variable, columns of INTEGER or DOUBLE type are treated as numerical variable. |
| HAS_ID | INTEGER | 0 | Specifies whether the input data table has an ID column. If it exists, the ID column must be the first column.<br>• 0: indicates there is no ID column in the data table. The algorithm will impute missing values in all the columns.<br>• 1: indicates that the first column of the data table is ID column. The algorithm treats the ID column as imputation type NONE. |
| THREAD_RATIO | DOUBLE | 0.0 | Specifies the ratio of total number of threads that can be used by this function. The value range is from 0 to 1, where 0 means only using 1 thread, and 1 means using at most all the currently available threads. Values outside the range are ignored and this function heuristically determines the number of threads to use. |

### Output Tables

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| RESULT | Columns | DATA_FEATURES (in input table) data type | DATA_FEATURES (in input table) column name | Each column type should match the type of the original input data. |
| STATISTICS/MODEL | 1st column | NVARCHAR(256) | STAT_NAME | Statistic names. |
| | 2nd column | NVARCHAR(5000) | STAT_VALUE | Statistic values. |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

#### Example 1: Impute missing values

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_MISSING_VALUES_DATA_TBL;
CREATE COLUMN TABLE PAL_MISSING_VALUES_DATA_TBL ("V0" INTEGER, "V1" INTEGER,
"V2" VARCHAR(5), "V3" DOUBLE, "V4" DOUBLE, "V5" DOUBLE);
INSERT INTO PAL_MISSING_VALUES_DATA_TBL VALUES (10,   0,    'D',  NULL, 1.4,
23.6);
INSERT INTO PAL_MISSING_VALUES_DATA_TBL VALUES (20,   1,    'A',  0.4,  1.3,
21.8);
INSERT INTO PAL_MISSING_VALUES_DATA_TBL VALUES (50,   1,    'C',  NULL, 1.6,
21.9);
INSERT INTO PAL_MISSING_VALUES_DATA_TBL VALUES (30,   NULL, 'B',  0.8,  1.7,
22.6);
INSERT INTO PAL_MISSING_VALUES_DATA_TBL VALUES (10,   0,    'A',  0.2,  NULL,
NULL);
INSERT INTO PAL_MISSING_VALUES_DATA_TBL VALUES (10,   0,    NULL, 0.5,  1.8,
19.7);
INSERT INTO PAL_MISSING_VALUES_DATA_TBL VALUES (NULL, 0,    'C',  0.5,  NULL,
17.8);
INSERT INTO PAL_MISSING_VALUES_DATA_TBL VALUES (10,   1,    'A',  0.6,  1.6,
24.9);
INSERT INTO PAL_MISSING_VALUES_DATA_TBL VALUES (20,   NULL, 'D',  0.9,  1.7,
22.2);
INSERT INTO PAL_MISSING_VALUES_DATA_TBL VALUES (30,   1,    'D',  0.4,  1.3,
NULL);
INSERT INTO PAL_MISSING_VALUES_DATA_TBL VALUES (50,   0,    NULL, 0.3,  1.2,
16.4);
INSERT INTO PAL_MISSING_VALUES_DATA_TBL VALUES (NULL, 1,    'B',  0.7,  1.2,
19.3);
INSERT INTO PAL_MISSING_VALUES_DATA_TBL VALUES (30,   1,    'A',  0.2,  1.1,
21.7);
INSERT INTO PAL_MISSING_VALUES_DATA_TBL VALUES (30,   0,    'D',  NULL, NULL,
NULL);
INSERT INTO PAL_MISSING_VALUES_DATA_TBL VALUES (NULL, 1,    'C',  0.5,  1.8,
18.6);
INSERT INTO PAL_MISSING_VALUES_DATA_TBL VALUES (20,   0,    'A',  0.6,  1.4,
17.9);
DROP TABLE #PAL_PARAMETER_TBL;
```

```
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL ("PARAM_NAME" VARCHAR
(256), "INT_VALUE" INTEGER, "DOUBLE_VALUE" DOUBLE, "STRING_VALUE" VARCHAR
(1000));
INSERT INTO #PAL_PARAMETER_TBL VALUES ('IMPUTATION_TYPE', 1, NULL, NULL); -- use
overall imputation type MODE_MEAN
INSERT INTO #PAL_PARAMETER_TBL VALUES ('CATEGORICAL_VARIABLE', NULL, NULL,
'V1'); -- indicate that 'V1' is a categorical variable
INSERT INTO #PAL_PARAMETER_TBL VALUES ('V1_IMPUTATION_TYPE', 101, NULL, '0'); --
set imputation type of column 'V1' to be 101 (SPECIFIED_CATEGORICAL_VALUE), and
the specified categorical value to be '0'
DROP TABLE PAL_MISSING_VALUES_STATISTICS_TBL;
CREATE COLUMN TABLE PAL_MISSING_VALUES_STATISTICS_TBL ("STAT_NAME"
NVARCHAR(256), "STAT_VALUE" NVARCHAR(5000));
CALL "_SYS_AFL"."PAL_MISSING_VALUE_HANDLING"(PAL_MISSING_VALUES_DATA_TBL,
#PAL_PARAMETER_TBL, ?, PAL_MISSING_VALUES_STATISTICS_TBL) WITH OVERVIEW;
```

**Expected Result**

RESULT:

| | V0 | V1 | V2 | V3 | V4 | V5 |
|---|---|---|---|---|---|---|
| 1 | 10 | 0 | D | 0.5076923076923077 | 1.4 | 23.6 |
| 2 | 20 | 1 | A | 0.4 | 1.3 | 21.8 |
| 3 | 50 | 1 | C | 0.5076923076923077 | 1.6 | 21.9 |
| 4 | 30 | 0 | B | 0.8 | 1.7 | 22.6 |
| 5 | 10 | 0 | A | 0.2 | 1.4692307692307693 | 20.646153846153844 |
| 6 | 10 | 0 | A | 0.5 | 1.8 | 19.7 |
| 7 | 24 | 0 | C | 0.5 | 1.4692307692307693 | 17.8 |
| 8 | 10 | 1 | A | 0.6 | 1.6 | 24.9 |
| 9 | 20 | 0 | D | 0.9 | 1.7 | 22.2 |
| 10 | 30 | 1 | D | 0.4 | 1.3 | 20.646153846153844 |
| 11 | 50 | 0 | A | 0.3 | 1.2 | 16.4 |
| 12 | 24 | 1 | B | 0.7 | 1.2 | 19.3 |
| 13 | 30 | 1 | A | 0.2 | 1.1 | 21.7 |
| 14 | 30 | 0 | D | 0.5076923076923077 | 1.4692307692307693 | 20.646153846153844 |
| 15 | 24 | 1 | C | 0.5 | 1.8 | 18.6 |
| 16 | 20 | 0 | A | 0.6 | 1.4 | 17.9 |

STATISTICS/MODEL:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | V0.NUMBER_OF_NULLS | 3 |
| 2 | V0.IMPUTATION_TYPE | MEAN |
| 3 | V0.IMPUTED_VALUE | 24 |
| 4 | V1.NUMBER_OF_NULLS | 2 |
| 5 | V1.IMPUTATION_TYPE | SPECIFIED_CATEGORICAL_VALUE |
| 6 | V1.IMPUTED_VALUE | 0 |
| 7 | V2.NUMBER_OF_NULLS | 2 |
| 8 | V2.IMPUTATION_TYPE | MODE |
| 9 | V2.IMPUTED_VALUE | A |
| 10 | V3.NUMBER_OF_NULLS | 3 |
| 11 | V3.IMPUTATION_TYPE | MEAN |
| 12 | V3.IMPUTED_VALUE | 0.507692307692307 |
| 13 | V4.NUMBER_OF_NULLS | 3 |
| 14 | V4.IMPUTATION_TYPE | MEAN |
| 15 | V4.IMPUTED_VALUE | 1.469230769230769 |
| 16 | V5.NUMBER_OF_NULLS | 3 |
| 17 | V5.IMPUTATION_TYPE | MEAN |
| 18 | V5.IMPUTED_VALUE | 20.64615384615384 |

Example 2: Impute missing values using ALS

The returned statistics table will not be suitable for imputing missing values in new tables with _SYS_AFL.PAL_MISSING_VALUE_HANDLING_WITH_MODEL.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_MISSING_VALUES_DATA_TBL;
CREATE COLUMN TABLE PAL_MISSING_VALUES_DATA_TBL ("V0" INTEGER, "V1" INTEGER,
"V2" VARCHAR(5), "V3" DOUBLE, "V4" DOUBLE, "V5" DOUBLE);
INSERT INTO PAL_MISSING_VALUES_DATA_TBL VALUES (10,   0,    'D',  NULL, 1.4,
23.6);
INSERT INTO PAL_MISSING_VALUES_DATA_TBL VALUES (20,   1,    'A',  0.4,  1.3,
21.8);
INSERT INTO PAL_MISSING_VALUES_DATA_TBL VALUES (50,   1,    'C',  NULL, 1.6,
21.9);
INSERT INTO PAL_MISSING_VALUES_DATA_TBL VALUES (30,   NULL, 'B',  0.8,  1.7,
22.6);
INSERT INTO PAL_MISSING_VALUES_DATA_TBL VALUES (10,   0,    'A',  0.2,  NULL,
NULL);
INSERT INTO PAL_MISSING_VALUES_DATA_TBL VALUES (10,   0,    NULL, 0.5,  1.8,
19.7);
INSERT INTO PAL_MISSING_VALUES_DATA_TBL VALUES (NULL, 0,    'C',  0.5,  NULL,
17.8);
INSERT INTO PAL_MISSING_VALUES_DATA_TBL VALUES (10,   1,    'A',  0.6,  1.6,
24.9);
INSERT INTO PAL_MISSING_VALUES_DATA_TBL VALUES (20,   NULL, 'D',  0.9,  1.7,
22.2);
```

```
INSERT INTO PAL_MISSING_VALUES_DATA_TBL VALUES (30,   1,    'D',  0.4,  1.3,
NULL);
INSERT INTO PAL_MISSING_VALUES_DATA_TBL VALUES (50,   0,    NULL, 0.3,  1.2,
16.4);
INSERT INTO PAL_MISSING_VALUES_DATA_TBL VALUES (NULL, 1,    'B',  0.7,  1.2,
19.3);
INSERT INTO PAL_MISSING_VALUES_DATA_TBL VALUES (30,   1,    'A',  0.2,  1.1,
21.7);
INSERT INTO PAL_MISSING_VALUES_DATA_TBL VALUES (30,   0,    'D',  NULL, NULL,
NULL);
INSERT INTO PAL_MISSING_VALUES_DATA_TBL VALUES (NULL, 1,    'C',  0.5,  1.8,
18.6);
INSERT INTO PAL_MISSING_VALUES_DATA_TBL VALUES (20,   0,    'A',  0.6,  1.4,
17.9);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL ("PARAM_NAME" VARCHAR
(256), "INT_VALUE" INTEGER, "DOUBLE_VALUE" DOUBLE, "STRING_VALUE" VARCHAR
(1000));
INSERT INTO #PAL_PARAMETER_TBL VALUES ('IMPUTATION_TYPE', 4, NULL, NULL); -- use
overall imputation type MODE_ALS
INSERT INTO #PAL_PARAMETER_TBL VALUES ('CATEGORICAL_VARIABLE', NULL, NULL,
'V1'); -- indicate that 'V1' is a categorical variable, so it will not involved
in the training of ALS model
INSERT INTO #PAL_PARAMETER_TBL VALUES ('ALS_FACTOR_NUMBER', 2, NULL, NULL); --
ALS_FACTOR_NUMBER should be set less than the number of numerical columns (which
is 4 in this example) to get a meaningful result
INSERT INTO #PAL_PARAMETER_TBL VALUES ('ALS_SEED', 1, NULL, NULL);
CALL "_SYS_AFL"."PAL_MISSING_VALUE_HANDLING"(PAL_MISSING_VALUES_DATA_TBL,
#PAL_PARAMETER_TBL, ?, ?);
```

### Expected Result

RESULT:

| | V0 | V1 | V2 | V3 | V4 | V5 |
|---|---|---|---|---|---|---|
| 1 | 10 | 0 | D | 0.30695687684157014 | 1.4 | 23.6 |
| 2 | 20 | 1 | A | 0.4 | 1.3 | 21.8 |
| 3 | 50 | 1 | C | 0.9306894630164242 | 1.6 | 21.9 |
| 4 | 30 | 0 | B | 0.8 | 1.7 | 22.6 |
| 5 | 10 | 0 | A | 0.2 | 1.3336681902102172 | 21.371752778776393 |
| 6 | 10 | 0 | A | 0.5 | 1.8 | 19.7 |
| 7 | 45 | 0 | C | 0.5 | 1.273300223923987 | 17.8 |
| 8 | 10 | 1 | A | 0.6 | 1.6 | 24.9 |
| 9 | 20 | 0 | D | 0.9 | 1.7 | 22.2 |
| 10 | 30 | 1 | D | 0.4 | 1.3 | 19.445476101127387 |
| 11 | 50 | 0 | A | 0.3 | 1.2 | 16.4 |
| 12 | 46 | 1 | B | 0.7 | 1.2 | 19.3 |
| 13 | 30 | 1 | A | 0.2 | 1.1 | 21.7 |
| 14 | 30 | 0 | D | 0.5603419174711064 | 1.4652600120891217 | 20.137886120100966 |
| 15 | 28 | 1 | C | 0.5 | 1.8 | 18.6 |
| 16 | 20 | 0 | A | 0.6 | 1.4 | 17.9 |

STATISTICS/MODEL:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | V0.NUMBER_OF_NULLS | 3 |
| 2 | V0.IMPUTATION_TYPE | ALS |
| 3 | V0.IMPUTED_VALUE | PAL_ALS_SPECIFIED (ITER 20, COST 0.223748, RMSE 0.442162) |
| 4 | V1.NUMBER_OF_NULLS | 2 |
| 5 | V1.IMPUTATION_TYPE | MODE |
| 6 | V1.IMPUTED_VALUE | 0 |
| 7 | V2.NUMBER_OF_NULLS | 2 |
| 8 | V2.IMPUTATION_TYPE | MODE |
| 9 | V2.IMPUTED_VALUE | A |
| 10 | V3.NUMBER_OF_NULLS | 3 |
| 11 | V3.IMPUTATION_TYPE | ALS |
| 12 | V3.IMPUTED_VALUE | PAL_ALS_SPECIFIED (ITER 20, COST 0.223748, RMSE 0.442162) |
| 13 | V4.NUMBER_OF_NULLS | 3 |
| 14 | V4.IMPUTATION_TYPE | ALS |
| 15 | V4.IMPUTED_VALUE | PAL_ALS_SPECIFIED (ITER 20, COST 0.223748, RMSE 0.442162) |
| 16 | V5.NUMBER_OF_NULLS | 3 |
| 17 | V5.IMPUTATION_TYPE | ALS |
| 18 | V5.IMPUTED_VALUE | PAL_ALS_SPECIFIED (ITER 20, COST 0.223748, RMSE 0.442162) |

## _SYS_AFL.PAL_MISSING_VALUE_HANDLING_WITH_MODEL

This procedure handles missing values in new tables using the statistics/model table returned by _SYS_AFL.PAL_MISSING_VALUE_HANDLING. In general, the new tables to be handled should have the same column structure (number of columns, column names, and column types) with the table with which the model is trained.

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<STATISTICS/MODEL table>` | IN |
| 3 | `<PARAMETER table>` | IN |
| 4 | `<RESULT table>` | OUT |
| 5 | `<STATISTICS table>` | OUT |

### Input Table

| Table | Column | Reference for Output Table | Data Type | Description |
|---|---|---|---|---|
| DATA | Columns | DATA_FEATURES | INTEGER, DOUBLE, DECIMAL(p, s), VARCHAR, or NVARCHAR | Attribute data |
| STATISTICS/MODEL | 1st column | | NVARCHAR(256) | Statistic names |
| | 2nd column | | NVARCHAR(5000) | Statistic values |

### Parameter Table

Mandatory Parameter

None.

Optional Parameter

The following parameter is optional. If it is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| HAS_ID | INTEGER | 0 | Specifies whether the input data table has an ID column. If it exists, the ID column must be the first column.<br>• 0: indicates there is no ID column in the data table. The algorithm will impute missing values in all the columns.<br>• 1: indicates that the first column of the data table is ID column. The algorithm treats the ID column as imputation type NONE. |

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| THREAD_RATIO | DOUBLE | 0.0 | Specifies the ratio of total number of threads that can be used by this function. The value range is from 0 to 1, where 0 means only using 1 thread, and 1 means using at most all the currently available threads. Values outside the range are ignored and this function heuristically determines the number of threads to use. |

### Output Tables

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| RESULT | Columns | DATA_FEATURES (in input table) data type | DATA_FEATURES (in input table) column name | Each column type should match the type of the original input data. |
| STATISTICS | 1st column | NVARCHAR(256) | STAT_NAME | Statistic names. |
| | 2nd column | NVARCHAR(5000) | STAT_VALUE | Statistic values. |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

Example: Use the statistics/model table returned in Example 1 of _SYS_AFL.PAL_MISSING_VALUE_HANDLING

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_MISSING_VALUES_DATA_TBL_WITH_ID;
CREATE COLUMN TABLE PAL_MISSING_VALUES_DATA_TBL_WITH_ID ("ID" INTEGER, "V0"
INTEGER, "V1" INTEGER, "V2" VARCHAR(5), "V3" DOUBLE, "V4" DOUBLE, "V5" DOUBLE);
INSERT INTO PAL_MISSING_VALUES_DATA_TBL_WITH_ID VALUES (0, 20,   1,    'B',
NULL, 1.5,  21.7);
INSERT INTO PAL_MISSING_VALUES_DATA_TBL_WITH_ID VALUES (1, 40,   1,    NULL,
0.6,  1.2,  24.3);
INSERT INTO PAL_MISSING_VALUES_DATA_TBL_WITH_ID VALUES (2, NULL, 0,    'D',
NULL, 1.8,  22.6);
INSERT INTO PAL_MISSING_VALUES_DATA_TBL_WITH_ID VALUES (3, 50,   NULL, 'C',
0.7,  1.1,  NULL);
INSERT INTO PAL_MISSING_VALUES_DATA_TBL_WITH_ID VALUES (4, 20,   1,    'A',
0.3,  NULL, 20.6);
DROP TABLE #PAL_PARAMETER_TBL;
```

```
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL ("PARAM_NAME" VARCHAR
(256), "INT_VALUE" INTEGER, "DOUBLE_VALUE" DOUBLE, "STRING_VALUE" VARCHAR
(1000));
INSERT INTO #PAL_PARAMETER_TBL VALUES ('HAS_ID', 1, NULL, NULL);
CALL
"_SYS_AFL"."PAL_MISSING_VALUE_HANDLING_WITH_MODEL"(PAL_MISSING_VALUES_DATA_TBL_WI
TH_ID, PAL_MISSING_VALUES_STATISTICS_TBL, #PAL_PARAMETER_TBL, ?, ?);
```

**Expected Result**

RESULT:

| | ID | V0 | V1 | V2 | V3 | V4 | V5 |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 20 | 1 | B | 0.507692307692307 | 1.5 | 21.7 |
| 2 | 1 | 40 | 1 | A | 0.6 | 1.2 | 24.3 |
| 3 | 2 | 24 | 0 | D | 0.507692307692307 | 1.8 | 22.6 |
| 4 | 3 | 50 | 0 | C | 0.7 | 1.1 | 20.64615384615384 |
| 5 | 4 | 20 | 1 | A | 0.3 | 1.469230769230769 | 20.6 |

STATISTICS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | ID.NUMBER_OF_NULLS | 0 |
| 2 | ID.IMPUTATION_TYPE | NONE |
| 3 | ID.IMPUTED_VALUE | NULL VALUES ARE NOT IMPUTED SINCE IMPUTATION TYPE IS 'NONE' |
| 4 | V0.NUMBER_OF_NULLS | 1 |
| 5 | V0.IMPUTATION_TYPE | SPECIFIED_NUMERICAL_VALUE |
| 6 | V0.IMPUTED_VALUE | 24 |
| 7 | V1.NUMBER_OF_NULLS | 1 |
| 8 | V1.IMPUTATION_TYPE | SPECIFIED_CATEGORICAL_VALUE |
| 9 | V1.IMPUTED_VALUE | 0 |
| 10 | V2.NUMBER_OF_NULLS | 1 |
| 11 | V2.IMPUTATION_TYPE | SPECIFIED_CATEGORICAL_VALUE |
| 12 | V2.IMPUTED_VALUE | A |
| 13 | V3.NUMBER_OF_NULLS | 2 |
| 14 | V3.IMPUTATION_TYPE | SPECIFIED_NUMERICAL_VALUE |
| 15 | V3.IMPUTED_VALUE | 0.507692307692307 |
| 16 | V4.NUMBER_OF_NULLS | 1 |
| 17 | V4.IMPUTATION_TYPE | SPECIFIED_NUMERICAL_VALUE |
| 18 | V4.IMPUTED_VALUE | 1.469230769230769 |
| 19 | V5.NUMBER_OF_NULLS | 1 |
| 20 | V5.IMPUTATION_TYPE | SPECIFIED_NUMERICAL_VALUE |
| 21 | V5.IMPUTED_VALUE | 20.64615384615384 |

## 5.8.9 Time Series Missing Value Handling

This function aims to handle time series missing values in a table.

Missing values in each column of the table can be handled by one of the following imputation types:

NONE: Does nothing. Leave the column untouched.

MODE: Fills all missing values with the mode of the column, that is, the value that appears most often in the column. This type only suits column with categorical variables.

ALLZERO: Fills all missing values with 0. This type only suits column with numerical variables.

MEAN: Fills all missing values with the mean of the column, that is, the average of all non-missing values in the column. This type only suits column with numerical variables.

MEDIAN: Fills all missing values with the median of the column, that is, the median of all non-missing values in the column. This type only suits column with numerical variables.

SPECIFIED_CATEGORICAL_VALUE: Fills all missing values with a specified categorical value. This type only suits column with categorical variables.

SPECIFIED_NUMERICAL_VALUE: Fills all missing values with a specified numerical value. This type only suits column with numerical variables.

SIMPLE_MOVING_AVG: Fills each missing value with the value imputed by simple moving average method. This type only suits column with numerical variables.

LINEAR_MOVING_AVG: Fills each missing value with the value imputed by linear moving average method. This type only suits column with numerical variables.

EXP_MOVING_AVG: Fills each missing value with the value imputed by exponential moving average method. This type only suits column with numerical variables.

LINEAR_INTERP: Fills each missing value with the value imputed by linear interpolation method. This type only suits column with numerical variables.

SPLINE_INTERP: Fills each missing value with the value imputed by spline interpolation method. This type only suits column with numerical variables.

SEADEC: Fills each missing value with the value imputed by using seasonally decomposed time series imputation method. This method first removes the seasonal part from the time series, imputes the deseasonalized series by base imputation method afterwards applies the seasonal part again. This type only suits column with numerical variables. And for time series with seasonality, this method is recommended.

LOCF: Last Observation Carried Forward. The missing value is filled by the last observed non-missing value. This type only suits column with numerical variables.

NOCB: Next Observation Carried Backward. The missing value is filled by the next observed non-missing value. This type only suits column with numerical variables.

By default, columns of integer type and double type are treated as numerical column, columns of string type are treated as categorical column. Columns of integer type can also be indicated as categorical column by explicitly setting as CATEGORICAL_VARIABLE. Besides, if a column of integer type is set with a categorical only imputation type (MODE or SPECIFIED_CATEGORICAL_VALUE), it will be treated as a categorical column automatically.

The output statistics table can serve as a model to handle missing values in new tables with procedure _SYS_AFL.PAL_IMPUTE_TIME_SERIES_WITH_MODEL. In general, the new tables to be handled must have the same column structure (number of columns, column names, and column types) with the table with which the model is trained.

## Prerequisites

None.

## _SYS_AFL.PAL_IMPUTE_TIME_SERIES

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<RESULT table>` | OUT |
| 3 | `<STATISTICS/MODEL table>` | OUT |

### Input Table

| Table | Column | Data Type | Description | Constraint |
|---|---|---|---|---|
| DATA | 1st column | INTEGER, SECOND-DATE, or DATE | Time stamp | This must be the first column. |
| | Feature columns | INTEGER, DOUBLE, DECIMAL(p, s), VAR-CHAR, or NVARCHAR | Features data | |

Parameter Table

Mandatory Parameters

None.

Optional parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| THREAD_RATIO | DOUBLE | 1 | The ratio of available threads.<br>• 0: single thread<br>• 0~1: percentage<br>• Others: heuristically determined | |
| IMPUTATION_TYPE | INTEGER | 2 | Chooses the overall default imputation type for all columns:<br>• 0: NONE<br>• 1: MODE_ALL-ZERO<br>• 2: MODE_MEAN<br>• 3: MODE_MEDIAN<br>• 4: MODE_SMA<br>• 5: MODE_LMA<br>• 6: MODE_EMA<br>• 7: MODE_LIN-TERP<br>• 8: MODE_SIN-TERP<br>• 9: MODE_SEADEC<br>• 10: MODE_LOCF<br>• 11: MODE_NOCB | |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| `<column name>`_IMPUTATION_TYPE | INTEGER | No default value | Sets imputation type of a specified column. This value overwrites the default imputation type deduced by the IMPUTATION_TYPE parameter:<br>• 0: NONE<br>• 1: MODE<br>• 2: ALLZERO<br>• 3: MEAN<br>• 4: MEDIAN<br>• 5: SPECIFIED_CATEGORICAL_VALUE<br>• 6: SPECIFIED_NUMERICAL_VALUE<br>• 7: SIMPLE_MOVING_AVG<br>• 8: LINEAR_MOVING_AVG<br>• 9: EXP_MOVING_AVG<br>• 10: LINEAR_INTERP<br>• 11: SPLINE_INTERP<br>• 12: SEADEC<br>• 13: LOCF<br>• 14: NOCB | |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| BASE_ALGORITHM | INTEGER | 1 | Chooses the base imputation method when imputation type is SEADEC:<br>• 0: ALLZERO<br>• 1: MEAN<br>• 2: MEDIAN<br>• 3: SIMPLE_MOVING_AVG<br>• 4: LINEAR_MOVING_AVG<br>• 5: EXP_MOVING_AVG<br>• 6: LINEAR_INTERP<br>• 7: SPLINE_INTERP<br>• 8: LOCF<br>• 9: NOCB | Valid only for columns when `<column name>_IMPUTATION_TYPE` is `SEADEC` or for numerical columns when `IMPUTATION_TYPE` is `MODE_SEADEC`. |
| ALPHA | DOUBLE | 0.2 | The criterion for the autocorrelation coefficient. The value range is (0, 1). A larger value indicates stricter requirement for seasonality. | Valid only for columns when `<column name>_IMPUTATION_TYPE` is `SEADEC` or for numerical columns when `IMPUTATION_TYPE` is `MODE_SEADEC`. |
| EXTRAPOLATION | INTEGER | 0 | Specifies whether to extrapolate the end-points.<br>• 0: No<br>• 1: Yes | Valid only for columns when `<column name>_IMPUTATION_TYPE` is `SEADEC` or for numerical columns when `IMPUTATION_TYPE` is `MODE_SEADEC`. Set to 1 when there's an end-point issue. |
| SMOOTH_WIDTH | INTEGER | 0 | Specifies the width of the moving average applied to non-seasonal data. 0 indicates linear fitting to extract trends. | Valid only for columns when `<column name>_IMPUTATION_TYPE` is `SEADEC` or for numerical columns when `IMPUTATION_TYPE` is `MODE_SEADEC`. Can not be larger than half of the data length. |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| AUXILIARY_NORMALITYTEST | INTEGER | 0 | Specifies whether to use normality test to identify model types.<br>• 0: No<br>• 1: Yes | Valid only for columns when `<column name>_IMPUTATION_TYPE` is `SEADEC` or for numerical columns when `IMPUTATION_TYPE` is `MODE_SEADEC`. |
| CATEGORICAL_VARIABLE | NVARCHAR | No default value | Indicates whether a column of INTEGER type is corresponding to a categorical variable. By default, columns of VARCHAR or NVARCHAR type are treated as categorical variable, columns of INTEGER or DOUBLE type are treated as numerical variable. | |

### Output table

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| RESULT | 1st column | ID (in input table) data type | ID (in input table) column name | ID |
| | Feature columns | Feature Columns (in input table) data type | Feature columns (in input table) column name | Each column type must match the type of the original input data. |
| STATISTICS/MODEL | 1st column | NVARCHAR(256) | STAT_NAME | Statistic names. |
| | 2nd column | NVARCHAR(5000) | STAT_VALUE | Statistic values. |

### Example 1

Assume that:

- DM_PAL is a schema belonging to USER1;
- USER1 has been assigned the AFL__SYS_AFL_AFLPAL_EXECUTE or AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION role.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_TS_DATA_TBL;
CREATE COLUMN TABLE PAL_TS_DATA_TBL (
    "TIME_STAMP" INTEGER,
    "TS0" INTEGER,
    "TS1" INTEGER,
    "TS2" DOUBLE
);
INSERT INTO PAL_TS_DATA_TBL VALUES (0, 10, 0, 23.6);
INSERT INTO PAL_TS_DATA_TBL VALUES (1, 20, 1, 21.8);
INSERT INTO PAL_TS_DATA_TBL VALUES (2, 50, 1, 21.9);
```

```
INSERT INTO PAL_TS_DATA_TBL VALUES (3, 30, NULL, 22.6);
INSERT INTO PAL_TS_DATA_TBL VALUES (4, 10, 0, NULL);
INSERT INTO PAL_TS_DATA_TBL VALUES (5, 10, 0, 19.7);
INSERT INTO PAL_TS_DATA_TBL VALUES (6, NULL, 0, 17.8);
INSERT INTO PAL_TS_DATA_TBL VALUES (7, 10, 1, 24.9);
INSERT INTO PAL_TS_DATA_TBL VALUES (8, 20, NULL, 22.2);
INSERT INTO PAL_TS_DATA_TBL VALUES (9, 30, 1, NULL);
INSERT INTO PAL_TS_DATA_TBL VALUES (10, 50, 1, 16.4);
INSERT INTO PAL_TS_DATA_TBL VALUES (11, NULL, 1, 19.3);
INSERT INTO PAL_TS_DATA_TBL VALUES (12, 30, 0, 21.7);
INSERT INTO PAL_TS_DATA_TBL VALUES (13, 30, 1, NULL);
INSERT INTO PAL_TS_DATA_TBL VALUES (14, NULL, 0, 18.6);
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL (
"PARAM_NAME" VARCHAR (256),
"INT_VALUE" INTEGER,
"DOUBLE_VALUE" DOUBLE,
"STRING_VALUE" VARCHAR (1000));
INSERT INTO PAL_PARAMETER_TBL VALUES('IMPUTATION_TYPE', 2, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES('CATEGORICAL_VARIABLE', NULL, NULL, 'TS1');
--INSERT INTO PAL_PARAMETER_TBL VALUES('TS1_IMPUTATION_TYPE', 5, NULL, '2');
--INSERT INTO PAL_PARAMETER_TBL VALUES('TS2_IMPUTATION_TYPE', 6, 20.0, NULL);
DROP TABLE PAL_IMPUTE_TS_STATISTICS_TBL;
CREATE COLUMN TABLE PAL_IMPUTE_TS_STATISTICS_TBL ("STAT_NAME" NVARCHAR(256),
"STAT_VALUE" NVARCHAR(5000));
do begin
    lt_data = select * from PAL_TS_DATA_TBL;
    lt_control = select * from PAL_PARAMETER_TBL;
    CALL _SYS_AFL.PAL_IMPUTE_TIME_SERIES(:lt_data, :lt_control, lt_res,
lt_stats);
    INSERT INTO PAL_IMPUTE_TS_STATISTICS_TBL SELECT * FROM :lt_stats;
    SELECT * FROM :lt_res;
    SELECT * FROM PAL_IMPUTE_TS_STATISTICS_TBL;
end;
```

### Expected Result

RESULT:

| | TIME_STAMP | TS0 | TS1 | TS2 |
|---|---|---|---|---|
| 1 | 0 | 10 | 0 | 23.6 |
| 2 | 1 | 20 | 1 | 21.8 |
| 3 | 2 | 50 | 1 | 21.9 |
| 4 | 3 | 30 | 1 | 22.6 |
| 5 | 4 | 10 | 0 | 20.8... |
| 6 | 5 | 10 | 0 | 19.7 |
| 7 | 6 | 25 | 0 | 17.8 |
| 8 | 7 | 10 | 1 | 24.9 |
| 9 | 8 | 20 | 1 | 22.2 |
| 10 | 9 | 30 | 1 | 20.8... |
| 11 | 10 | 50 | 1 | 16.4 |
| 12 | 11 | 25 | 1 | 19.3 |
| 13 | 12 | 30 | 0 | 21.7 |
| 14 | 13 | 30 | 1 | 20.8... |
| 15 | 14 | 25 | 0 | 18.6 |

STATISTICS/MODEL:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | TIME_STAMP.IMPUTATION_TYPE | NONE |
| 2 | TIME_STAMP.NUMBER_OF_NULLS | 0 |
| 3 | TS0.IMPUTATION_TYPE | MEAN |
| 4 | TS0.IMPUTED_VALUE | 25 |
| 5 | TS0.NUMBER_OF_NULLS | 3 |
| 6 | TS1.IMPUTATION_TYPE | MODE |
| 7 | TS1.IMPUTED_VALUE | 1 |
| 8 | TS1.NUMBER_OF_NULLS | 2 |
| 9 | TS2.IMPUTATION_TYPE | MEAN |
| 10 | TS2.IMPUTED_VALUE | 20.875 |
| 11 | TS2.NUMBER_OF_NULLS | 3 |

### Example 2

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_SEASONALITY_DATA_TBL;
CREATE COLUMN TABLE PAL_SEASONALITY_DATA_TBL (
    "TIME_STAMP" INTEGER,
    "SERIES" DOUBLE
);
INSERT INTO PAL_SEASONALITY_DATA_TBL VALUES(0   , NULL);
INSERT INTO PAL_SEASONALITY_DATA_TBL VALUES(1   , 34.72);
INSERT INTO PAL_SEASONALITY_DATA_TBL VALUES(2   , 10.34);
INSERT INTO PAL_SEASONALITY_DATA_TBL VALUES(3   , -12.89);
INSERT INTO PAL_SEASONALITY_DATA_TBL VALUES(4   , NULL);
INSERT INTO PAL_SEASONALITY_DATA_TBL VALUES(5   , 57.3);
INSERT INTO PAL_SEASONALITY_DATA_TBL VALUES(6   , 6.735);
INSERT INTO PAL_SEASONALITY_DATA_TBL VALUES(7   , -19.365);
INSERT INTO PAL_SEASONALITY_DATA_TBL VALUES(8  , NULL);
INSERT INTO PAL_SEASONALITY_DATA_TBL VALUES(9  , 52.455);
INSERT INTO PAL_SEASONALITY_DATA_TBL VALUES(10  , 8.455);
INSERT INTO PAL_SEASONALITY_DATA_TBL VALUES(11  , 13.595);
INSERT INTO PAL_SEASONALITY_DATA_TBL VALUES(12  , NULL);
INSERT INTO PAL_SEASONALITY_DATA_TBL VALUES(13  , 54.81);
INSERT INTO PAL_SEASONALITY_DATA_TBL VALUES(14  , 4.625);
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL (
"PARAM_NAME" VARCHAR(100),
"INT_VALUE" INTEGER,
"DOUBLE_VALUE" DOUBLE,
"STRING_VALUE" VARCHAR(100)
);
INSERT INTO PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 0.5, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES('IMPUTATION_TYPE', 9, NULL, NULL);
call _SYS_AFL.PAL_IMPUTE_TIME_SERIES(PAL_SEASONALITY_DATA_TBL,
PAL_PARAMETER_TBL, ?, ?);
```

### Expected Result

RESULT:

| | TIME_STAMP | SERIES |
|---|---|---|
| 1 | 0 | 62.28630208333333 |
| 2 | 1 | 34.72 |
| 3 | 2 | 10.34 |
| 4 | 3 | -12.89 |
| 5 | 4 | 25.391302083333... |
| 6 | 5 | 57.3 |
| 7 | 6 | 6.735 |
| 8 | 7 | -19.365 |
| 9 | 8 | 19.731302083333... |
| 10 | 9 | 52.455 |
| 11 | 10 | 8.455 |
| 12 | 11 | 13.595 |
| 13 | 12 | 37.38880208333334 |
| 14 | 13 | 54.81 |
| 15 | 14 | 4.625 |

STATISTICS/MODEL:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | TIME_STAMP.IMPUTATION_TYPE | NONE |
| 2 | TIME_STAMP.NUMBER_OF_NULLS | 0 |
| 3 | SERIES.IMPUTATION_TYPE | SEADEC |
| 4 | SERIES.SEASONALITY | additive |
| 5 | SERIES.PERIOD | 4 |
| 6 | SERIES.SEASONALITY_PARAMET... | {"alpha":0.2,"extrapolation":false,"smooth_widt... |
| 7 | SERIES.NUMBER_OF_NULLS | 4 |

## _SYS_AFL.PAL_IMPUTE_TIME_SERIES_WITH_MODEL

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<STATISTICS/MODEL table>` | IN |
| 3 | `<PARAMETER table>` | IN |
| 4 | `<RESULT table>` | OUT |
| 4 | `<STATISTICS table>` | OUT |

### Input Table

| Table | Column | Data Type | Description | Constraint |
|---|---|---|---|---|
| DATA | 1st column | INTEGER, SECOND-DATE, or DATE | Time Stamp | This must be the first column. |

| Table | Column | Data Type | Description | Constraint |
|---|---|---|---|---|
| | Feature columns | INTEGER, DOUBLE, DECIMAL(p, s), VARCHAR, or NVARCHAR | Features data | |
| STATISTICS/MODEL | 1st column | | NVARCHAR(256) | Statistic names. |
| | 2nd column | | NVARCHAR(5000) | Statistic values. |

Parameter Table

Mandatory Parameters

None.

Optional parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| THREAD_RATIO | DOUBLE | 1 | The ratio of available threads.<br>• 0: single thread<br>• 0~1: percentage<br>• Others: heuristically determined | |

### Output table

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| RESULT | 1st column | ID (in input table) data type | ID (in input table) column name | ID |
| | Feature columns | Feature Columns (in input table) data type | Feature Columns (in input table) column name | Each column type must match the type of the original input data. |
| STATISTICS | 1st column | NVARCHAR(256) | STAT_NAME | Statistic names. |
| | 2nd column | NVARCHAR(5000) | STAT_VALUE | Statistic values. |

### Example

Assume that:

- DM_PAL is a schema belonging to USER1;
- USER1 has been assigned the AFL__SYS_AFL_AFLPAL_EXECUTE or AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION role.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_TS_DATA_W_TBL;
CREATE COLUMN TABLE PAL_TS_DATA_W_TBL (
    "TIME_STAMP" INTEGER,
    "TS0" INTEGER,
    "TS1" INTEGER,
```

```
    "TS2" DOUBLE
);
INSERT INTO PAL_TS_DATA_W_TBL VALUES (0, 40, 1, 24.3);
INSERT INTO PAL_TS_DATA_W_TBL VALUES (1, 20, 1, 22.6);
INSERT INTO PAL_TS_DATA_W_TBL VALUES (2, NULL, 0, 23.8);
INSERT INTO PAL_TS_DATA_W_TBL VALUES (3, 18, NULL, 20.6);
INSERT INTO PAL_TS_DATA_W_TBL VALUES (4, 14, 1, NULL);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
"PARAM_NAME" VARCHAR (256),
"INT_VALUE" INTEGER,
"DOUBLE_VALUE" DOUBLE,
"STRING_VALUE" VARCHAR (1000));
INSERT INTO #PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 0.5, NULL);
CALL "_SYS_AFL"."PAL_IMPUTE_TIME_SERIES_WITH_MODEL"(PAL_TS_DATA_W_TBL,
PAL_IMPUTE_TS_STATISTICS_TBL, #PAL_PARAMETER_TBL, ?, ?);
```

**Expected Result**

RESULT:

| | TIME_STAMP | TS0 | TS1 | TS2 |
|---|---|---|---|---|
| 1 | 0 | 40 | 1 | 24.3 |
| 2 | 1 | 20 | 1 | 22.6 |
| 3 | 2 | 25 | 0 | 23.8 |
| 4 | 3 | 18 | 1 | 20.6 |
| 5 | 4 | 14 | 1 | 20.8... |

STATISTICS/MODEL:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | TIME_STAMP.IMPUTATION_TYPE | NONE |
| 2 | TIME_STAMP.NUMBER_OF_NULLS | 0 |
| 3 | TS0.IMPUTATION_TYPE | SPECIFIED_NUMERICAL_VALUE |
| 4 | TS0.IMPUTED_VALUE | 25 |
| 5 | TS0.NUMBER_OF_NULLS | 1 |
| 6 | TS1.IMPUTATION_TYPE | SPECIFIED_CATEGORICAL_VALUE |
| 7 | TS1.IMPUTED_VALUE | 1 |
| 8 | TS1.NUMBER_OF_NULLS | 1 |
| 9 | TS2.IMPUTATION_TYPE | SPECIFIED_NUMERICAL_VALUE |
| 10 | TS2.IMPUTED_VALUE | 20.875 |
| 11 | TS2.NUMBER_OF_NULLS | 1 |

## 5.8.10 Multidimensional Scaling (MDS)

This function serves as a tool for dimensional reduction or data visualization.

The function embeds the samples in N-dimension in a lower K-dimensional space by applying a non-linear transformation – classical multidimensional scaling. The characteristic of this transformation is that it is able, or does the best it could, to preserve the distances between entities after reducing to a lower dimension.

There are two kinds of input formats supported by this function: a N×N dissimilarity matrix, or a usual entity – feature matrix. The former is a symmetric matrix, with each element representing the distance (dissimilarity) between two entities. The later can be converted to a dissimilarity matrix using a method specified by the user.

The computation is straightforward: the dissimilarity matrix is firstly squared and double centered, then going through an eigen-decomposition procedure. In the end the lower dimensional embedding is calculated from

the eigen-values and eigen-vectors. It is worth noting that the lowest dimension that the classical scaling can obtain is the largest number of the positive eigen-value. Therefore, in some cases the user-specified K is not able to be achieved.

For the sake of compactness, this function outputs the resulting matrix in the row-major order. The user can readily decide the value of K by examining the proportion of variation explained reported by this function, which is calculated as

$$\frac{\sum_{i=1}^{K}\lambda_i}{\sum_{i=1}^{n-1}|\lambda_i|},$$

where $\lambda_i$ is the eigen-value, K is the number of used dimension, and n is the number of total entities.

## Prerequisites

- The input data should be all numeric, and no categorical variable is allowed.
- The input data should be a symmetric matrix if the input type is specified as a dissimilarity matrix.
- The input data does not contain null value.
- The number of valid input data should be at least equal to the value of the K parameter.

## _SYS_AFL.PAL_MDS

This function finds the lower dimensional embedding of a high dimensional data using classical multidimensional scaling.

### Overview of all Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<INPUT table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<RESULT table>` | OUT |
| 4 | `<STATISTICS table>` | OUT |

### Input Tables

| Table | Column | Reference for Output Table | Data Type | Description |
|---|---|---|---|---|
| INPUT | 1st column | DATA_ID | INTEGER, VARCHAR, or NVARCHAR | ID |

| Table | Column | Reference for Output Table | Data Type | Description |
|---|---|---|---|---|
| | Feature columns | DATA_FEATURES | INTEGER, DOUBLE, or DECIMAL(p,s) | Attribute data |

### Parameter Table

Mandatory Parameters

The following parameter is mandatory and must be given a value.

| Name | Data Type | Description |
|---|---|---|
| INPUT_TYPE | INTEGER | The type of the input table:<br>• 0: Observation – feature matrix<br>• 1: Dissimilarity matrix |

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| THREAD_RATIO | DOUBLE | 0 | Specifies the ratio of total number of threads that can be used by this function. The value range is from 0 to 1, where 0 means only using 1 thread, and 1 means using at most all the currently available threads. Values outside the range will be ignored and this function heuristically determines the number of threads to use. | |
| K | INTEGER | 2 | The number of dimension that the input dataset is to be reduced to. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| DISTANCE_LEVEL | INTEGER | 2 | The type of distance during the calculation of dissimilarity matrix:<br>• 1: Manhattan distance<br>• 2: Euclidean distance<br>• 3: Minkowski distance | Only valid if INPUT_TYPE=0 |
| MINKOWSKI_POWER | DOUBLE | 3 | When you use the Minkowski distance, this parameter controls the value of power. | Only valid if INPUT_TYPE=0 and DISTANCE_LEVEL=3 |

### Output Tables

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| RESULT | 1st column | DATA_ID (in input table) data type | DATA_ID (in input table) column name | ID |
| | 2nd column | INTEGER | DIMENSION | Dimension |
| | 1st column | DOUBLE | VALUE | Value |
| STATISTICS | 2nd column | NVARCHAR(100) | STAT_NAME | Name of statistics |
| | 3rd column | DOUBLE | STAT_VALUE | Value of statistics |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA DM_PAL;
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE
    #PAL_PARAMETER_TBL
    ("PARAM_NAME" VARCHAR(256), "INT_VALUE" INTEGER, "DOUBLE_VALUE" DOUBLE,
"STRING_VALUE" VARCHAR(1000));
INSERT INTO #PAL_PARAMETER_TBL VALUES ('INPUT_TYPE',1,NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('K',2,NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('THREAD_RATIO',NULL,0.5,NULL);
DROP TABLE PAL_MDS_DATA_TBL;
CREATE COLUMN TABLE PAL_MDS_DATA_TBL ("ID" varchar(50), "X1" DOUBLE, "X2" DOUBLE,
    "X3" DOUBLE, "X4" DOUBLE);;
```

```
INSERT INTO PAL_MDS_DATA_TBL VALUES (1, 0.0000000, 0.9047814, 0.9085961,
0.9103063);
INSERT INTO PAL_MDS_DATA_TBL VALUES (2, 0.9047814, 0.0000000, 0.2514457,
0.5975016);
INSERT INTO PAL_MDS_DATA_TBL VALUES (3, 0.9085961, 0.2514457, 0.0000000,
0.4403572);
INSERT INTO PAL_MDS_DATA_TBL VALUES (4, 0.9103063, 0.5975016, 0.4403572,
0.0000000);
CALL _SYS_AFL.PAL_MDS(PAL_MDS_DATA_TBL,"#PAL_PARAMETER_TBL", ?, ?);
```

**Expected Result**

RESULT:

| | ID | DIMENSION | VALUE |
|---|---|---|---|
| 1 | 1 | 1 | 0.6519174131067575 |
| 2 | 1 | 2 | -0.015858606078213176 |
| 3 | 2 | 1 | -0.2177371579016654 |
| 4 | 2 | 2 | -0.2531945605576968 |
| 5 | 3 | 1 | -0.24990695105980829 |
| 6 | 3 | 2 | -0.07294967855404891 |
| 7 | 4 | 1 | -0.18427330414528353 |
| 8 | 4 | 2 | 0.34200284518995866 |

STATISTICS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | acheived K | 2 |
| 2 | proportion of variation explaind | 0.9789012732023387 |

## 5.8.11 Partition

The algorithm partitions an input dataset randomly into three disjoints subsets called training, testing, and validation set. The proportion of each subset is defined as a parameter. Let us remark that the union of these three subsets might not be the complete initial dataset.

Two different partitions can be obtained:

1. **Random Partition**, which randomly divides all the data.
2. **Stratified Partition**, which divides each subpopulation randomly.

In the second case, the dataset needs to have at least one categorical attribute (for example, of type VARCHAR). The initial dataset will first be subdivided according to the different categorical values of this attribute. Each mutually exclusive subset will then be randomly split to obtain the training, testing, and validation subsets. This ensures that all "categorical values" or "strata" will be present in the sampled subset.

## Prerequisites

- There is no missing or null data in the column used for stratification.
- The column used for stratification must be categorical (INTEGER, VARCHAR, or NVARCHAR).

## _SYS_AFL.PAL_PARTITION

This function reads the input data and generates training, testing, and validation data with the partition algorithm.

### Overview of all Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<OUTPUT table>` | OUT |

### Input Table

| Table | Column | Reference for Output Table | Data Type | Description |
|---|---|---|---|---|
| DATA | 1st column | DATA_ID | INTEGER, VARCHAR, or NVARCHAR | Key of input data table |
| | Feature columns | | INTEGER, VARCHAR, NVARCHAR, DOUBLE, or DECIMAL(p, s) | Data columns.<br>The column used for stratification must be categorical (INTEGER, VARCHAR, or NVARCHAR). |

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| THREAD_RATIO | DOUBLE | 0 | Specifies the ratio of total number of threads that can be used by this function. The value range is from 0 to 1, where 0 means only using 1 thread, and 1 means using at most all the currently available threads. Values outside the range will be ignored and this function heuristically determines the number of threads to use. | |
| PARTITION_METHOD | INTEGER | 0 | Indicates the partition method:<br>• 0: Random partition<br>• Not 0: Stratified partition | |
| RANDOM_SEED | INTEGER | 0 | Indicates the seed used to initialize the random number generator.<br>• 0: Uses the system time<br>• Not 0: Uses the specified seed | |
| STRATIFIED_COLUMN | VARCHAR | No default value | Indicates which column is used for stratification. | Valid only when PARTITION_METHOD is set to a non-zero value (stratified partition). |
| TRAINING_PERCENT | DOUBLE | 0.8 | The percentage of training data.<br>Value range: 0 ≤ value ≤1 | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| TESTING_PERCENT | DOUBLE | 0.1 | The percentage of testing data.<br>Value range: 0 ≤ value ≤1 | |
| VALIDATION_PERCENT | DOUBLE | 0.1 | The percentage of validation data.<br>Value range: 0 ≤ value ≤1 | |
| TRAINING_SIZE | INTEGER | No default value | Row size of training data.<br>Value range: ≥ 0 | If both TRAINING_PERCENT and TRAINING_SIZE are specified, TRAINING_PERCENT takes precedence. |
| TESTING_SIZE | INTEGER | No default value | Row size of testing data.<br>Value range: ≥ 0 | If both TESTING_PERCENT and TESTING_SIZE are specified, TESTING_PERCENT takes precedence. |
| VALIDATION_SIZE | INTEGER | No default value | Row size of validation data.<br>Value range: ≥ 0 | If both VALIDATION_PERCENT and VALIDATION_SIZE are specified, VALIDATION_PERCENT takes precedence. |

## Output Table

| Table | Column | Data Type | Column | Description |
|---|---|---|---|---|
| OUTPUT | 1st column | DATA_ID (in input table) data type | DATA_ID (in input table) column name | ID. Key of the data table. |
| | 2nd column | INTEGER | PARTITION_TYPE | Indicates which partition the table belongs to.<br>• 1: training data<br>• 2: testing data<br>• 3: validation data |

## Example

Assume that:

- DM_PAL is a schema belonging to USER1;
- USER1 has been assigned the AFL__SYS_AFL_AFLPAL_EXECUTE or AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION role.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_PARTITION_DATA_TBL;
CREATE COLUMN TABLE PAL_PARTITION_DATA_TBL(
     "ID" INTEGER,
     "HomeOwner" VARCHAR (100),
     "MaritalStatus" VARCHAR (100),
     "AnnualIncome" DOUBLE,
     "DefaultedBorrower" VARCHAR (100)
);
INSERT INTO PAL_PARTITION_DATA_TBL VALUES (0,  'YES', 'Single',   125, 'NO');
INSERT INTO PAL_PARTITION_DATA_TBL VALUES (1,  'NO',  'Married',  100, 'NO');
INSERT INTO PAL_PARTITION_DATA_TBL VALUES (2,  'NO',  'Single',    70, 'NO');
INSERT INTO PAL_PARTITION_DATA_TBL VALUES (3,  'YES', 'Married',  120, 'NO');
INSERT INTO PAL_PARTITION_DATA_TBL VALUES (4,  'NO',  'Divorced',  95, 'YES');
INSERT INTO PAL_PARTITION_DATA_TBL VALUES (5,  'NO',  'Married',   60, 'NO');
INSERT INTO PAL_PARTITION_DATA_TBL VALUES (6,  'YES', 'Divorced', 220, 'NO');
INSERT INTO PAL_PARTITION_DATA_TBL VALUES (7,  'NO',  'Single',    85, 'YES');
INSERT INTO PAL_PARTITION_DATA_TBL VALUES (8,  'NO',  'Married',   75, 'NO');
INSERT INTO PAL_PARTITION_DATA_TBL VALUES (9,  'NO',  'Single',    90, 'YES');
INSERT INTO PAL_PARTITION_DATA_TBL VALUES (10, 'YES', 'Single',   125, 'NO');
INSERT INTO PAL_PARTITION_DATA_TBL VALUES (11, 'NO',  'Married',  100, 'NO');
INSERT INTO PAL_PARTITION_DATA_TBL VALUES (12, 'NO',  'Single',    70, 'NO');
INSERT INTO PAL_PARTITION_DATA_TBL VALUES (13, 'YES', 'Married',  120, 'NO');
INSERT INTO PAL_PARTITION_DATA_TBL VALUES (14, 'NO',  'Divorced',  95, 'YES');
INSERT INTO PAL_PARTITION_DATA_TBL VALUES (15, 'NO',  'Married',   60, 'NO');
INSERT INTO PAL_PARTITION_DATA_TBL VALUES (16, 'YES', 'Divorced', 220, 'NO');
INSERT INTO PAL_PARTITION_DATA_TBL VALUES (17, 'NO',  'Single',    85, 'YES');
INSERT INTO PAL_PARTITION_DATA_TBL VALUES (18, 'NO',  'Married',   75, 'NO');
INSERT INTO PAL_PARTITION_DATA_TBL VALUES (19, 'NO',  'Single',    90, 'YES');
INSERT INTO PAL_PARTITION_DATA_TBL VALUES (20, 'YES', 'Single',   125, 'NO');
INSERT INTO PAL_PARTITION_DATA_TBL VALUES (21, 'NO',  'Married',  100, 'NO');
INSERT INTO PAL_PARTITION_DATA_TBL VALUES (22, 'NO',  'Single',    70, 'NO');
INSERT INTO PAL_PARTITION_DATA_TBL VALUES (23, 'YES', 'Married',  120, 'NO');
INSERT INTO PAL_PARTITION_DATA_TBL VALUES (24, 'NO',  'Divorced',  95, 'YES');
INSERT INTO PAL_PARTITION_DATA_TBL VALUES (25, 'NO',  'Married',   60, 'NO');
INSERT INTO PAL_PARTITION_DATA_TBL VALUES (26, 'YES', 'Divorced', 220, 'NO');
INSERT INTO PAL_PARTITION_DATA_TBL VALUES (27, 'NO',  'Single',    85, 'YES');
INSERT INTO PAL_PARTITION_DATA_TBL VALUES (28, 'NO',  'Married',   75, 'YES');
INSERT INTO PAL_PARTITION_DATA_TBL VALUES (29, 'NO',  'Single',    90, 'YES');
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PARTITION_METHOD',0,null,null);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('RANDOM_SEED',23,null,null);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('TRAINING_PERCENT', null,0.6,null);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('TESTING_PERCENT', null,0.2,null);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('VALIDATION_PERCENT', null,0.2,null);
CALL "_SYS_AFL"."PAL_PARTITION"(PAL_PARTITION_DATA_TBL, #PAL_PARAMETER_TBL, ?);
```

### Expected Result

| | ID | PARTITION_TYPE |
|---|---|---|
| 1 | 1 | 1 |
| 2 | 25 | 1 |
| 3 | 0 | 1 |
| 4 | 12 | 1 |
| 5 | 28 | 1 |
| 6 | 17 | 1 |
| 7 | 21 | 1 |
| 8 | 27 | 1 |
| 9 | 29 | 1 |
| 10 | 24 | 1 |
| 11 | 18 | 1 |
| 12 | 23 | 1 |
| 13 | 7 | 1 |
| 14 | 22 | 1 |
| 15 | 10 | 1 |
| 16 | 3 | 1 |
| 17 | 13 | 1 |
| 18 | 5 | 1 |
| 19 | 26 | 2 |
| 20 | 16 | 2 |
| 21 | 14 | 2 |
| 22 | 9 | 2 |
| 23 | 2 | 2 |
| 24 | 11 | 2 |
| 25 | 8 | 3 |
| 26 | 6 | 3 |
| 27 | 19 | 3 |
| 28 | 15 | 3 |
| 29 | 20 | 3 |
| 30 | 4 | 3 |

> **Note**
>
> The wrapper procedure generation approach to call PAL functions is still supported. For more information, refer to the older versions (prior to SAP HANA Platform 2.0 SPS 02) of this document.

## 5.8.12 Power Transform

Power Transform is the family of non-linear transform methods using the power function to transform continuous data. The transformation can stabilize the variance of data, minimize skewness, and approximate its distribution as Gaussian.

Gaussian distributed data is usually assumed in many machine learning algorithms. However, the data obtained may not conform to the Gaussian distribution. Thus a non-linear transformer, like Power Transform,

can be used to make data more Gaussian-like. The effect after transformation depends on the actual distribution of original data.

Power Transform supports Box-Cox transformation and Yeo-Johnson transformation. Both transformations are monotonically increasing functions with one hyper-parameter, normally named λ. The main difference lies in the domain of two transformation functions. While Box-Cox transformation has definitions only on strict positive data, Yeo-Johnson transformation has definitions on the whole real domain. For Yeo-Johnson transformation, when restricting the data to positive only, it is similar to Box-Cox transformation.

Box-Cox transformation is defined by the following function ($x^{(\lambda)}$ represents the value after transformation):

$$x^{(\lambda)}=\begin{cases}\frac{x^{\lambda}-1}{\lambda} & \text{if } \lambda\neq 0\\ \ln(x) & \text{if } \lambda=0\end{cases}$$

This function requires x to be strictly positive. Hyper-parameter λ needs to be fitted with user provided data, by maximizing log-likelihood of transformed data conforming to the Gaussian distribution.

Yeo-Johnson transformation is defined by the following function ($x^{(\lambda)}$ represents the value after transformation):

$$x^{(\lambda)}=\begin{cases}\frac{(x+1)^{\lambda}-1}{\lambda} & \text{if } \lambda\neq 0,\ x\geq 0\\ \ln(x+1) & \text{if } \lambda=0,\ x\geq 0\\ -\frac{(-x+1)^{(2-\lambda)}-1}{2-\lambda} & \text{if } \lambda\neq 2,\ x<0\\ -\ln(-x+1) & \text{if } \lambda=2,\ x<0\end{cases}$$

This transformation removes the restriction on user provided data. It can be any real data. Hyper-parameter λ can be fitted by maximizing log-likelihood of transformed data conforming to the Gaussian distribution.

PAL provides two stored procedures for the Power Transform algorithm:

- `_SYS_AFL.PAL_POWER_TRANSFORMER_FIT` takes training data and configurations as input and outputs the fitted λ and optional standardization related statistics.
- `_SYS_AFL.PAL_POWER_TRANSFORMER_TRANSFORM` takes prediction data and fitted parameters as input and outputs the transformed result for each row of prediction data.

## _SYS_AFL.PAL_POWER_TRANSFORMER_FIT

### Prerequisites

- The Data Input table must have an ID column as the first column. It must not have NULL value and no duplicate ID is allowed.
- No categorical data is supported since Power Transform only supports continuous data. Integer type data is also treated as continuous data.
- No NULL value is allowed in any feature columns.
- Empty Data Input table is forbidden.
- Each data column of the Data Input table can not be constant. This implies that the Data Input table should contains at least two rows of data.

This procedure mainly implements the fitting function, plus the optional standardization operation for each feature. Each feature input in Power Transform is independent, so they can be processed in parallel. Basically

for each input feature, the first step is to fit the λ hyper-parameter according to the user input configuration. With the obtained fitted hyper-parameter, the procedure transforms the corresponding feature. Finally, the procedure optionally standardizes the transformed feature, and record mean and standard deviation statistics.

This procedure supports two transformations: Box-Cox Transformation and Yeo-Johnson Transformation. Both transformations have one hyper-parameter λ which needs to be fitted with your input data.

Currently this procedure minimizes negative log-likelihood target function of transformed random variables for each feature to fit this value, assuming it conforms to the Gaussian distribution. You need to specify an interval for each feature to limit the range where the procedure tries to find the best λ value to minimize the target function. There is a default interval specified by `INTERVAL_START` and `INTERVAL_END` for all features in the Data Input table. This default interval can be overridden by the individual interval specified using the `FEATURE_<FEATURE_NAME>_INTERVAL_START` and `FEATURE_<FEATURE_NAME>_INTERVAL_END` parameters.

The interval can serve as a constraint or a hint. The `INTERVAL_HINT` parameter specifies whether the interval is a hint or not. When the interval is used as a constraint, the final fitted λ value is forced to be inside the given interval. Otherwise this interval just serves as the hint for the start interval to probe. The final fitted λ value may or may not fall within the given interval.

Either choice has its pros and cons. When the given interval serves as a constraint, the result is more predictable since it must be inside the interval. But it may not be the best candidate due to the predefined constraint. When the given interval serves as a hint, it is more intelligent to obtain the best λ value, but meanwhile is also prone to fail to get the result due to finite machine precision.

During the process of computing the optimal λ value, sometimes it fails due to finite machine precision, especially for features with low variance statistic or for large absolute λ value. To the extreme, the constant feature is forbidden and it is meaningless to transform a constant feature to Gaussian-like data. Upon failure, PAL raises a message reporting the failed feature. You can adjust the input interval for the corresponding failed feature with feature specific interval configuration, or can try to specify the interval of the failed feature as a hint.

> **Note**
>
> There is a known limitation for the current release. The computed λ value may be sub-optimal. Computation of global optimum is a hard problem and needs much more time and computing resource, so there is a weigh between quality of result and the time and computing resource spent.

The output of this procedure consists of two tables, one for model and the other one for transformed result. The Model table basically stores the feature name, fitted value, and optional mean and standard deviation value for each feature. It uses UTF-8 escaped JSON format. The Transformed Result output table may be empty, depending on the `OUTPUT_TRANSFORM` configuration. Each value is obtained by applying the power transformation and optional standardization, depending on the `STANDARDIZE` configuration.

**Overview of All Tables**

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | DATA table | IN |
| 2 | PARAMETER table | IN |
| 3 | MODEL table | OUT |
| 4 | TRANSFORMED_RESULT table | OUT |

### Input Table

Data Table

| Column | Reference for Output Table | Data Type | Description |
|---|---|---|---|
| 1st column | ID | INTEGER, VARCHAR, NVARCHAR, BIGINT | ID data, duplicated values in this column is not allowed. |
| Feature columns | FEATURES | INTEGER, DOUBLE, DECIMAL(p, s) | Feature data, the variance for each column cannot be 0. |

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| THREAD_RATIO | DOUBLE | 1.0 | Specifies the ratio of total number of threads that can be used by this function. The value range is from 0 to 1, where 0 means only using 1 thread, and 1 means using at most all the currently available threads. Values outside the range will be ignored and this function heuristically determines the number of threads to use. |
| METHOD | INTEGER | 0 | Specifies the Power Transform Model:<br>• 0: Yeo-Johnson Model<br>• 1: Box-Cox Model |
| STANDARDIZE | INTEGER | 1 | Indicates whether to perform standardization after Power Transform:<br>• 0: No<br>• 1: Yes |
| OUTPUT_TRANSFORM | INTEGER | 0 | Indicates whether to output transformed result:<br>• 0: No<br>• 1: Yes |

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| MAX_ITERATION | INTEGER | 500 | Maximum number of iterations to fit the λ value. Valid value must be strictly positive. If convergence is not reached after the specified number of iterations, an error is generated. |
| TOLERANCE | DOUBLE | 1.0E-11 | Absolute tolerance to control the accuracy of the fitted λ value. Valid range is `(0, 1)`. |
| INTERVAL_HINT | INTEGER | 0 | Indicates whether the specified interval is a hint or not.<br>• 0: Interval defines the range of λ value<br>• 1: Interval specifies a hint for the initial search range only. The final λ value may fall outside this range. |
| FEATURE_<FEATURE_NAME>_INTERVAL_HINT | INTEGER | defined by INTERVAL_HINT | Feature specific `INTERVAL_HINT` for the specified feature, which overrides the `INTERVAL_HINT` parameter. |
| INTERVAL_START | DOUBLE | -2.0 | Start point of the interval. |
| INTERVAL_END | DOUBLE | 2.0 | End point of the interval. |
| FEATURE_<FEATURE_NAME>_INTERVAL_START | DOUBLE | defined by INTERVAL_START | Start point of the interval, which overrides the `INTERVAL_START` parameter. |
| FEATURE_<FEATURE_NAME>_INTERVAL_END | DOUBLE | defined by INTERVAL_END | End point of the interval, which overrides the `INTERVAL_END` parameter. |

### Output Table

Model Table

| Column | Data Type | Column Name | Description |
|---|---|---|---|
| 1st column | INTEGER | ROW_ID | Row ID |
| 2nd column | NVARCHAR(5000) | MODEL_CONTENT | JSON Format encoded with escaped UTF-8 |

### Transformed Result Table

| Column | Data Type | Column Name | Description |
|---|---|---|---|
| 1st column | ID (in input table) data type | ID (in input table) column name | Referenced by "ID" |
| 2nd column | DOUBLE | FEATURE (in input table) column name prefixed with "TRANSFORMED_" | Referenced by "FEATURES" |

## Examples

### Example 1: Fit model with given data and configure them with different intervals

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_POWER_TRANSFORM_TRAIN_DATA_TBL;
CREATE COLUMN TABLE PAL_POWER_TRANSFORM_TRAIN_DATA_TBL ("ID" INTEGER, "X1"
DOUBLE, "X2" DOUBLE, "X3" DOUBLE);
INSERT INTO PAL_POWER_TRANSFORM_TRAIN_DATA_TBL(ID, X1, X2, X3) VALUES
(0,5.155951363694313,4.076506774438141,6.775578240123138);
INSERT INTO PAL_POWER_TRANSFORM_TRAIN_DATA_TBL(ID, X1, X2, X3) VALUES
(1,5.016602867279994,4.298159629915712,5.972896878286871);
INSERT INTO PAL_POWER_TRANSFORM_TRAIN_DATA_TBL(ID, X1, X2, X3) VALUES
(2,6.330503142514262,4.3079927168024446,4.503660363833093);
INSERT INTO PAL_POWER_TRANSFORM_TRAIN_DATA_TBL(ID, X1, X2, X3) VALUES
(3,4.418133874557737,7.572598100863792,5.582568159406703);
INSERT INTO PAL_POWER_TRANSFORM_TRAIN_DATA_TBL(ID, X1, X2, X3) VALUES
(4,4.043450306707487,4.078382082912484,5.039633969808214);
INSERT INTO PAL_POWER_TRANSFORM_TRAIN_DATA_TBL(ID, X1, X2, X3) VALUES
(5,6.309902963930603,5.716670045148155,5.365876422148958);
INSERT INTO PAL_POWER_TRANSFORM_TRAIN_DATA_TBL(ID, X1, X2, X3) VALUES
(6,9.459266606686224,4.456605337786643,6.528845492078271);
INSERT INTO PAL_POWER_TRANSFORM_TRAIN_DATA_TBL(ID, X1, X2, X3) VALUES
(7,7.042910662171584,5.363604409362707,5.621450381520459);
INSERT INTO PAL_POWER_TRANSFORM_TRAIN_DATA_TBL(ID, X1, X2, X3) VALUES
(8,4.52491025786601,4.820377426072063,4.0940495788733395);
INSERT INTO PAL_POWER_TRANSFORM_TRAIN_DATA_TBL(ID, X1, X2, X3) VALUES
(9,10.085880972733657,4.797140870188886,4.967799257755148);
INSERT INTO PAL_POWER_TRANSFORM_TRAIN_DATA_TBL(ID, X1, X2, X3) VALUES
(10,6.274808607310309,4.087428452689988,4.614905343077156);
INSERT INTO PAL_POWER_TRANSFORM_TRAIN_DATA_TBL(ID, X1, X2, X3) VALUES
(11,4.416353864586462,8.60136467772791,5.295219180385827);
INSERT INTO PAL_POWER_TRANSFORM_TRAIN_DATA_TBL(ID, X1, X2, X3) VALUES
(12,6.204025193299364,6.034236800100728,4.992341419035431);
INSERT INTO PAL_POWER_TRANSFORM_TRAIN_DATA_TBL(ID, X1, X2, X3) VALUES
(13,5.223059748059121,7.395916235686572,4.1072557288544616);
INSERT INTO PAL_POWER_TRANSFORM_TRAIN_DATA_TBL(ID, X1, X2, X3) VALUES
(14,5.52254174106756,5.1980190368842845,7.716796419800201);
INSERT INTO PAL_POWER_TRANSFORM_TRAIN_DATA_TBL(ID, X1, X2, X3) VALUES
(15,4.8140726282581205,5.987543315087799,6.196331328967135);
INSERT INTO PAL_POWER_TRANSFORM_TRAIN_DATA_TBL(ID, X1, X2, X3) VALUES
(16,4.4429461072686145,6.240440141610463,7.042161278878741);
INSERT INTO PAL_POWER_TRANSFORM_TRAIN_DATA_TBL(ID, X1, X2, X3) VALUES
(17,4.355412962823212,5.182920672969789,4.486751411317423);
INSERT INTO PAL_POWER_TRANSFORM_TRAIN_DATA_TBL(ID, X1, X2, X3) VALUES
(18,7.020273410920357,3.8736115238569817,7.3447648834718215);
INSERT INTO PAL_POWER_TRANSFORM_TRAIN_DATA_TBL(ID, X1, X2, X3) VALUES
(19,4.762024626259294,4.961158730017223,4.6924661064778315);
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL ("PARAM_NAME" VARCHAR(256), "INT_VALUE"
INTEGER, "DOUBLE_VALUE" DOUBLE, "STRING_VALUE" VARCHAR(1000));
-- Default value is put in square bracket
-- [0]: Yeo-Johnson; 1: Box-Cox
INSERT INTO PAL_PARAMETER_TBL VALUES ('METHOD', 0, NULL, NULL);
-- 0: Do not standardize; [1]: Standardize
INSERT INTO PAL_PARAMETER_TBL VALUES ('STANDARDIZE', 1, NULL, NULL);
-- [0]: Do not output transform result; 1: Output transform result
```

```
INSERT INTO PAL_PARAMETER_TBL VALUES ('OUTPUT_TRANSFORM', 1, NULL, NULL);
-- Maximum allowed iteration for optimization algorithm to compute lambda value,
valid range is (0, +inf), default is 500
INSERT INTO PAL_PARAMETER_TBL VALUES ('MAX_ITERATION', 500, NULL, NULL);
-- Control the tolerable absolute accuracy error, valid range is (0, 1), default
value is 1.0E-11
INSERT INTO PAL_PARAMETER_TBL VALUES ('TOLERANCE', NULL, 1E-11, NULL);
-- Define an internal for lambda value, final result with be restricted within
the given range, any value is allowed, even interval_start > interval_end,
default value for interval_start is -2.0, default value for interval_end is 2.0
INSERT INTO PAL_PARAMETER_TBL VALUES ('INTERVAL_START', NULL, -4.0, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('INTERVAL_END', NULL, 4.0, NULL);
-- Define thread_ratio, valid range is [0.0, 1.0], default value is 1.0, which
mean to use all currently available CPU resource
INSERT INTO PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 1.0, NULL);
-- Specify the role of given interval: [0]: enforce final fitted hyper-parameter
within given interval; 1: interval only serves as a hint, there is no guarantee
final fitted hyper-parameter fall within given interval
INSERT INTO PAL_PARAMETER_TBL VALUES ('INTERVAL_HINT', 0, NULL, NULL);
-- Override INTERVAL_HINT as 1 for feature X1
INSERT INTO PAL_PARAMETER_TBL VALUES ('FEATURE_X1_INTERVAL_HINT', 1, NULL, NULL);
-- Override Interval Range for feature X2
INSERT INTO PAL_PARAMETER_TBL VALUES ('FEATURE_X2_INTERVAL_START', NULL, -3.0,
NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('FEATURE_X2_INTERVAL_END', NULL, 1.0,
NULL);
DROP TABLE PAL_POWER_TRANSFORM_MODEL_TBL;
CREATE COLUMN TABLE PAL_POWER_TRANSFORM_MODEL_TBL ("ROW_ID" INTEGER,
"MODEL_CONTENT" NVARCHAR(5000));
DROP TABLE PAL_POWER_TRANSFORM_RESULT;
CREATE COLUMN TABLE PAL_POWER_TRANSFORM_RESULT ("ID" INTEGER, "TRANSFORMED_X1"
DOUBLE, "TRANSFORMED_X2" DOUBLE, "TRANSFORMED_X3" DOUBLE);
DO BEGIN
    lt_data = SELECT * FROM PAL_POWER_TRANSFORM_TRAIN_DATA_TBL;
    lt_param = SELECT * FROM PAL_PARAMETER_TBL;
    CALL _SYS_AFL.PAL_POWER_TRANSFORMER_FIT (:lt_data, :lt_param, lt_model,
lt_transformed_result);
    INSERT INTO PAL_POWER_TRANSFORM_MODEL_TBL
    SELECT * FROM :lt_model;
    INSERT INTO PAL_POWER_TRANSFORM_RESULT
    SELECT * FROM :lt_transformed_result;
END;
SELECT * FROM PAL_POWER_TRANSFORM_MODEL_TBL;
SELECT * FROM PAL_POWER_TRANSFORM_RESULT;
```

### Expected Result

Model Table

| | ROW_ID | |
|---|---|---|
| 1 | 1 | {"version":1.0,"method":"yeo-johnson","model-content":{"feature-names":["X1","X2","X3"],"means":[0.4278968124640327,0.4873345601059932,0.78926 |

Transformed Result Table

| | ID | TRANSFORMED_X1 | TRANSFORMED_X2 | TRANSFORMED_X3 |
|---|---|---|---|---|
| 1 | 0 | -0.15396599225825483 | -1.2315676901703851 | 1.131415621368073 |
| 2 | 1 | -0.2955615378249058 | -0.8886256244096423 | 0.539368268355072 |
| 3 | 2 | 0.7119113205683459 | -0.874397865247655 | -1.0197405864379752 |
| 4 | 3 | -1.0478552434950041 | 1.4891652409896192 | 0.1963892143414169 |
| 5 | 4 | -1.6775381052408669 | -1.2284768407445266 | -0.3585072830375077 |
| 6 | 5 | 0.7005394380207413 | 0.5641981123605561 | -0.013147440525400138 |
| 7 | 6 | 1.689401934915368 | -0.6686302058273594 | 0.9635470870449141 |
| 8 | 7 | 1.0482735343262983 | 0.29118462738104905 | 0.2324628942462693 |
| 9 | 8 | -0.893630094492104 | -0.22969822349941935 | -1.6241379014200092 |
| 10 | 9 | 1.7861185147853353 | -0.25529237274810107 | -0.4398917670766944 |
| 11 | 10 | 0.680920705728241 | -1.213614684913252 | -0.8715719681192005 |
| 12 | 11 | -1.0505117337867493 | 1.787681433877953 | -0.08475383370426495 |
| 13 | 12 | 0.6403850494975325 | 0.7755242292315714 | -0.41185620876547696 |
| 14 | 13 | -0.08947644961736705 | 1.426585451386232 | -1.6030595094070779 |
| 15 | 14 | 0.17219141455565493 | 0.14680683330792751 | 1.6804242859654648 |
| 16 | 15 | -0.5217531401186498 | 0.7462458960415905 | 0.7181286949907106 |
| 17 | 16 | -1.0111236396034793 | 0.8981280598059745 | 1.300638063210759 |
| 18 | 17 | -1.143228307189227 | 0.13306306056211326 | -1.0428151656374975 |
| 19 | 18 | 1.0390749378167776 | -1.587310718041004 | 1.4789749684759257 |
| 20 | 19 | -0.5841726065879506 | -0.08096871934309888 | -0.7718674338674684 |

# _SYS_AFL.PAL_POWER_TRANSFORMER_TRANSFORM

## Prerequisites

- The Data Input table must have an ID column as the first column. It must not have a NULL value and no duplicate ID is allowed.
- No categorical data is supported since Power Transform only supports continuous data. Integer Type data is also treated as continuous data.
- No NULL value is allowed in any feature columns.
- The Data Input table must contain columns whose name should exactly match the feature names mentioned in the Model Input table.

## Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | DATA table | IN |
| 2 | MODEL table | IN |
| 3 | PARAMETER table | IN |
| 4 | TRANSFORMED_RESULT table | OUT |

## Input Tables

Data Table

| Column | Reference for Output Table | Data Type | Description |
|---|---|---|---|
| 1st column | ID | INTEGER, VARCHAR, NVARCHAR, BIGINT | ID data, duplicated values in this column is not allowed. |
| Feature columns | FEATURES | INTEGER, DOUBLE, DECIMAL(p, s) | Feature data. |

Model Table

| Column | Data Type | Column Name | Description |
|---|---|---|---|
| 1st column | INTEGER | ROW_ID | Row ID |
| 2nd column | VARCHAR, NVARCHAR, CLOB, NCLOB | MODEL_CONTENT | JSON format encoded with escaped UTF-8 |

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| THREAD_RATIO | DOUBLE | 1.0 | Specifies the ratio of total number of threads that can be used by this function. The value range is from 0 to 1, where 0 means only using 1 thread, and 1 means using at most all the currently available threads. Values outside the range will be ignored and this function heuristically determines the number of threads to use. |
| DIRECTION | INTEGER | 0 | Specifies the direction of transformation:<br>• 0: Transforms from raw data to Gaussian-like data.<br>• 1: Transforms from Gaussian-like data to raw data. |

### Output Table

Transformed Result Table

| Column | Data Type | Column Name | Description |
|---|---|---|---|
| 1st column | ID (in input table) data type | ID (in input table) column name | Referenced by "ID" |
| 2nd column | DOUBLE | FEATURE (in input table) column name prefixed with "TRANSFORMED_" | Referenced by "FEATURES" |

### Examples

### Example 1: Transform given data with pre-trained model

This example reuses the output trained model in Example 1 of the Fit procedure.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_POWER_TRANSFORM_PRED_DATA_TBL;
CREATE COLUMN TABLE PAL_POWER_TRANSFORM_PRED_DATA_TBL ("ID" INTEGER, "X1"
DOUBLE, "X2" DOUBLE, "X3" DOUBLE);
INSERT INTO PAL_POWER_TRANSFORM_PRED_DATA_TBL(ID, X1, X2, X3) VALUES
(0,6.991912980386701,4.670970173420526,4.477527687118971);
INSERT INTO PAL_POWER_TRANSFORM_PRED_DATA_TBL(ID, X1, X2, X3) VALUES
(1,4.160349494791189,5.646340683571756,5.472922703397652);
INSERT INTO PAL_POWER_TRANSFORM_PRED_DATA_TBL(ID, X1, X2, X3) VALUES
(2,6.255789538923059,6.144605769096749,7.999251528207772);
INSERT INTO PAL_POWER_TRANSFORM_PRED_DATA_TBL(ID, X1, X2, X3) VALUES
(3,5.420097306622307,6.904864153403108,6.962576805885151);
INSERT INTO PAL_POWER_TRANSFORM_PRED_DATA_TBL(ID, X1, X2, X3) VALUES
(4,5.793421772603856,4.517881305038014,4.953930733068763);
INSERT INTO PAL_POWER_TRANSFORM_PRED_DATA_TBL(ID, X1, X2, X3) VALUES
(5,6.8300404803883765,5.478248746162876,5.7146424123630215);
INSERT INTO PAL_POWER_TRANSFORM_PRED_DATA_TBL(ID, X1, X2, X3) VALUES
(6,4.471869356988835,5.240600291706576,4.191433278882979);
INSERT INTO PAL_POWER_TRANSFORM_PRED_DATA_TBL(ID, X1, X2, X3) VALUES
(7,3.759174209656167,4.691744344441516,3.8204191340493154);
INSERT INTO PAL_POWER_TRANSFORM_PRED_DATA_TBL(ID, X1, X2, X3) VALUES
(8,4.908710332344595,4.60047789755537,5.902754276628165);
INSERT INTO PAL_POWER_TRANSFORM_PRED_DATA_TBL(ID, X1, X2, X3) VALUES
(9,4.904309197786225,5.684147101844333,4.954790307997223);
INSERT INTO PAL_POWER_TRANSFORM_PRED_DATA_TBL(ID, X1, X2, X3) VALUES
(10,5.150543954032488,7.554681004633586,4.943970548460832);
INSERT INTO PAL_POWER_TRANSFORM_PRED_DATA_TBL(ID, X1, X2, X3) VALUES
(11,6.821817387467123,4.796385999539298,4.470520739217951);
INSERT INTO PAL_POWER_TRANSFORM_PRED_DATA_TBL(ID, X1, X2, X3) VALUES
(12,6.612000371270597,4.299181834650753,6.578031707872528);
INSERT INTO PAL_POWER_TRANSFORM_PRED_DATA_TBL(ID, X1, X2, X3) VALUES
(13,6.485878476547363,4.96807796234731,5.259653673467295);
INSERT INTO PAL_POWER_TRANSFORM_PRED_DATA_TBL(ID, X1, X2, X3) VALUES
(14,4.930265818722317,4.193547704240974,6.453566343266206);
INSERT INTO PAL_POWER_TRANSFORM_PRED_DATA_TBL(ID, X1, X2, X3) VALUES
(15,6.920819090201766,4.605442724704531,6.267915315123228);
INSERT INTO PAL_POWER_TRANSFORM_PRED_DATA_TBL(ID, X1, X2, X3) VALUES
(16,4.693542928384049,9.056507077213578,6.3682878843316795);
INSERT INTO PAL_POWER_TRANSFORM_PRED_DATA_TBL(ID, X1, X2, X3) VALUES
(17,5.50224844563645,5.067995183131792,4.5167161566174);
INSERT INTO PAL_POWER_TRANSFORM_PRED_DATA_TBL(ID, X1, X2, X3) VALUES
(18,4.985586122525704,5.942695716838256,6.111872081868994);
INSERT INTO PAL_POWER_TRANSFORM_PRED_DATA_TBL(ID, X1, X2, X3) VALUES
(19,4.621055832129848,4.940330094835402,5.064032310349425);
DROP TABLE PAL_PARAMETER_PRED_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_PRED_TBL ("PARAM_NAME" VARCHAR(256),
"INT_VALUE" INTEGER, "DOUBLE_VALUE" DOUBLE, "STRING_VALUE" VARCHAR(1000));
-- Default value is put in square bracket
-- Define thread_ratio, valid range is [0.0, 1.0], default value is 1.0, which
mean to use all currently available CPU resource
INSERT INTO PAL_PARAMETER_PRED_TBL VALUES ('THREAD_RATIO', NULL, 1.0, NULL);
-- [0]: Transform raw data to gaussian-like data; 1: Transform gaussian-like
data back to raw data
INSERT INTO PAL_PARAMETER_PRED_TBL VALUES ('DIRECTION', 0, NULL, NULL);
DROP TABLE PAL_POWER_TRANSFORM_PRED_RESULT;
CREATE COLUMN TABLE PAL_POWER_TRANSFORM_PRED_RESULT ("ID" INTEGER,
"TRANSFORMED_X1" DOUBLE, "TRANSFORMED_X2" DOUBLE, "TRANSFORMED_X3" DOUBLE);
DO BEGIN
    lt_data = SELECT * FROM PAL_POWER_TRANSFORM_PRED_DATA_TBL;
    lt_model = SELECT * FROM PAL_POWER_TRANSFORM_MODEL_TBL;
    lt_param = SELECT * FROM PAL_PARAMETER_PRED_TBL;
    CALL _SYS_AFL.PAL_POWER_TRANSFORMER_TRANSFORM
(:lt_data, :lt_model, :lt_param, lt_transformed_result);
    INSERT INTO PAL_POWER_TRANSFORM_PRED_RESULT
```

```
    SELECT * FROM :lt_transformed_result;
END;
SELECT * FROM PAL_POWER_TRANSFORM_PRED_RESULT;
```

**Expected Results**

Transformed Result Table

| | ID | TRANSFORMED_X1 | TRANSFORMED_X2 | TRANSFORMED_X3 |
|---|---|---|---|---|
| 1 | 0 | 1.0274290090061804 | -0.3997920271538501 | -1.0554653457709404 |
| 2 | 1 | -1.4647834728706168 | 0.5132604158129469 | 0.09220777276328675 |
| 3 | 2 | 0.6701572469280428 | 0.842464860283103 | 1.8216954024497678 |
| 4 | 3 | 0.08720410981383458 | 1.2301198930750543 | 1.2513663080681372 |
| 5 | 4 | 0.37693281156623576 | -0.5885816021847741 | -0.45584185250068116 |
| 6 | 5 | 0.9582640297824047 | 0.3847357161123293 | 0.31713640723873493 |
| 7 | 6 | -0.9689984244888201 | 0.18503268652127014 | -1.4713517027344147 |
| 8 | 7 | -2.2692163758405925 | -0.37533855301942104 | -2.088258978174046 |
| 9 | 8 | -0.4128700081725995 | -0.48480399081536035 | 0.4807384462072417 |
| 10 | 9 | -0.4178062057639734 | 0.5408421128532319 | -0.454850991035925 |
| 11 | 10 | -0.15926385917122213 | 1.4829949997470482 | -0.467345274144793 |
| 12 | 11 | 0.9546236335750553 | -0.25612899068488515 | -1.065105235468178 |
| 13 | 12 | 0.8572881324356715 | -0.8871428848304778 | 0.9979292369878365 |
| 14 | 13 | 0.7943843948374897 | -0.07392860178415304 | -0.12144043399458627 |
| 15 | 14 | -0.3888684228393494 | -1.0450217950303982 | 0.9100011269248537 |
| 16 | 15 | 0.9976266919445499 | -0.4787115188037804 | 0.7729578457548354 |
| 17 | 16 | -0.669193000914717 | 1.8915257062556359 | 0.8479521381353966 |
| 18 | 17 | 0.1557068159540261 | 0.025068727452997153 | -1.0020259059158168 |
| 19 | 18 | -0.32857093688890615 | 0.7175678874977686 | 0.6519445961162311 |
| 20 | 19 | -0.7629448054182556 | -0.10230970637886792 | -0.3313264674250406 |

## 5.8.13 Principal Component Analysis (PCA)

Principal component analysis (PCA) aims at reducing the dimensionality of multivariate data while accounting for as much of the variation in the original data set as possible. This technique is especially useful when the variables within the data set are highly correlated.

Principal components seek to transform the original variables to a new set of variables that are:

- linear combinations of the variables in the data set;
- uncorrelated with each other;
- ordered according to the amount of variations of the original variables that they explain.

Suppose the data matrix is $X \in R^{m \times n}$, with $m$ samples of data and each sample has $n$ variables. In PAL, the principal component analysis of $X$ is achieved by its SVD (singular value decomposition):

$X=USV^T$,

where the values of the projections of the observations on the principal components, i.e. the matrix of loadings, is given by $V$:

$Loadings=V$,

and the matrix of scores is obtained as:

$Scores=US=USV^T\ V=XV$

Note that if there exists one variable which has constant value across data items, you cannot scale variables any more.

> **Note**
>
> Due to numerical reason, the signs of components in result loadings matrix might be different from those in the versions prior to SAP HANA Platform 2.0 SPS 02. However, this will not impact the correctness of the result since different signs of a component are considered equivalent in PCA.

## Prerequisites

- No missing or null data in the inputs.
- The data is numeric, not categorical.
- Each column must have at least two different values.

## _SYS_AFL.PAL_PCA

This is a principal component analysis procedure.

### Overview of all Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<LOADINGS table>` | OUT |
| 4 | `<LOADINGS_INFORMATION table>` | OUT |
| 5 | `<SCORES table>` | OUT |
| 6 | `<SCALING_INFORMATION table>` | OUT |

### Input Table

| Table | Column | Reference for Output Table | Data Type | Description |
|---|---|---|---|---|
| DATA | 1st column | ID | INTEGER, VARCHAR, or NVARCHAR | ID |
| | Feature columns | FEATURES | INTEGER, DOUBLE, or DECIMAL(p, s) | Features |

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| THREAD_RATIO | DOUBLE | 1.0 | Specifies the ratio of total number of threads that can be used by this function. The value range is from 0 to 1, where 0 means only using 1 thread, and 1 means using at most all the currently available threads. Values outside the range will be ignored and this function heuristically determines the number of threads to use. |
| SCALING | INTEGER | 0 | Specifies whether the variables should be scaled to have unit variance before the analysis takes place.<br>• 0: No<br>• 1: Yes |
| SCORES | INTEGER | 0 | Specifies whether to output the scores on each principal component.<br>• 0: No<br>• 1: Yes |

### Output Tables

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| LOADINGS | 1st column | NVARCHAR(100) | COMPONENT_ID | Principal component ID |
| | Loading columns | DOUBLE | LOADINGS_ + FEATURES (in input table) column name | Loadings |
| LOADINGS_INFORMATION | 1st column | NVARCHAR(100) | COMPONENT_ID | Principal component ID |

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| | 2nd column | DOUBLE | SD | Standard deviations of the principal components |
| | 3rd column | DOUBLE | VAR_PROP | Variance proportion to the total variance explained by each principal component |
| | 4th column | DOUBLE | CUM_VAR_PROP | Cumulative variance proportion to the total variance explained by each principal component |
| SCORES | 1st column | ID (in input table) data type | ID (in input table) column name | Data item ID |
| | Score columns | DOUBLE | COMPONENT_ + i (i sequences from 1 to the number of columns in FEATURES (in input table)) | Scores |
| SCALING_INFORMATION | 1st column | INTEGER | VARIABLE_ID | Variable ID |
| | 2nd column | DOUBLE | MEAN | Mean value of each variable |
| | 3rd column | DOUBLE | SCALE | Scale value of each variable |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA "DM_PAL";
DROP TABLE PAL_PCA_DATA_TBL;
CREATE COLUMN TABLE PAL_PCA_DATA_TBL ("ID" INTEGER, "X1" DOUBLE, "X2" DOUBLE,
"X3" DOUBLE, "X4" DOUBLE, "X5" DOUBLE, "X6" DOUBLE);
INSERT INTO PAL_PCA_DATA_TBL VALUES (1, 12, 52, 20, 44, 48, 16);
INSERT INTO PAL_PCA_DATA_TBL VALUES (2, 12, 57, 25, 45, 50, 16);
INSERT INTO PAL_PCA_DATA_TBL VALUES (3, 12, 54, 21, 45, 50, 16);
INSERT INTO PAL_PCA_DATA_TBL VALUES (4, 13, 52, 21, 46, 51, 17);
INSERT INTO PAL_PCA_DATA_TBL VALUES (5, 14, 54, 24, 46, 51, 17);
INSERT INTO PAL_PCA_DATA_TBL VALUES (6, 22, 52, 25, 54, 58, 26);
INSERT INTO PAL_PCA_DATA_TBL VALUES (7, 22, 56, 26, 55, 58, 27);
INSERT INTO PAL_PCA_DATA_TBL VALUES (8, 17, 52, 21, 45, 52, 17);
INSERT INTO PAL_PCA_DATA_TBL VALUES (9, 15, 53, 24, 45, 53, 18);
```

```
INSERT INTO PAL_PCA_DATA_TBL VALUES (10, 23, 54, 23, 53, 57, 24);
INSERT INTO PAL_PCA_DATA_TBL VALUES (11, 25, 54, 23, 55, 58, 25);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL ("PARAM_NAME"
VARCHAR(256), "INT_VALUE" INTEGER, "DOUBLE_VALUE" DOUBLE, "STRING_VALUE"
VARCHAR(1000));
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SCALING', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SCORES', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 0.5, NULL);
DROP TABLE PAL_PCA_LOADINGS_TBL;
DROP TABLE PAL_PCA_LOADINGS_INFORMATION;
DROP TABLE PAL_PCA_SCORES;
DROP TABLE PAL_PCA_SCALING_INFORMATION_TBL;
CREATE COLUMN TABLE PAL_PCA_LOADINGS_TBL ("COMPONENT_ID" NVARCHAR(100),
"LOADINGS_X1" DOUBLE, "LOADINGS_X2" DOUBLE, "LOADINGS_X3" DOUBLE, "LOADINGS_X4"
DOUBLE, "LOADINGS_X5" DOUBLE, "LOADINGS_X6" DOUBLE);
CREATE COLUMN TABLE PAL_PCA_LOADINGS_INFORMATION ("COMPONENT_ID" NVARCHAR(100),
"SD" DOUBLE, "VAR_PROP" DOUBLE, "CUM_VAR_PROP" DOUBLE);
CREATE COLUMN TABLE PAL_PCA_SCORES ("ID" INTEGER, "COMPONENT_1" DOUBLE,
"COMPONENT_2" DOUBLE, "COMPONENT_3" DOUBLE, "COMPONENT_4" DOUBLE, "COMPONENT_5"
DOUBLE, "COMPONENT_6" DOUBLE);
CREATE COLUMN TABLE PAL_PCA_SCALING_INFORMATION_TBL ("VARIABLE_ID" INTEGER,
"MEAN" DOUBLE, "SCALE" DOUBLE);
CALL _SYS_AFL.PAL_PCA (PAL_PCA_DATA_TBL, #PAL_PARAMETER_TBL,
PAL_PCA_LOADINGS_TBL, PAL_PCA_LOADINGS_INFORMATION, PAL_PCA_SCORES,
PAL_PCA_SCALING_INFORMATION_TBL) WITH OVERVIEW;
```

**Expected Result**

LOADINGS:

| | COMPONENT_ID | LOADINGS_X1 | LOADINGS_X2 | LOADINGS_X3 | LOADINGS_X4 | LOADINGS_X5 | LOADINGS_X6 |
|---|---|---|---|---|---|---|---|
| 1 | Comp1 | 0.45294914528159846 | 0.13899936408534194 | 0.32821054855366505 | 0.47048759140095997 | 0.4720970616704375 | 0.4728211872267412 |
| 2 | Comp2 | -0.23943986612392326 | 0.7836762522372017 | 0.5313532760455476 | -0.10914669994079657 | -0.13752463728748518 | -0.1239263904769926 |
| 3 | Comp3 | -0.20976200429503894 | -0.5872632991924557 | 0.7459306160756265 | -0.21932229761535155 | 0.08099550837568655 | -0.00683318142910... |
| 4 | Comp4 | -0.6301652184225305 | -0.07120016929438... | -0.04735043512438... | 0.4957339731116037 | -0.3282957911618624 | 0.491985742765841... |
| 5 | Comp5 | -0.5353844722069161 | 0.09669986295964501 | -0.21144789449537... | -0.17349821839060436 | 0.7931316257692921 | 0.011957581554266... |
| 6 | Comp6 | 0.09879670168361362 | 0.08504233228783303 | -0.08109729996619... | -0.6654114141767857 | -0.12171755762426063 | 0.7203073839986043 |

LOADINGS_INFORMATION:

| | COMPONENT_ID | SD | VAR_PROP | CUM_VAR_PROP |
|---|---|---|---|---|
| 1 | Comp1 | 2.0708452571912694 | 0.7147333465385954 | 0.7147333465385954 |
| 2 | Comp2 | 1.1528557732691949 | 0.22151273899335208 | 0.9362460855319475 |
| 3 | Comp3 | 0.5394023255908542 | 0.04849247814213696 | 0.9847385636740845 |
| 4 | Comp4 | 0.25890394362931... | 0.011171875337801... | 0.9959104390118865 |
| 5 | Comp5 | 0.12492794025277... | 0.002601165042633... | 0.9985116040545199 |
| 6 | Comp6 | 0.09450066493353... | 0.001488395945480... | 1 |

SCORES:

| | ID | COMPONENT_1 | COMPONENT_2 | COMPONENT_3 | COMPONENT_4 | COMPONENT_5 | COMPONENT_6 |
|---|---|---|---|---|---|---|---|
| 1 | 1 | -2.5949626091856435 | -0.91473117805658 | -0.2395109185244051 | 0.32503028740527456 | -0.19871465748219405 | 0.1261377849424523 |
| 2 | 2 | -1.0116178915478713 | 2.636463041223107 | -0.11677373744675223 | -0.0716985617494279 | -0.05774909489033618 | -0.032425398635060554 |
| 3 | 3 | -1.914850599578437 | 0.1822364665066... | -0.5658719191081283 | 0.14942755222035248 | 0.19344339811646008 | -0.02123461923627623 |
| 4 | 4 | -1.6588018494761192 | -0.880246887731... | 0.06054198923857351 | 0.23678081666380074 | 0.14845816155614736 | -0.1183892624803024 |
| 5 | 5 | -0.9120712600039894 | 0.7969536351865... | 0.44264179023409894 | -0.04356505124817... | -0.16047523969849217 | -0.11976286117271706 |
| 6 | 6 | 2.431462002709938 | -0.934381379489... | 0.9377010351991636 | 0.23735188588875783 | -0.03249811943075974 | -0.04145976052255662 |
| 7 | 7 | 3.1301641330543877 | 1.1358332631687... | -0.12830532211380283 | 0.26028110035830454 | 0.05608905056469285 | 0.1355212766727144 |
| 8 | 8 | -1.2751684705213444 | -1.083311226290... | -0.037349407104622... | -0.4569168460297653 | -0.028854686308435... | 0.0706767749066935 |
| 9 | 9 | -0.6506761797485409 | 0.2087287686314... | 0.8371710425526957 | -0.29839016918884... | 0.1366509863483933 | 0.0870039035823032 |
| 10 | 10 | 1.9224981962744248 | -0.470893281997... | -0.5163754694212782 | -0.1614099564837212 | 0.0088325908850141... | 0.016873883984317994 |
| 11 | 11 | 2.534024528023197 | -0.676651221151... | -0.6738690835055562 | -0.1768910578365609 | -0.0651823896604905 | -0.10294172204156993 |

SCALING_INFORMATION:

| | VARIABLE_ID | MEAN | SCALE |
|---|---|---|---|
| 1 | 1 | 17 | 5.039841267341661 |
| 2 | 2 | 53.63636363636363 | 1.6895400127092153 |
| 3 | 3 | 23 | 2 |
| 4 | 4 | 48.45454545454545 | 4.655397649259112 |
| 5 | 5 | 53.27272727272727 | 3.7707004413214555 |
| 6 | 6 | 19.90909090909091 | 4.5266885347800425 |

## _SYS_AFL.PAL_PCA_PROJECT

This is a principal component analysis projection procedure.

### Overview of all Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<LOADINGS table>` | IN |
| 3 | `<SCALING_INFORMATION table>` | IN |
| 4 | `<PARAMETER table>` | IN |
| 5 | `<PROJECTION table>` | OUT |

### Input Tables

| Table | Column | Reference for Output Table | Data Type | Description | Constraint |
|---|---|---|---|---|---|
| DATA | 1st column | ID | INTEGER | ID | |

| Table | Column | Reference for Output Table | Data Type | Description | Constraint |
|---|---|---|---|---|---|
| | Feature columns | FEATURES | INTEGER, DOUBLE, or DECIMAL(p, s) | Features | |
| LOADINGS | 1st column | | NVARCHAR | Principal component ID | This is a full model, which means that it is a square matrix and the dimension is equal to the number of variables in input data. |
| | Loading columns | | DOUBLE | Loadings. | |
| SCALING_INFORMATION | 1st column | | INTEGER | Variable ID | Row number of this table should be equal to the number of variables in input data. |
| | 2nd column | | DOUBLE | Mean value of each variable | |
| | 3rd column | | DOUBLE | Scale value of each variable | |

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| THREAD_RATIO | DOUBLE | 1.0 | Specifies the ratio of total number of threads that can be used by this function. The value range is from 0 to 1, where 0 means only using 1 thread, and 1 means using at most all the currently available threads. Values outside the range will be ignored and this function heuristically determines the number of threads to use. |

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| SCALING | INTEGER | 0 | Specifies whether the variables should be scaled to have unit variance before the analysis takes place.<br>• 0: No<br>• 1: Yes |
| MAX_COMPONENTS | INTEGER | Number of variables | Specifies the number of components to be retained.<br>Value range: 0 < MAX_COMPONENTS ≤ variable number of original data |

> **Note**
>
> Set the SCALING parameter to the same value you used in _SYS_AFL.PAL_PCA to train the model. If you provide a different value, PAL issues a SQL warning when you run the procedure. PAL then automatically adjusts the SCALING parameter to match the model.

### Output Table

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| SCORES | 1st column | ID (in input table) data type | ID (in input table) column name | Data item ID |
| | Score columns | DOUBLE | COMPONENT_ + i (i sequences from 1 to number of columns in FEATURES (in input table)) | Scores |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.
- Table `PAL_PCA_LOADINGS_TBL` and table `PAL_PCA_SCALING_INFO_TBL` are output of PCA in the previous example.

```
SET SCHEMA "DM_PAL";
DROP TABLE PAL_PCA_PROJECT_DATA_TBL;
CREATE COLUMN TABLE PAL_PCA_PROJECT_DATA_TBL ("ID" INTEGER, "X1" DOUBLE, "X2"
DOUBLE, "X3" DOUBLE, "X4" DOUBLE, "X5" DOUBLE, "X6" DOUBLE);
INSERT INTO PAL_PCA_PROJECT_DATA_TBL VALUES (1, 2, 32, 10, 54, 38, 20);
INSERT INTO PAL_PCA_PROJECT_DATA_TBL VALUES (2, 9, 57, 20, 25, 48, 19);
INSERT INTO PAL_PCA_PROJECT_DATA_TBL VALUES (3, 12, 24, 28, 35, 30, 20);
```

```
INSERT INTO PAL_PCA_PROJECT_DATA_TBL VALUES (4, 15, 42, 27, 36, 61, 27);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL ("PARAM_NAME"
VARCHAR(256), "INT_VALUE" INTEGER, "DOUBLE_VALUE" DOUBLE, "STRING_VALUE"
VARCHAR(1000));
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MAX_COMPONENTS', 4, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SCALING', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 0.5, NULL);
CALL _SYS_AFL.PAL_PCA_PROJECT (PAL_PCA_PROJECT_DATA_TBL, PAL_PCA_LOADINGS_TBL,
PAL_PCA_SCALING_INFORMATION_TBL, #PAL_PARAMETER_TBL, ?);
```

**Expected Result**

SCORES:

| | ID | COMPONENT_1 | COMPONENT_2 | COMPONENT_3 | COMPONENT_4 | COMPONENT_5 | COMPONENT_6 |
|---|---|---|---|---|---|---|---|
| 1 | 1 | -6.603741527545116 | 12.352443837611009 | -2.7068428225172037 | 5.025234762683863 | ? | ? |
| 2 | 2 | -4.060068807988197 | -1.9103252645918178 | 0.9619990768389393 | -1.207742710356588 | ? | ? |
| 3 | 3 | -6.331078830027881 | 11.018844535223138 | -12.507987030133343 | 2.35913598150486... | ? | ? |
| 4 | 4 | -0.0312231746867... | 4.423641401385762 | -6.361790121224823 | -0.5825785566358... | ? | ? |

> **Note**
>
> The wrapper procedure generation approach to call PAL functions is still supported. For more information, refer to the older versions (prior to SAP HANA Platform 2.0 SPS 02) of this document.

## 5.8.14 Random Distribution Sampling

Random distribution sampling is a random generation function with a given distribution.

In PAL, this procedure supports many different univariate distributions and a multivariate distribution. The probability density functions of different distributions are defined as follows:

**Univariate Distributions**

- Bernoulli
  $$P(i|p) = \begin{cases} p \ , & i = 1 \\ 1-p, & i = 0 \end{cases}$$
- Beta
  $$p(x|\alpha,\beta) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{\mathrm{B}(\alpha,\beta)}, where\ \mathrm{B}(\alpha,\beta) = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)}, \quad 0 \le x \le 1$$
- Binomial
  $$P(i|t,p) = \binom{t}{i} \cdot p^i \cdot (1-p)^{t-i} \quad , \quad i \ge 0$$
- Cauchy
  $$p(x|a,b) = \frac{1}{\pi b \cdot \left[1 + \left(\frac{x-a}{b}\right)^2\right]}$$
- Chi-squared
  $$p(x|n) = \frac{1}{\Gamma\left({}^{n}/_{2}\right) \cdot 2^{n/2}} \cdot x^{\frac{n}{2}-1} \cdot e^{-\frac{x}{2}} \quad , \quad x \ge 0$$
- Exponential
  $$p(x|\lambda) = \lambda e^{-\lambda x} \quad , \quad x > 0$$
- Extreme Value
  $$p(x|a,b) = \frac{1}{b} \cdot z(x) \cdot e^{-z(x)} \quad where \quad z(x) = e^{\frac{a-x}{b}}$$
- Fisher F
  $$p(x|m,n) = \frac{\Gamma\left(\frac{m+n}{2}\right)}{\Gamma\left(\frac{m}{2}\right) \cdot \Gamma\left(\frac{n}{2}\right)} \cdot \frac{\left(\frac{mx}{n}\right)^{\frac{m}{2}}}{x \cdot \left(1 + \frac{mx}{n}\right)^{\frac{m+n}{2}}} \quad , \quad x \ge 0$$
- Gamma
  $$p(x|\alpha,\beta) = \frac{1}{\Gamma(\alpha) \cdot \beta^{\alpha}} \cdot x^{\alpha-1} \cdot e^{\frac{-x}{\beta}} \quad , \quad x \ge 0$$
- Geometric
  $$P(i|p) = p \cdot (1-p)^i \quad , \quad i \ge 0$$

- Lognormal

$$p(x|m,s) = \frac{1}{sx\sqrt{2\pi}} \cdot e^{-\frac{(\ln x - m)^2}{2s^2}} \quad , \quad x \geq 0$$

- Negative Binomial

$$P(i|k,p) = \binom{k+i-1}{i} \cdot p^k \cdot (1-p)^i \quad , \quad i \geq 0$$

- Normal

$$p(x|\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} \cdot e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

- PERT

$$p(x|a,b,m,\lambda) = Beta(x|\alpha,\beta) * (b-a) + a, \quad a \leq x \leq b,$$

*Where*

$$\alpha = \frac{(\mu-a)(2m-a-b)}{(m-\mu)(b-a)}, \beta = \frac{\alpha(b-\mu)}{(\mu-a)}, \mu = \frac{a+b+\lambda m}{(\lambda+2)}$$

- Poisson

$$p(i|\theta) = \frac{\theta^i}{i!} e^{-\theta}, \qquad i \geq 0$$

- Student T

$$p(x|n) = \frac{1}{\sqrt{n\pi}} \cdot \frac{\Gamma\left(\frac{n+1}{2}\right)}{\Gamma\left(\frac{n}{2}\right)} \cdot \left(1 + \frac{x^2}{n}\right)^{-\frac{n+1}{2}}$$

- Uniform

$$p(x|a,b) = \frac{1}{b-a} \quad , \quad a \leq x < b$$

- Weibull

$$p(x|a,b) = \frac{a}{b} \cdot \left(\frac{x}{b}\right)^{a-1} \cdot e^{-\left(\frac{x}{b}\right)^a} \quad , \quad x \geq 0$$

### Multivariate Distribution

- Multinomial

$$P(i_1, \ldots, i_k | n, p_1, \ldots, p_k) = \frac{n!}{i_1! \ldots i_k!} p_1^{i_1} \ldots p_k^{i_k}, \quad \sum_{j=1}^{k} i_j = n$$

## Prerequisites

None.

## _SYS_AFL.PAL_DISTRIBUTION_RANDOM

This is a random generation procedure with a given distribution.

### Overview of all Tables

| Position | Table Name | Parameter Type |
| --- | --- | --- |
| 1 | `<DISTRIBUTION_PARAMETER table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<RESULT table>` | OUT |

### Input Table

| Table | Column | Data Type | Description |
|---|---|---|---|
| DISTRIBUTION_PARAMETER | 1st column | VARCHAR or NVARCHAR | Name of distribution parameters. The supported distributions and their parameters are as follows:<br>• BERNOULLI<br>  • SUCCESS_FRACTION (default: 0.5) in [0, 1]<br>• BETA<br>  • SHAPE1 (default: 0.5) in (0, +∞)<br>  • SHAPE2 (default: 0.5) in (0, +∞)<br>• BINOMIAL<br>  • TRIALS (default: 1) in [0, +∞)<br>  • SUCCESS_FRACTION (default: 0.5) in [0, 1]<br>• CAUCHY<br>  • LOCATION (default: 0) in (-∞, +∞)<br>  • SCALE (default: 1) in (0, +∞)<br>• CHI_SQUARED<br>  • DEGREES_OF_FREEDOM (default: 1) in (0, +∞)<br>• EXPONENTIAL<br>  • RATE (default: 1) in (0, +∞)<br>• EXTREME_VALUE Value<br>  • LOCATION (default: 0) in (-∞, +∞)<br>  • SCALE (default: 1) in (0, +∞)<br>• FISHER_F<br>  • DEGREES_OF_FREEDOM1 (default: 1) in (0, +∞)<br>  • DEGREES_OF_FREEDOM2 (default: 1) in (0, +∞)<br>• GAMMA |

| Table | Column | Data Type | Description |
|---|---|---|---|
| | | | • SHAPE (default: 1) in (0, +∞)<br>• SCALE (default: 1) in (0, +∞)<br>• GEOMETRIC<br>• SUCCESS_FRACTION (default: 0.5) in [0, 1]<br>• LOGNORMAL<br>• LOCATION (default: 0) in (-∞, +∞)<br>• SCALE (default: 1) in (0, +∞)<br>• NEGATIVE_BINOMIAL<br>• SUCCESSES (default: 1) in (0, +∞)<br>• SUCCESS_FRACTION (default: 0.5) in [0, 1]<br>• NORMAL<br>• MEAN (default: 0) in (-∞, +∞)<br>• VARIANCE (default: 1) in (0, +∞)<br>• SD (default: 1) in (0, +∞)<br>VARIANCE and SD cannot be used together. Choose one from them.<br>• PERT<br>• MIN (default: -1) in (-∞, +∞)<br>• MODE (default: 0) in (-∞, +∞)<br>• MAX (default: 1) in (-∞, +∞)<br>• SCALE (default: 4) in (0, +∞)<br>(MIN < MODE < MAX)<br>• POISSON<br>• THETA (default: 1) in (0, +∞)<br>• STUDENT_T<br>• DEGREES_OF_FREEDOM (default: 1) in (0, +∞) |

| Table | Column | Data Type | Description |
|---|---|---|---|
| | | | • UNIFORM<br>  • MIN (default: 0) in (-∞, +∞)<br>  • MAX (default: 1) in (-∞, +∞)<br>(MIN < MAX)<br>• WEIBULL<br>  • SHAPE (default: 1) in (0, +∞)<br>  • SCALE (default: 1) in (0, +∞) |
| | 2nd column | VARCHAR or NVARCHAR | Value of distribution parameters |

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| NUM_RANDOM | INTEGER | 100 | Specifies the number of random data to be generated. |
| SEED | INTEGER | 0 | Indicates the seed used to initialize the random number generator.<br>• 0: Uses the system time<br>• Not 0: Uses the specified seed<br>Note that when multithreading is enabled, the random number sequences of different runs might be different even when SEED is set the same. |

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| THREAD_RATIO | DOUBLE | 0 | Specifies the ratio of total number of threads that can be used by this function. The value range is from 0 to 1, where 0 means only using 1 thread, and 1 means using at most all the currently available threads. Values outside the range will be ignored and this function heuristically determines the number of threads to use. |

### Output Table

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| RESULT | 1st column | INTEGER | ID | ID |
| | 2nd column | DOUBLE | GENERATED_NUMBER | Randomly generated numbers |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_SAMPLING_DISTRIBUTION_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_SAMPLING_DISTRIBUTION_PARAMETER_TBL ("NAME"
VARCHAR(100), "VALUE" VARCHAR(100));
INSERT INTO PAL_SAMPLING_DISTRIBUTION_PARAMETER_TBL VALUES ('DISTRIBUTIONNAME',
'UNIFORM');
INSERT INTO PAL_SAMPLING_DISTRIBUTION_PARAMETER_TBL VALUES ('MIN', '2');
INSERT INTO PAL_SAMPLING_DISTRIBUTION_PARAMETER_TBL VALUES ('MAX', '2.1');
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL ("PARAM_NAME"
VARCHAR(256), "INT_VALUE" INTEGER, "DOUBLE_VALUE" DOUBLE, "STRING_VALUE"
VARCHAR(1000));
INSERT INTO #PAL_PARAMETER_TBL VALUES ('NUM_RANDOM', 10000, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SEED', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 0, NULL);
CALL _SYS_AFL.PAL_DISTRIBUTION_RANDOM (PAL_SAMPLING_DISTRIBUTION_PARAMETER_TBL,
#PAL_PARAMETER_TBL, ?);
```

### Expected Result

RESULT:

| | ID | GENERATED_NUMBER |
|---|---|---|
| 1 | 0 | 2.095142321946089 |
| 2 | 1 | 2.0972546639544216 |
| 3 | 2 | 2.0233884234668036 |
| 4 | 3 | 2.0669977409388687 |
| 5 | 4 | 2.0711408105114995 |
| 6 | 5 | 2.0369952048428956 |
| 7 | 6 | 2.065684272600018 |
| 8 | 7 | 2.0505459382252367 |
| 9 | 8 | 2.0473018492451174 |
| 10 | 9 | 2.0187000964960315 |
| 11 | 10 | 2.0038720751166443 |
| 12 | 11 | 2.0933210388078534 |
| 13 | 12 | 2.034019564790146 |
| 14 | 13 | 2.0793808234262783 |
| 15 | 14 | 2.096256957111645 |
| 16 | 15 | 2.0592964463998094 |
| 17 | 16 | 2.0157299512348392 |
| 18 | 17 | 2.0444895281490054 |
| 19 | 18 | 2.0401280516599947 |
| 20 | 19 | 2.0464206295872645 |
| 21 | 20 | 2.038833071742825 |

## _SYS_AFL.PAL_DISTRIBUTION_RANDOM_MULTIVARIATE

This is a random generation procedure with a given multivariate distribution.

### Overview of all Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DISTRIBUTION_PARAMETER table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<RESULT table>` | OUT |

### Input Table

| Table | Column | Reference for Output Table | Data Type | Description |
|---|---|---|---|---|
| DISTRIBUTION_PARAMETER | 1st column | | VARCHAR or NVARCHAR | For multinomial distribution: number of trials. (Hence should be a string of an INTEGER value.) |
| | Other columns | VARIABLE | DOUBLE | For multinomial distribution: success fractions of each category. (These success fractions should sum to 1). |

### Parameter Table

Mandatory Parameters

The following parameter is mandatory and must be given a value.

| Name | Data Type | Description |
|---|---|---|
| DISTRIBUTIONNAME | VARCHAR | Distribution name:<br>• MULTINOMIAL |

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| NUM_RANDOM | INTEGER | 100 | Specifies the number of random data to be generated. |
| SEED | INTEGER | 0 | Indicates the seed used to initialize the random number generator.<br>• 0: Uses the system time<br>• Not 0: Uses the specified seed<br>Note that when multithreading is enabled, the random number sequences of different runs might be different even when SEED is set the same. |

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| THREAD_RATIO | DOUBLE | 0 | Specifies the ratio of total number of threads that can be used by this function. The value range is from 0 to 1, where 0 means only using 1 thread, and 1 means using at most all the currently available threads. Values outside the range will be ignored and this function heuristically determines the number of threads to use. |

### Output Table

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| RESULT | 1st column | INTEGER | ID | ID |
| | Random number columns | DOUBLE | RANDOM_ + VARIABLES (in input table) column name | Randomly generated numbers |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_SAMPLING_MVAR_DISTRIBUTION_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_SAMPLING_MVAR_DISTRIBUTION_PARAMETER_TBL ("TRIALS"
VARCHAR(100), "P1" DOUBLE, "P2" DOUBLE, "P3" DOUBLE, "P4" DOUBLE);
INSERT INTO PAL_SAMPLING_MVAR_DISTRIBUTION_PARAMETER_TBL VALUES ('100', 0.1,
0.2, 0.3, 0.4);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL ("PARAM_NAME"
VARCHAR(256), "INT_VALUE" INTEGER, "DOUBLE_VALUE" DOUBLE, "STRING_VALUE"
VARCHAR(1000));
INSERT INTO #PAL_PARAMETER_TBL VALUES ('DISTRIBUTIONNAME', NULL, NULL,
'MULTINOMIAL');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('NUM_RANDOM', 10, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SEED', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 0, NULL);
CALL _SYS_AFL.PAL_DISTRIBUTION_RANDOM_MULTIVARIATE
(PAL_SAMPLING_MVAR_DISTRIBUTION_PARAMETER_TBL, #PAL_PARAMETER_TBL, ?);
```

### Expected Result

RESULT:

| | ID | RANDOM_P1 | RANDOM_P2 | RANDOM_P3 | RANDOM_P4 |
|---|---|---|---|---|---|
| 1 | 0 | 7 | 20 | 30 | 43 |
| 2 | 1 | 9 | 28 | 23 | 40 |
| 3 | 2 | 7 | 22 | 32 | 39 |
| 4 | 3 | 11 | 19 | 28 | 42 |
| 5 | 4 | 12 | 18 | 29 | 41 |
| 6 | 5 | 13 | 24 | 20 | 43 |
| 7 | 6 | 8 | 19 | 26 | 47 |
| 8 | 7 | 15 | 15 | 20 | 50 |
| 9 | 8 | 15 | 25 | 22 | 38 |
| 10 | 9 | 8 | 26 | 35 | 31 |

> **Note**
>
> The wrapper procedure generation approach to call PAL functions is still supported. For more information, refer to the older versions (prior to SAP HANA Platform 2.0 SPS 02) of this document.

## 5.8.15 Sampling

In business scenarios the number of records in the database is usually quite large, and it is common to use a small portion of the records as representatives, so that a rough impression of the dataset can be given by analyzing sampling.

This release of PAL provides eight sampling methods, including:

- First_N
- Middle_N
- Last_N
- Every_$N_{th}$
- SimpleRandom_WithReplacement
- SimpleRandom_WithoutReplacement
- Systematic
- Stratified

### Prerequisites

The input data does not contain null value.

# _SYS_AFL.PAL_SAMPLING

This function takes samples from a population.

### Overview of all Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<RESULT table>` | OUT |

### Input Table

| Table | Column | Reference for Output Table | Data Type | Description |
|---|---|---|---|---|
| DATA | Feature columns | DATA_FEATURES | INTEGER, VARCHAR, NVARCHAR, DOUBLE, or DECIMAL(p,s) | Any data users need |

### Parameter Table

Mandatory Parameters

The following parameters are mandatory and must be given a value.

| Name | Data Type | Description | Dependency |
|---|---|---|---|
| SAMPLING_METHOD | INTEGER | Sampling method:<br>• 0: First_*N*<br>• 1: Middle_*N*<br>• 2: Last_*N*<br>• 3: Every_*N*th<br>• 4: SimpleRandom_WithReplacement<br>• 5: SimpleRandom_WithoutReplacement<br>• 6: Systematic<br>• 7: Stratified_WithReplacement<br>• 8: Stratified_WithoutReplacement<br>**Note**: For the random methods (method 4, 5, 6 in the above list), the system time is used for the seed. | |
| INTERVAL | INTEGER | The interval between two samples. | Only required when SAMPLING_METHOD is 3. If this parameter is not specified, the SAMPLING_SIZE parameter will be used. |
| COLUMN_CHOOSE | INTEGER | The column that is used to do the stratified sampling. | Only required when SAMPLING_METHOD is 7 or 8.<br>The value is between [1, the data size]. |

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| SAMPLING_SIZE | INTEGER | 1 | Number of the samples. |
| RANDOM_SEED | INTEGER | 0 | Indicates the seed used to initialize the random number generator. It can be set to 0 or a positive value.<br>• 0: Uses the system time<br>• Not 0: Uses the specified seed |

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| PERCENTAGE | DOUBLE | 0.1 | Percentage of the samples.<br>Use this parameter when SAMPLING_SIZE is not set.<br>If both SAMPLING_SIZE and PERCENTAGE are specified, PERCENTAGE takes precedence. |

### Output Table

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| RESULT | Columns | DATA_FEATURES (in input table) data type | DATA_FEATURES (in input table) column name | The Output Table has the same structure as defined in the Input Table. |

### Examples

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

### Example 1

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_SAMPLING_DATA_TBL;
CREATE COLUMN TABLE PAL_SAMPLING_DATA_TBL (
    "EMPNO" INTEGER,
    "GENDER" VARCHAR (50),
    "INCOME" DOUBLE
);
INSERT INTO PAL_SAMPLING_DATA_TBL VALUES (1, 'male', 4000.5);
INSERT INTO PAL_SAMPLING_DATA_TBL VALUES (2, 'male', 5000.7);
INSERT INTO PAL_SAMPLING_DATA_TBL VALUES (3, 'female', 5100.8);
INSERT INTO PAL_SAMPLING_DATA_TBL VALUES (4, 'male', 5400.9);
INSERT INTO PAL_SAMPLING_DATA_TBL VALUES (5, 'female', 5500.2);
INSERT INTO PAL_SAMPLING_DATA_TBL VALUES (6, 'male', 5540.4);
INSERT INTO PAL_SAMPLING_DATA_TBL VALUES (7, 'male', 4500.9);
INSERT INTO PAL_SAMPLING_DATA_TBL VALUES (8, 'female', 6000.8);
INSERT INTO PAL_SAMPLING_DATA_TBL VALUES (9, 'male', 7120.8);
INSERT INTO PAL_SAMPLING_DATA_TBL VALUES (10, 'female', 8120.9);
INSERT INTO PAL_SAMPLING_DATA_TBL VALUES (11, 'female', 7453.9);
INSERT INTO PAL_SAMPLING_DATA_TBL VALUES (12, 'male', 7643.8);
INSERT INTO PAL_SAMPLING_DATA_TBL VALUES (13, 'male', 6754.3);
INSERT INTO PAL_SAMPLING_DATA_TBL VALUES (14, 'male', 6759.9);
INSERT INTO PAL_SAMPLING_DATA_TBL VALUES (15, 'male', 9876.5);
INSERT INTO PAL_SAMPLING_DATA_TBL VALUES (16, 'female', 9873.2);
INSERT INTO PAL_SAMPLING_DATA_TBL VALUES (17, 'male', 9889.9);
INSERT INTO PAL_SAMPLING_DATA_TBL VALUES (18, 'male', 9910.4);
INSERT INTO PAL_SAMPLING_DATA_TBL VALUES (19, 'male', 7809.3);
INSERT INTO PAL_SAMPLING_DATA_TBL VALUES (20, 'female', 8705.7);
INSERT INTO PAL_SAMPLING_DATA_TBL VALUES (21, 'male', 8756.0);
```

```
INSERT INTO PAL_SAMPLING_DATA_TBL VALUES (22, 'female', 7843.2);
INSERT INTO PAL_SAMPLING_DATA_TBL VALUES (23, 'male', 8576.9);
INSERT INTO PAL_SAMPLING_DATA_TBL VALUES (24, 'male', 9560.9);
INSERT INTO PAL_SAMPLING_DATA_TBL VALUES (25, 'female', 8794.9);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SAMPLING_METHOD', 0, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SAMPLING_SIZE', 8, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('INTERVAL', 5, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('COLUMN_CHOOSE', 2, NULL, NULL);
CALL "_SYS_AFL"."PAL_SAMPLING"(PAL_SAMPLING_DATA_TBL, #PAL_PARAMETER_TBL, ?);
TRUNCATE TABLE #PAL_PARAMETER_TBL;
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SAMPLING_METHOD', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SAMPLING_SIZE', 8, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('INTERVAL', 5, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('COLUMN_CHOOSE', 2, NULL, NULL);
CALL "_SYS_AFL"."PAL_SAMPLING"(PAL_SAMPLING_DATA_TBL, #PAL_PARAMETER_TBL, ?);
TRUNCATE TABLE #PAL_PARAMETER_TBL;
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SAMPLING_METHOD', 2, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SAMPLING_SIZE', 8, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('INTERVAL', 5, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('COLUMN_CHOOSE', 2, NULL, NULL);
CALL "_SYS_AFL"."PAL_SAMPLING"(PAL_SAMPLING_DATA_TBL, #PAL_PARAMETER_TBL, ?);
TRUNCATE TABLE #PAL_PARAMETER_TBL;
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SAMPLING_METHOD', 3, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SAMPLING_SIZE', 8, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('INTERVAL', 5, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('COLUMN_CHOOSE', 2, NULL, NULL);
CALL "_SYS_AFL"."PAL_SAMPLING"(PAL_SAMPLING_DATA_TBL, #PAL_PARAMETER_TBL, ?);
TRUNCATE TABLE #PAL_PARAMETER_TBL;
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SAMPLING_METHOD', 4, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SAMPLING_SIZE', 8, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('INTERVAL', 5, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('COLUMN_CHOOSE', 2, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('RANDOM_SEED', 10, NULL, NULL);
CALL "_SYS_AFL"."PAL_SAMPLING"(PAL_SAMPLING_DATA_TBL, #PAL_PARAMETER_TBL, ?);
TRUNCATE TABLE #PAL_PARAMETER_TBL;
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SAMPLING_METHOD', 5, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SAMPLING_SIZE', 8, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('INTERVAL', 5, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('COLUMN_CHOOSE', 2, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('RANDOM_SEED', 10, NULL, NULL);
CALL "_SYS_AFL"."PAL_SAMPLING"(PAL_SAMPLING_DATA_TBL, #PAL_PARAMETER_TBL, ?);
TRUNCATE TABLE #PAL_PARAMETER_TBL;
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SAMPLING_METHOD', 6, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SAMPLING_SIZE', 8, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('INTERVAL', 5, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('COLUMN_CHOOSE', 2, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('RANDOM_SEED', 10, NULL, NULL);
CALL "_SYS_AFL"."PAL_SAMPLING"(PAL_SAMPLING_DATA_TBL, #PAL_PARAMETER_TBL, ?);
TRUNCATE TABLE #PAL_PARAMETER_TBL;
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SAMPLING_METHOD', 7, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SAMPLING_SIZE', 8, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('INTERVAL', 5, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('COLUMN_CHOOSE', 2, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('RANDOM_SEED', 0, NULL, NULL);
CALL "_SYS_AFL"."PAL_SAMPLING"(PAL_SAMPLING_DATA_TBL, #PAL_PARAMETER_TBL, ?);
TRUNCATE TABLE #PAL_PARAMETER_TBL;
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SAMPLING_METHOD', 0, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PERCENTAGE', NULL, 0.1, NULL);
CALL "_SYS_AFL"."PAL_SAMPLING"(PAL_SAMPLING_DATA_TBL, #PAL_PARAMETER_TBL, ?);
```

### Expected Result

RESULT:

If method is 0 and SAMPLING_SIZE is 8:

| | EMPNO | GENDER | INCOME |
|---|---|---|---|
| 1 | 1 | male | 4,000.5 |
| 2 | 2 | male | 5,000.7 |
| 3 | 3 | female | 5,100.8 |
| 4 | 4 | male | 5,400.9 |
| 5 | 5 | female | 5,500.2 |
| 6 | 6 | male | 5,540.4 |
| 7 | 7 | male | 4,500.9 |
| 8 | 8 | female | 6,000.8 |

If method is 1 and SAMPLING_SIZE is 8:

| | EMPNO | GENDER | INCOME |
|---|---|---|---|
| 1 | 10 | female | 8,120.9 |
| 2 | 11 | female | 7,453.9 |
| 3 | 12 | male | 7,643.8 |
| 4 | 13 | male | 6,754.3 |
| 5 | 14 | male | 6,759.9 |
| 6 | 15 | male | 9,876.5 |
| 7 | 16 | female | 9,873.2 |
| 8 | 17 | male | 9,889.9 |

If method is 2 and SAMPLING_SIZE is 8:

| | EMPNO | GENDER | INCOME |
|---|---|---|---|
| 1 | 25 | female | 8,794.9 |
| 2 | 24 | male | 9,560.9 |
| 3 | 23 | male | 8,576.9 |
| 4 | 22 | female | 7,843.2 |
| 5 | 21 | male | 8,756 |
| 6 | 20 | female | 8,705.7 |
| 7 | 19 | male | 7,809.3 |
| 8 | 18 | male | 9,910.4 |

If method is 3 and INTERVAL is 5:

| | EMPNO | GENDER | INCOME |
|---|---|---|---|
| 1 | 5 | female | 5,500.2 |
| 2 | 10 | female | 8,120.9 |
| 3 | 15 | male | 9,876.5 |
| 4 | 20 | female | 8,705.7 |
| 5 | 25 | female | 8,794.9 |

If method is 4, RANDOM_SEED is 10, and SAMPLING_SIZE is 8:

| | EMPNO | GENDER | INCOME |
|---|---|---|---|
| 1 | 2 | male | 5,000.7 |
| 2 | 13 | male | 6,754.3 |
| 3 | 6 | male | 5,540.4 |
| 4 | 7 | male | 4,500.9 |
| 5 | 15 | male | 9,876.5 |
| 6 | 20 | female | 8,705.7 |
| 7 | 13 | male | 6,754.3 |
| 8 | 11 | female | 7,453.9 |

If method is 5, RANDOM_SEED is 10, and SAMPLING_SIZE is 8:

| | EMPNO | GENDER | INCOME |
|---|---|---|---|
| 1 | 2 | male | 5,000.7 |
| 2 | 13 | male | 6,754.3 |
| 3 | 7 | male | 4,500.9 |
| 4 | 8 | female | 6,000.8 |
| 5 | 17 | male | 9,889.9 |
| 6 | 21 | male | 8,756 |
| 7 | 14 | male | 6,759.9 |
| 8 | 11 | female | 7,453.9 |

If method is 6, RANDOM_SEED is 10, and SAMPLING_SIZE is 8:

| | EMPNO | GENDER | INCOME |
|---|---|---|---|
| 1 | 1 | male | 4,000.5 |
| 2 | 4 | male | 5,400.9 |
| 3 | 7 | male | 4,500.9 |
| 4 | 10 | female | 8,120.9 |
| 5 | 13 | male | 6,754.3 |
| 6 | 16 | female | 9,873.2 |
| 7 | 19 | male | 7,809.3 |
| 8 | 22 | female | 7,843.2 |

If method is 7 and SAMPLING_SIZE is 8:

| | EMPNO | GENDER | INCOME |
|---|---|---|---|
| 1 | 21 | male | 8,756 |
| 2 | 1 | male | 4,000.5 |
| 3 | 14 | male | 6,759.9 |
| 4 | 14 | male | 6,759.9 |
| 5 | 24 | male | 9,560.9 |
| 6 | 5 | female | 5,500.2 |
| 7 | 8 | female | 6,000.8 |
| 8 | 8 | female | 6,000.8 |

If method is 0 and PERCENTAGE is 0.1:

| | EMPNO | GENDER | INCOME |
|---|---|---|---|
| 1 | 1 | male | 4,000.5 |
| 2 | 2 | male | 5,000.7 |
| 3 | 3 | female | 5,100.8 |

**Example 2**

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_SAMPLING_DATA_TBL;
CREATE COLUMN TABLE PAL_SAMPLING_DATA_TBL (
    "EMPNO" INTEGER,
    "GENDER" VARCHAR (50),
    "DEP" VARCHAR (50),
    "LOC" INTEGER,
    "INCOME" DOUBLE
);
INSERT INTO PAL_SAMPLING_DATA_TBL VALUES (1, 'F', 'DEV', 1, 4000.5);
INSERT INTO PAL_SAMPLING_DATA_TBL VALUES (2, 'M', 'DEV', 1, 5000.7);
INSERT INTO PAL_SAMPLING_DATA_TBL VALUES (3, 'F', 'SALES', 1, 5100.8);
INSERT INTO PAL_SAMPLING_DATA_TBL VALUES (4, 'M', 'SALES', 1, 5400.9);
INSERT INTO PAL_SAMPLING_DATA_TBL VALUES (5, 'F', 'DEV', 0, 5500.2);
INSERT INTO PAL_SAMPLING_DATA_TBL VALUES (6, 'M', 'DEV', 0, 5540.4);
INSERT INTO PAL_SAMPLING_DATA_TBL VALUES (7, 'F', 'SALES', 0, 4500.9);
INSERT INTO PAL_SAMPLING_DATA_TBL VALUES (8, 'M', 'SALES', 0, 6000.8);
INSERT INTO PAL_SAMPLING_DATA_TBL VALUES (9, 'F', 'DEV', 1, 7120.8);
INSERT INTO PAL_SAMPLING_DATA_TBL VALUES (10, 'M', 'DEV', 1, 8120.9);
INSERT INTO PAL_SAMPLING_DATA_TBL VALUES (11, 'F', 'SALES', 1, 7453.9);
INSERT INTO PAL_SAMPLING_DATA_TBL VALUES (12, 'M', 'SALES', 1, 7643.8);
INSERT INTO PAL_SAMPLING_DATA_TBL VALUES (13, 'F', 'DEV', 0, 6754.3);
INSERT INTO PAL_SAMPLING_DATA_TBL VALUES (14, 'M', 'DEV', 0, 6759.9);
INSERT INTO PAL_SAMPLING_DATA_TBL VALUES (15, 'F', 'SALES', 0, 9876.5);
INSERT INTO PAL_SAMPLING_DATA_TBL VALUES (16, 'M', 'SALES', 0, 9873.2);
INSERT INTO PAL_SAMPLING_DATA_TBL VALUES (17, 'F', 'DEV', 1, 9889.9);
INSERT INTO PAL_SAMPLING_DATA_TBL VALUES (18, 'M', 'DEV', 1, 9910.4);
INSERT INTO PAL_SAMPLING_DATA_TBL VALUES (19, 'F', 'SALES', 1, 7809.3);
INSERT INTO PAL_SAMPLING_DATA_TBL VALUES (20, 'M', 'SALES', 1, 8705.7);
INSERT INTO PAL_SAMPLING_DATA_TBL VALUES (21, 'F', 'DEV', 0, 8756.0);
INSERT INTO PAL_SAMPLING_DATA_TBL VALUES (22, 'M', 'DEV', 0, 7843.2);
INSERT INTO PAL_SAMPLING_DATA_TBL VALUES (23, 'F', 'SALES', 0, 8576.9);
INSERT INTO PAL_SAMPLING_DATA_TBL VALUES (24, 'M', 'SALES', 0, 9560.9);
INSERT INTO PAL_SAMPLING_DATA_TBL VALUES (25, 'F', 'DEV', 1, 8794.9);
INSERT INTO PAL_SAMPLING_DATA_TBL VALUES (26, 'M', 'DEV', 1, 9873.2);
INSERT INTO PAL_SAMPLING_DATA_TBL VALUES (27, 'F', 'SALES', 1, 9889.9);
INSERT INTO PAL_SAMPLING_DATA_TBL VALUES (28, 'M', 'SALES', 1, 9910.4);
INSERT INTO PAL_SAMPLING_DATA_TBL VALUES (29, 'F', 'DEV', 0, 7809.3);
```

```
INSERT INTO PAL_SAMPLING_DATA_TBL VALUES (30, 'M', 'DEV', 0, 8705.7);
INSERT INTO PAL_SAMPLING_DATA_TBL VALUES (31, 'F', 'SALES', 0, 8756.0);
INSERT INTO PAL_SAMPLING_DATA_TBL VALUES (32, 'M', 'SALES', 0, 7843.2);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SAMPLING_METHOD', 7, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SAMPLING_SIZE', 8, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('INTERVAL', 5, NULL, NULL);
INSERT into #PAL_PARAMETER_TBL VALUES ('PERCENTAGE', null, 0.5, null);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('COLUMN_CHOOSE', 2, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('COLUMN_CHOOSE', 3, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('COLUMN_CHOOSE', 4, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('RANDOM_SEED', 10, NULL, NULL);
CALL "_SYS_AFL"."PAL_SAMPLING"(PAL_SAMPLING_DATA_TBL, #PAL_PARAMETER_TBL, ?);
```

**Expected Result**

| | EMPNO | GENDER | DEP | LOC | INCOME |
|---|---|---|---|---|---|
| 1 | 1 | F | DEV | 1 | 4,000.5 |
| 2 | 9 | F | DEV | 1 | 7,120.8 |
| 3 | 2 | M | DEV | 1 | 5,000.7 |
| 4 | 10 | M | DEV | 1 | 8,120.9 |
| 5 | 19 | F | SALES | 1 | 7,809.3 |
| 6 | 27 | F | SALES | 1 | 9,889.9 |
| 7 | 20 | M | SALES | 1 | 8,705.7 |
| 8 | 12 | M | SALES | 1 | 7,643.8 |
| 9 | 13 | F | DEV | 0 | 6,754.3 |
| 10 | 21 | F | DEV | 0 | 8,756 |
| 11 | 6 | M | DEV | 0 | 5,540.4 |
| 12 | 14 | M | DEV | 0 | 6,759.9 |
| 13 | 7 | F | SALES | 0 | 4,500.9 |
| 14 | 7 | F | SALES | 0 | 4,500.9 |
| 15 | 16 | M | SALES | 0 | 9,873.2 |
| 16 | 16 | M | SALES | 0 | 9,873.2 |

> **Note**
>
> The wrapper procedure generation approach to call PAL functions is still supported. For more information, refer to the older versions (prior to SAP HANA Platform 2.0 SPS 02) of this document.

## 5.8.16 Scale

In real world scenarios the collected continuous attributes are usually distributed within different ranges. It is a common practice to have the data well scaled so that data mining algorithms like neural networks, nearest neighbor classification and clustering can give more reliable results.

This release of PAL provides three scaling methods described below. In the following, $X_{ip}$ and $Y_{ip}$ are the original value and transformed value of the i-th record and p-th attribute, respectively.

1. **Min-Max Normalization**
   Each transformed value is within the range [new_minA, new_maxA], where `new_minA` and `new_maxA` are user-specified parameters. Supposing that `minA` and `maxA` are the minimum and maximum values of attribute A, we get the following calculation formula:
   $Y_{ip}$ = ($X_{ip}$ − minA) × (new_maxA - new_minA) / (maxA - minA) + new_minA
2. **Z-Score Normalization** (or zero-mean normalization).
   PAL uses three z-score methods.
   - **Mean-Standard Deviation**
     The transformed values have mean 0 and standard deviation 1. The transformation is made as follows:

     $$Y_{ip} = \frac{X_{ip} - \mu_p}{\sigma_p}$$

     Where $\mu_p$ and $\sigma_p$ are mean and standard deviations of the original values of the *p*-th attributes.
   - **Mean-Mean Absolute Deviation**
     The transformation is made as follows:

     $$Y_{ip} = \frac{X_{ip} - \mu_p}{s_p}$$

     $$\mu_p = \frac{1}{N}\sum_{i=1}^{N} X_{ip}$$

     $$s_p = \frac{1}{N}\sum_{i=1}^{N} |X_{ip} - \mu_p|$$

   - **Median-Median Absolute Deviation**
     The transformation is made as follows:

     $$Y_{ip} = \frac{X_{ip} - m_p}{s_p}$$

     $$m_p = \text{median}[X_{1p}, X_{2p}, \ldots, X_{np}]$$

     $$s_p = \text{median}[|X_{1p} - m_p|, |X_{2p} - m_p|, \ldots, |X_{np} - m_p|]$$

3. **Normalization by Decimal Scaling**

This method transforms the data by moving the decimal point of the values, so that the maximum absolute value for each attribute is less than or equal to 1. Mathematically, $Y_{ip} = X_{ip} \times 10^{K_p}$ for each attribute p, where $K_p$ is selected so that
$\max(|Y_{1p}|, |Y_{2p}|, ..., |Y_{np}|) \leq 1$

## Prerequisites

- The input data does not contain null value.
- The data is numeric, not categorical.

## _SYS_AFL.PAL_SCALE

This function normalizes the data.

### Overview of all Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<RESULT table>` | OUT |
| 4 | `<MODEL table>` | OUT |
| 5 | `<STATISTICS table>` | OUT |
| 6 | `<PLACEHOLDER table>` | OUT |

### Input Table

| Table | Column | Reference for Output Table | Data Type | Description |
|---|---|---|---|---|
| DATA | 1st column | DATA_ID | INTEGER, VARCHAR, or NVARCHAR | ID |
| | Other columns | DATA_FEATURES | INTEGER, DOUBLE, or DECIMAL(p, s) | Variable *Xn* |

### Parameter Table

Mandatory Parameters

The following parameters are mandatory and must be given a value.

| Name | Data Type | Description | Dependency |
|---|---|---|---|
| SCALING_METHOD | INTEGER | Scaling method:<br>• 0: Min-max normalization<br>• 1: Z-Score normalization<br>• 2: Decimal scaling normalization | |
| Z-SCORE_METHOD | INTEGER | • 0: Mean-Standard deviation<br>• 1: Mean-Mean absolute deviation<br>• 2: Median-Median absolute deviation | Only valid when SCALING_METHOD is 1. |
| NEW_MAX | DOUBLE | The new maximum value of the min-max normalization method | Only valid when SCALING_METHOD is 0. |
| NEW_MIN | DOUBLE | The new minimum value of min-max normalization method | Only valid when SCALING_METHOD is 0. |

Optional Parameter

The following parameter is optional. If it is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| THREAD_RATIO | DOUBLE | 0 | Specifies the ratio of total number of threads that can be used by this function. The value range is from 0 to 1, where 0 means only using 1 thread, and 1 means using at most all the currently available threads. Values outside the range will be ignored and this function heuristically determines the number of threads to use. |

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| DIVISION_BY_ZERO_HANDLER | INTEGER | 1 | • 0: Ignores the column when encountering a division by zero. The column is not scaled.<br>• 1: Throws an error when encountering a division by zero. |

### Output Table

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| RESULT | 1st column | DATA_ID (in input table) data type | DATA_ID (in input table) column name | ID |
| | 2nd column | DATA_FEATURES (in input table) data type | DATA_FEATURES (in input table) column name | Variable X*n* |
| MODEL | 1st column | INTEGER | ID | Scaling model ID |
| | 2nd column | NVARCHAR | MODEL_CONTENT | Binning model saved as JSON string<br>The table must be a column table. The minimum length of each unit (row) is 5000. |
| STATISTICS | 1st column | NVARCHAR(256) | STAT_NAME | Statistics |
| | 2nd column | NVARCHAR(1000) | STAT_VALUE | |
| PLACEHOLDER | 1st column | NVARCHAR(256) | PARAM_NAME | Placeholder for future features |
| | 2nd column | INTEGER | INT_VALUE | |
| | 3rd column | DOUBLE | DOUBLE_VALUE | |
| | 4th column | NVARCHAR(1000) | STRING_VALUE | |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA DM_PAL;
```

```
DROP TABLE PAL_SCALING_DATA_TBL;
CREATE COLUMN TABLE PAL_SCALING_DATA_TBL (
    "ID" INTEGER,
    "X1" DOUBLE,
    "X2" DOUBLE
);
INSERT INTO PAL_SCALING_DATA_TBL VALUES (0, 6.0, 9.0) ;
INSERT INTO PAL_SCALING_DATA_TBL VALUES (1, 12.1, 8.3) ;
INSERT INTO PAL_SCALING_DATA_TBL VALUES (2, 13.5, 15.3) ;
INSERT INTO PAL_SCALING_DATA_TBL VALUES (3, 15.4, 18.7) ;
INSERT INTO PAL_SCALING_DATA_TBL VALUES (4, 10.2, 19.8) ;
INSERT INTO PAL_SCALING_DATA_TBL VALUES (5, 23.3, 20.6) ;
INSERT INTO PAL_SCALING_DATA_TBL VALUES (6, 24.4,24.3) ;
INSERT INTO PAL_SCALING_DATA_TBL VALUES (7, 30.6, 25.3) ;
INSERT INTO PAL_SCALING_DATA_TBL VALUES (8, 32.5, 27.6) ;
INSERT INTO PAL_SCALING_DATA_TBL VALUES (9, 25.6, 28.5) ;
INSERT INTO PAL_SCALING_DATA_TBL VALUES (10, 38.7, 29.4) ;
INSERT INTO PAL_SCALING_DATA_TBL VALUES (11, 38.7, 29.4) ;
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SCALING_METHOD',0,NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('Z-SCORE_METHOD',0, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('NEW_MAX', NULL,1.0, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('NEW_MIN', NULL,0.0, NULL);
CALL "_SYS_AFL"."PAL_SCALE"(PAL_SCALING_DATA_TBL,
#PAL_PARAMETER_TBL, ?, ?, ?, ?);
```

### Expected Result

RESULT:

| | ID | X1 | X2 |
|---|---|---|---|
| 1 | 0 | 0 | 0.033175355450236935 |
| 2 | 1 | 0.18654434250764523 | 0 |
| 3 | 2 | 0.22935779816513757 | 0.3317535545023697 |
| 4 | 3 | 0.2874617737003058 | 0.4928909952606635 |
| 5 | 4 | 0.12844036697247702 | 0.5450236966824645 |
| 6 | 5 | 0.529051987767584 | 0.5829383886255924 |
| 7 | 6 | 0.5626911314984708 | 0.7582938388625593 |
| 8 | 7 | 0.7522935779816513 | 0.8056872037914693 |
| 9 | 8 | 0.8103975535168194 | 0.9146919431279622 |
| 10 | 9 | 0.599388379204893 | 0.957345971563981 |
| 11 | 10 | 1 | 1 |
| 12 | 11 | 1 | 1 |

MODEL:

| | ID | MODEL_CONTENT |
|---|---|---|
| 1 | 0 | {"Control":{"DIVISION_BY_ZERO_HANDLER":1,"NEW_MAX":1.0,"NEW_MIN":0.0,"SCALING_METHOD":0},"Scalin... |

STATISTICS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | X1.MIN | 6 |
| 2 | X1.MAX | 38.7 |
| 3 | X2.MIN | 8.3 |
| 4 | X2.MAX | 29.4 |

PLACEHOLDER: reserved for future use

> **Note**
>
> The wrapper procedure generation approach to call PAL functions is still supported. For more information, refer to the older versions (prior to SAP HANA Platform 2.0 SPS 02) of this document.

> **Note**
>
> The output RESULT table's columns DATA_FEATURES are of the same structure as the input DATA table's DATA_FEATURES columns. If the input columns are of type INTEGERS, the output columns are also of type INTEGERS. It could cause loss of precision. In order to make the output columns be of type DOUBLE, the input columns should be of type DOUBLE.

## 5.8.17 Scale with Model

Scale with Model is used to scale data based on the previous scaling model generated by the scale procedure.

It is assumed that new data is from similar distribution and will not update the scaling model.

### Prerequisites

- No missing or null data in the inputs.
- Data types must be identical to those in the scale procedure.
- The data types of the ID columns in the data input table and the result output table must be identical.

### _SYS_AFL.PAL_SCALE_WITH_MODEL

This function directly scales data based on the previous scaling model, without running the scale procedure once more.

**Overview of all Tables**

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<MODEL table>` | IN |
| 3 | `<PARAMETER table>` | IN |
| 4 | `<RESULT table>` | OUT |

### Input Tables

| Table | Column | Reference for Output Table | Data Type | Description |
|---|---|---|---|---|
| DATA | 1st column | DATA_ID | INTEGER, VARCHAR, or NVARCHAR | ID |
| | Feature columns | DATA_FEATURES | INTEGER, DOUBLE, or DECIMAL(p, s) | Raw data |
| MODEL | 1st column | | INTEGER | Scaling model ID |
| | 2nd column | | VARCHAR, or NVARCHAR | Scaling model saved as JSON string |

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| THREAD_RATIO | DOUBLE | 0 | Specifies the ratio of total number of threads that can be used by this function. The value range is from 0 to 1, where 0 means only using one thread, and 1 means using at most all the available threads. Values outside the range are ignored and this function heuristically determines the number of threads to use. |

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| DIVISION_BY_ZERO_HANDLER | INTEGER | The value in the scale model | • 0: Ignores the column when encountering a division by zero. The column is not scaled.<br>• 1: Throws an error when encountering a division by zero. |

### Output Table

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| RESULT | 1st column | DATA_ID (in input table) data type | DATA_ID (in input table) column name | ID |
| | Other columns | DATA_FEATURES (in input table) data type | DATA_FEATURES (in input table) column name | Scaled data |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_SCALING_DATA_TBL;
CREATE COLUMN TABLE PAL_SCALING_DATA_TBL (
    "ID" INT,
    "X1" DOUBLE,
    "X2" DOUBLE
);
INSERT INTO PAL_SCALING_DATA_TBL VALUES (0,   6.0, 9.0);
INSERT INTO PAL_SCALING_DATA_TBL VALUES (1,  12.1, 8.3);
INSERT INTO PAL_SCALING_DATA_TBL VALUES (2,  13.5, 15.3);
INSERT INTO PAL_SCALING_DATA_TBL VALUES (3,  15.4, 18.7);
INSERT INTO PAL_SCALING_DATA_TBL VALUES (4,  10.2, 19.8);
INSERT INTO PAL_SCALING_DATA_TBL VALUES (5,  23.3, 20.6);
INSERT INTO PAL_SCALING_DATA_TBL VALUES (6,  24.4, 24.3);
INSERT INTO PAL_SCALING_DATA_TBL VALUES (7,  30.6, 25.3);
INSERT INTO PAL_SCALING_DATA_TBL VALUES (8,  32.5, 27.6);
INSERT INTO PAL_SCALING_DATA_TBL VALUES (9,  25.6, 28.5);
INSERT INTO PAL_SCALING_DATA_TBL VALUES (10, 38.7, 29.4);
INSERT INTO PAL_SCALING_DATA_TBL VALUES (11, 38.7, 29.4);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SCALING_METHOD',1,NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('Z-SCORE_METHOD',0, NULL, NULL);
```

```
INSERT INTO #PAL_PARAMETER_TBL VALUES ('THREAD_NUMBER',2, NULL, NULL);
DROP TABLE PAL_SCALING_MODEL_TBL;
CREATE COLUMN TABLE PAL_SCALING_MODEL_TBL (
    "JID" INT,
    "JSMODEL" VARCHAR(5000)
);
CALL "_SYS_AFL"."PAL_SCALE"(PAL_SCALING_DATA_TBL, #PAL_PARAMETER_TBL, ?,
PAL_SCALING_MODEL_TBL, ?, ?) WITH OVERVIEW;
------ Scaling with Model --------
DROP TABLE PAL_SCALING_DATA_TBL;
CREATE COLUMN TABLE PAL_SCALING_DATA_TBL (
    "ID" INTEGER,
    "S_X1" DOUBLE,
    "S_X2" DOUBLE
);
INSERT INTO PAL_SCALING_DATA_TBL VALUES (0 ,6, 9);
INSERT INTO PAL_SCALING_DATA_TBL VALUES (1 , 6,7);
INSERT INTO PAL_SCALING_DATA_TBL VALUES (2, 4, 4);
INSERT INTO PAL_SCALING_DATA_TBL VALUES (3, 1, 2);
INSERT INTO PAL_SCALING_DATA_TBL VALUES (4, 9,-2);
INSERT INTO PAL_SCALING_DATA_TBL VALUES (5, 4, 5.0);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
CALL "_SYS_AFL"."PAL_SCALE_WITH_MODEL"(PAL_SCALING_DATA_TBL,
PAL_SCALING_MODEL_TBL, #PAL_PARAMETER_TBL, ?);
```

**Expected Result**

RESULT:

| | ID | S_X1 | S_X2 |
|---|---|---|---|
| 1 | 0 | -1.4873873740932577 | -1.6536838199986237 |
| 2 | 1 | -1.4873873740932577 | -1.9214868677716803 |
| 3 | 2 | -1.6667707759939523 | -2.3231914394312647 |
| 4 | 3 | -1.935845878844994 | -2.5909944872043216 |
| 5 | 4 | -1.218312271242216 | -3.1266005827504344 |
| 6 | 5 | -1.6667707759939523 | -2.1892899155447365 |

> **Note**
>
> The wrapper procedure generation approach to call PAL functions is still supported. For more information, refer to the older versions (prior to SAP HANA Platform 2.0 SPS 02) of this document.

## Related Information

Scale [page 1167]

## 5.8.18 SMOTE

Synthetic minority over-sampling technique (SMOTE) proposes an over-sampling approach in which the minority class is over-sampled by creating "synthetic" examples in "feature space".

The minority class is over-sampled by taking each minority class sample and introducing synthetic examples along the line segments joining any/all of the k minority class nearest neighbors. Depending upon the amount of over-sampling required, neighbors from the k nearest neighbors are randomly chosen. For instance, if the amount of over-sampling needed is 200%, only two neighbors from the five nearest neighbors are chosen and one sample is generated in the direction of each.

Synthetic samples are generated in the following way. Take the difference between the feature vector (sample) under consideration and its nearest neighbor. Multiply this difference by a random number between 0 and 1, and add it to the feature vector under consideration.

For categorical features, the distance calculation is aliened with PAL_KNN.The categorical feature is given the value occuring in the majority of the k-nearest neighbors.

### Prerequisites

- No missing or null data in input data.
- No ID column.

### _SYS_AFL.PAL_SMOTE

This function performs over-sampling by generating synthetic samples for minority classes.

**Overview of All Tables**

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<Result table>` | OUT |

**Input Table**

| Table | Column | Data Type | Description |
|---|---|---|---|
| DATA | Feature columns | INTEGER, DOUBLE, VARCHAR or NVARCHAR | Feature data can be continuous or categorical. |

| Table | Column | Data Type | Description |
|---|---|---|---|
| | Dependent variable column | INTEGER, VARCHAR or NVARCHAR | Dependent variable |

### Parameter Table

Mandatory Parameters

None

Optional Parameters

The following parameter is optional. If it is not specified. PAL will use its default value.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| THREAD_RATIO | DOUBLE | 0.0 | Specifies the ratio of total number of threads that can be used by this function. The value range is from 0 to 1, where 0 means only using 1 thread, and 1 means using at most all the currently available threads. Values outside the range will be ignored and this function heuristically determines the number of threads to use. | |
| RANDOM_SEED | INTEGER | 0 | Specifies the seed for random number generator.<br>• 0: Uses the current time (in second) as seed.<br>• Others: Uses the specified value as seed. | |
| K_NEAREST_NEIGHBOURS | INTEGER | 1 | Number of nearest neighbors (k). | |
| DEPENDENT_VARIABLE | NVARCHAR | No default value | Specifies the dependent variable by name, or the first column in data table is considered to be the dependent variable. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| MINORITY_CLASS | NVARCHAR | All classes except majority class are re-sampled to match the majority class sample amount. | Specifies which minority class value in dependent variable column will perform SMOTE. | |
| SMOTE_AMOUNT | INTEGER | The synthetic samples are generated until the minority class sample amount matches the majority class sample amount. | Amount of SMOTE, N%. For example, 200 means 200%, which means each minority class sample will generate two synthetic samples. | MINORITY_CLASS is presented by you. |
| CATEGORICAL_VARIABLE | NVARCHAR | No default value | Indicates whether or not a column data is actually corresponding to a category variable even the data type of this column is INTEGER.<br>By default, VARCHAR or NVARCHAR is category variable, and INTEGER or DOUBLE is continuous variable. | |
| VARIABLE_WEIGHT | NVARCHAR for variable name and DOUBLE for weight value | No default value | Specifies the weight of a variable participating in distance calculation. The value must be greater or equal to 0. Defaults to 1 for variables not specified. | |
| CATEGORY_WEIGHTS | DOUBLE | 0.707 | Represents the weight of category attributes. The value must be greater or equal to 0. | |
| METHOD | INTEGER | 0 | Searching method when finding K nearest neighbor.<br>• 0: Brute force searching<br>• 1: KD-tree searching | |

## Output Tables

| Name | Column | Data Type | Description |
|---|---|---|---|
| RESULT | Columns | DATA_FEATURES (in Input Table) data type | The Output Table has the same structure as defined in the Input Table. |

### Examples

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

Example 1: Given MINORITY_CLASS and SMOTE_AMOUNT

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_SMOTE_DATA_TBL;
CREATE COLUMN TABLE PAL_SMOTE_DATA_TBL (
    "X1" INTEGER,
    "X2" DOUBLE,
    "X3" DOUBLE,
    "TYPE" INTEGER
);
INSERT INTO PAL_SMOTE_DATA_TBL VALUES (2, 1, 3.5, 1);
INSERT INTO PAL_SMOTE_DATA_TBL VALUES (3, 10, 7.6, 1);
INSERT INTO PAL_SMOTE_DATA_TBL VALUES (3, 10, 5.5, 2);
INSERT INTO PAL_SMOTE_DATA_TBL VALUES (3, 10, 4.7, 1);
INSERT INTO PAL_SMOTE_DATA_TBL VALUES (7, 1000, 8.5, 1);
INSERT INTO PAL_SMOTE_DATA_TBL VALUES (8, 1000, 9.4, 2);
INSERT INTO PAL_SMOTE_DATA_TBL VALUES (6, 1000, 0.34, 1);
INSERT INTO PAL_SMOTE_DATA_TBL VALUES (8, 999, 7.4, 2);
INSERT INTO PAL_SMOTE_DATA_TBL VALUES (7, 999, 3.5, 1);
INSERT INTO PAL_SMOTE_DATA_TBL VALUES (6, 1000, 7, 1);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 0.1, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('RANDOM_SEED', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('DEPENDENT_VARIABLE', NULL, NULL, 'TYPE');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MINORITY_CLASS', NULL, NULL, '2');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SMOTE_AMOUNT', 200, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('K_NEAREST_NEIGHBOURS', 2, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('METHOD', 1, NULL, NULL);
DROP TABLE PAL_SMOTE_RESULT_TBL;
CREATE COLUMN TABLE PAL_SMOTE_RESULT_TBL (
    "X1" INTEGER,
    "X2" DOUBLE,
    "X3" DOUBLE,
    "TYPE" INTEGER
);
CALL _SYS_AFL.PAL_SMOTE (PAL_SMOTE_DATA_TBL, "#PAL_PARAMETER_TBL",
PAL_SMOTE_RESULT_TBL) WITH OVERVIEW;
SELECT * FROM PAL_SMOTE_RESULT_TBL;
```

### Expected Results

RESULT:

| | X1 | X2 | X3 | TYPE |
|---|---|---|---|---|
| 1 | 2 | 1 | 3.5 | 1 |
| 2 | 3 | 10 | 7.6 | 1 |
| 3 | 3 | 10 | 5.5 | 2 |
| 4 | 3 | 10 | 4.7 | 1 |
| 5 | 7 | 1,000 | 8.5 | 1 |
| 6 | 8 | 1,000 | 9.4 | 2 |
| 7 | 6 | 1,000 | 0.34 | 1 |
| 8 | 8 | 999 | 7.4 | 2 |
| 9 | 7 | 999 | 3.5 | 1 |
| 10 | 6 | 1,000 | 7 | 1 |
| 11 | 8 | 973.1... | 7.350... | 2 |
| 12 | 7 | 888.0... | 8.959... | 2 |
| 13 | 8 | 999.0... | 7.513... | 2 |
| 14 | 8 | 999.5... | 8.424... | 2 |
| 15 | 4 | 131.5... | 5.733... | 2 |
| 16 | 5 | 340.7... | 6.135... | 2 |

Example 2: When MINORITY_CLASS and SMOTE_AMOUNT are not presented, all classes except majority class are re-sampled to match the majority class sample amount

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_SMOTE_DATA_TBL;
CREATE COLUMN TABLE PAL_SMOTE_DATA_TBL (
    "X1" INTEGER,
    "X2" DOUBLE,
    "X3" DOUBLE,
    "TYPE" INTEGER
);
INSERT INTO PAL_SMOTE_DATA_TBL VALUES (2, 1, 3.5, 1);
INSERT INTO PAL_SMOTE_DATA_TBL VALUES (3, 10, 7.6, 1);
INSERT INTO PAL_SMOTE_DATA_TBL VALUES (3, 10, 5.5, 2);
INSERT INTO PAL_SMOTE_DATA_TBL VALUES (3, 10, 4.7, 1);
INSERT INTO PAL_SMOTE_DATA_TBL VALUES (7, 1000, 8.5, 1);
INSERT INTO PAL_SMOTE_DATA_TBL VALUES (8, 1000, 9.4, 2);
INSERT INTO PAL_SMOTE_DATA_TBL VALUES (6, 1000, 0.34, 1);
INSERT INTO PAL_SMOTE_DATA_TBL VALUES (8, 999, 7.4, 2);
INSERT INTO PAL_SMOTE_DATA_TBL VALUES (7, 999, 3.5, 1);
INSERT INTO PAL_SMOTE_DATA_TBL VALUES (6, 1000, 7, 1);
INSERT INTO PAL_SMOTE_DATA_TBL VALUES (3, 10, 5.5, 3);
INSERT INTO PAL_SMOTE_DATA_TBL VALUES (8, 999, 7.4, 3);
INSERT INTO PAL_SMOTE_DATA_TBL VALUES (8, 1000, 9.4, 3);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 0.1, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('RANDOM_SEED', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('DEPENDENT_VARIABLE', NULL, NULL, 'TYPE');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('K_NEAREST_NEIGHBOURS', 2, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('METHOD', 1, NULL, NULL);
DROP TABLE PAL_SMOTE_RESULT_TBL;
```

```
CREATE COLUMN TABLE PAL_SMOTE_RESULT_TBL (
    "X1" INTEGER,
    "X2" DOUBLE,
    "X3" DOUBLE,
    "TYPE" INTEGER
);
CALL _SYS_AFL.PAL_SMOTE (PAL_SMOTE_DATA_TBL, "#PAL_PARAMETER_TBL",
PAL_SMOTE_RESULT_TBL) WITH OVERVIEW;
SELECT * FROM PAL_SMOTE_RESULT_TBL;
```

**Expected Results**

RESULT:

| | X1 | X2 | X3 | TYPE |
|---|---|---|---|---|
| 1 | 2 | 1 | 3.5 | 1 |
| 2 | 3 | 10 | 7.6 | 1 |
| 3 | 3 | 10 | 5.5 | 2 |
| 4 | 3 | 10 | 4.7 | 1 |
| 5 | 7 | 1,000 | 8.5 | 1 |
| 6 | 8 | 1,000 | 9.4 | 2 |
| 7 | 6 | 1,000 | 0.34 | 1 |
| 8 | 8 | 999 | 7.4 | 2 |
| 9 | 7 | 999 | 3.5 | 1 |
| 10 | 6 | 1,000 | 7 | 1 |
| 11 | 3 | 10 | 5.5 | 3 |
| 12 | 8 | 999 | 7.4 | 3 |
| 13 | 8 | 1,000 | 9.4 | 3 |
| 14 | 8 | 973.1... | 7.350... | 2 |
| 15 | 7 | 888.0... | 8.959... | 2 |
| 16 | 8 | 999.0... | 7.513... | 2 |
| 17 | 8 | 999.5... | 8.424... | 2 |
| 18 | 8 | 973.1... | 7.350... | 3 |
| 19 | 7 | 888.0... | 8.959... | 3 |
| 20 | 8 | 999.9... | 9.286... | 3 |
| 21 | 8 | 999.4... | 8.375... | 3 |

Example 3: When MINORITY_CLASS is presented and SMOTE_AMOUNT is not, the specified class is re-sampled to match the majority class sample amount

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_SMOTE_DATA_TBL;
CREATE COLUMN TABLE PAL_SMOTE_DATA_TBL (
    "X1" INTEGER,
    "X2" DOUBLE,
    "X3" DOUBLE,
    "TYPE" INTEGER
);
INSERT INTO PAL_SMOTE_DATA_TBL VALUES (2, 1, 3.5, 1);
INSERT INTO PAL_SMOTE_DATA_TBL VALUES (3, 10, 7.6, 1);
INSERT INTO PAL_SMOTE_DATA_TBL VALUES (3, 10, 5.5, 2);
INSERT INTO PAL_SMOTE_DATA_TBL VALUES (3, 10, 4.7, 1);
INSERT INTO PAL_SMOTE_DATA_TBL VALUES (7, 1000, 8.5, 1);
INSERT INTO PAL_SMOTE_DATA_TBL VALUES (8, 1000, 9.4, 2);
INSERT INTO PAL_SMOTE_DATA_TBL VALUES (6, 1000, 0.34, 1);
```

```
INSERT INTO PAL_SMOTE_DATA_TBL VALUES (8, 999, 7.4, 2);
INSERT INTO PAL_SMOTE_DATA_TBL VALUES (7, 999, 3.5, 1);
INSERT INTO PAL_SMOTE_DATA_TBL VALUES (6, 1000, 7, 1);
INSERT INTO PAL_SMOTE_DATA_TBL VALUES (3, 10, 5.5, 3);
INSERT INTO PAL_SMOTE_DATA_TBL VALUES (8, 999, 7.4, 3);
INSERT INTO PAL_SMOTE_DATA_TBL VALUES (8, 1000, 9.4, 3);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 0.1, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('RANDOM_SEED', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('DEPENDENT_VARIABLE', NULL, NULL, 'TYPE');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MINORITY_CLASS', NULL, NULL, '2');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('K_NEAREST_NEIGHBOURS', 2, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('METHOD', 1, NULL, NULL);
DROP TABLE PAL_SMOTE_RESULT_TBL;
CREATE COLUMN TABLE PAL_SMOTE_RESULT_TBL (
    "X1" INTEGER,
    "X2" DOUBLE,
    "X3" DOUBLE,
    "TYPE" INTEGER
);
CALL _SYS_AFL.PAL_SMOTE (PAL_SMOTE_DATA_TBL, "#PAL_PARAMETER_TBL",
PAL_SMOTE_RESULT_TBL) WITH OVERVIEW;
SELECT * FROM PAL_SMOTE_RESULT_TBL;
```

### Expected Results

RESULT:

| | X1 | X2 | X3 | TYPE |
|---|---|---|---|---|
| 1 | 2 | 1 | 3.5 | 1 |
| 2 | 3 | 10 | 7.6 | 1 |
| 3 | 3 | 10 | 5.5 | 2 |
| 4 | 3 | 10 | 4.7 | 1 |
| 5 | 7 | 1,000 | 8.5 | 1 |
| 6 | 8 | 1,000 | 9.4 | 2 |
| 7 | 6 | 1,000 | 0.34 | 1 |
| 8 | 8 | 999 | 7.4 | 2 |
| 9 | 7 | 999 | 3.5 | 1 |
| 10 | 6 | 1,000 | 7 | 1 |
| 11 | 3 | 10 | 5.5 | 3 |
| 12 | 8 | 999 | 7.4 | 3 |
| 13 | 8 | 1,000 | 9.4 | 3 |
| 14 | 8 | 973.1... | 7.350... | 2 |
| 15 | 7 | 888.0... | 8.959... | 2 |
| 16 | 8 | 999.0... | 7.513... | 2 |
| 17 | 8 | 999.5... | 8.424... | 2 |

**Example 4:**

Given a categorical feature:

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_SMOTE_DATA_TBL;
CREATE COLUMN TABLE PAL_SMOTE_DATA_TBL (
    "X1" INTEGER,
    "X2" DOUBLE,
    "X3" NVARCHAR(50),
    "TYPE" INTEGER
);
INSERT INTO PAL_SMOTE_DATA_TBL VALUES (2, 1, '3', 1);
INSERT INTO PAL_SMOTE_DATA_TBL VALUES (3, 10, '7', 1);
INSERT INTO PAL_SMOTE_DATA_TBL VALUES (3, 10, '7', 2);
INSERT INTO PAL_SMOTE_DATA_TBL VALUES (3, 10, '7', 1);
INSERT INTO PAL_SMOTE_DATA_TBL VALUES (6, 1000, '7', 2);
INSERT INTO PAL_SMOTE_DATA_TBL VALUES (7, 1000, '3', 1);
INSERT INTO PAL_SMOTE_DATA_TBL VALUES (8, 1000, '3', 2);
INSERT INTO PAL_SMOTE_DATA_TBL VALUES (6, 1000, '3', 1);
INSERT INTO PAL_SMOTE_DATA_TBL VALUES (8, 999, '7', 2);
INSERT INTO PAL_SMOTE_DATA_TBL VALUES (7, 999, '3', 1);
INSERT INTO PAL_SMOTE_DATA_TBL VALUES (6, 1000, '7', 1);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 0.1, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('RANDOM_SEED', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('DEPENDENT_VARIABLE', NULL, NULL, 'TYPE');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('K_NEAREST_NEIGHBOURS', 3, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('METHOD', 1, NULL, NULL);
DROP TABLE PAL_SMOTE_RESULT_TBL;
CREATE COLUMN TABLE PAL_SMOTE_RESULT_TBL (
"X1" INTEGER,
"X2" DOUBLE,
"X3" DOUBLE,
"TYPE" INTEGER
);
CALL _SYS_AFL.PAL_SMOTE (PAL_SMOTE_DATA_TBL, "#PAL_PARAMETER_TBL",
PAL_SMOTE_RESULT_TBL) WITH OVERVIEW;
SELECT * FROM PAL_SMOTE_RESULT_TBL;
```

**Expected Results**

RESULT:

| | X1 | X2 | X3 | TYPE |
|---|---|---|---|---|
| 1 | 2 | 1 | 3 | 1 |
| 2 | 3 | 10 | 7 | 1 |
| 3 | 3 | 10 | 7 | 2 |
| 4 | 3 | 10 | 7 | 1 |
| 5 | 7 | 1,000 | 3 | 1 |
| 6 | 8 | 1,000 | 3 | 2 |
| 7 | 6 | 1,000 | 3 | 1 |
| 8 | 8 | 999 | 7 | 2 |
| 9 | 7 | 999 | 3 | 1 |
| 10 | 6 | 1,000 | 7 | 1 |
| 11 | 6 | 1,000 | 7 | 2 |
| 12 | 8 | 974.0... | 7 | 2 |
| 13 | 4 | 121.9... | 7 | 2 |
| 14 | 8 | 999.9... | 7 | 2 |

## 5.8.19 SMOTETomek

Class to perform over-sampling using SMOTE and cleaning using Tomek links.

SMOTETomek combines over-sampling and under-sampling using SMOTE and Tomek links.

### Prerequisites

- No missing or null data in input data.
- No ID column.

### _SYS_AFL.PAL_SMOTETOMEK

This function performs over-sampling using SMOTE and cleaning using Tomek links.

**Overview of All Tables**

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |

| Position | Table Name | Parameter Type |
|---|---|---|
| 3 | `<Result table>` | OUT |

### Input Table

| Table | Column | Data Type | Description |
|---|---|---|---|
| DATA | Feature columns | INTEGER, DOUBLE, VARCHAR, or NVARCHAR | Feature data can be continuous or categorical |
| | Dependent variable column | INTEGER, VARCHAR or NVARCHAR | Dependent variable |

### Parameter Table

Mandatory Parameters

None

Optional Parameters

The following parameter is optional. If it is not specified. PAL will use its default value.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| THREAD_RATIO | DOUBLE | 0.0 | Specifies the ratio of total number of threads that can be used by this function. The value range is from 0 to 1, where 0 means only using 1 thread, and 1 means using at most all the currently available threads. Values outside the range will be ignored and this function heuristically determines the number of threads to use. | |
| RANDOM_SEED | INTEGER | 0 | Specifies the seed for random number generator.<br>• 0: Uses the current time (in second) as seed.<br>• Others: Uses the specified value as seed. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| K_NEAREST_NEIGHBOURS | INTEGER | 1 | Number of nearest neighbors (k). | |
| DEPENDENT_VARIABLE | NVARCHAR | No default value | Specifies the dependent variable by name, or the first column in data table is considered to be the dependent variable. | |
| MINORITY_CLASS | NVARCHAR | All classes except majority class are re-sampled to match the majority class sample amount. | Specifies which minority class value in dependent variable column will perform SMOTE. | |
| SMOTE_AMOUNT | INTEGER | The synthetic samples are generated until the minority class sample amount matches the majority class sample amount. | Amount of SMOTE, N%. For example, 200 means 200%, which means each minority class sample will generate two synthetic samples. | MINORITY_CLASS is presented by you. |
| CATEGORICAL_VARIABLE | NVARCHAR | No default value | Indicates whether or not a column data is actually corresponding to a category variable even the data type of this column is INTEGER.<br><br>By default, VARCHAR or NVARCHAR is category variable, and INTEGER or DOUBLE is continuous variable. | |
| VARIABLE_WEIGHT | NVARCHAR for variable name and DOUBLE for weight value | No default value | pecifies the weight of a variable participating in distance calculation. The value must be greater or equal to 0. Defaults to 1 for variables not specified. | |
| CATEGORY_WEIGHTS | DOUBLE | 0.707 | Represents the weight of category attributes. The value must be greater or equal to 0. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| METHOD | INTEGER | 0 | Searching method when finding K nearest neighbor.<br>• 0: Brute force searching<br>• 1: KD-tree searching | |
| SAMPLING_STRATEGY | INTEGER | 0 | Specify the class targeted by the resampling.<br>• 0: Resamples only the majority class.<br>• 1: Resamples all classes but the minority class.<br>• 2: Resamples all classes but the majority class.<br>• 3: Resamples all classes. | |

### Output Tables

| Name | Column | Data Type | Description |
|---|---|---|---|
| RESULT | Columns | DATA_FEATURES (in Input Table) data type | The Output Table has the same structure as defined in the Input Table. |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

Example 1

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_SMOTETOMEK_DATA_TBL;
CREATE COLUMN TABLE PAL_SMOTETOMEK_DATA_TBL (
    "X1" INTEGER,
    "X2" DOUBLE,
    "X3" DOUBLE,
    "TYPE" INTEGER
);
INSERT INTO PAL_SMOTETOMEK_DATA_TBL VALUES (2, 1, 3.5, 1);
INSERT INTO PAL_SMOTETOMEK_DATA_TBL VALUES (3, 10, 7.6, 1);
INSERT INTO PAL_SMOTETOMEK_DATA_TBL VALUES (3, 10, 5.5, 2);
INSERT INTO PAL_SMOTETOMEK_DATA_TBL VALUES (3, 10, 4.7, 1);
INSERT INTO PAL_SMOTETOMEK_DATA_TBL VALUES (7, 1000, 8.5, 1);
INSERT INTO PAL_SMOTETOMEK_DATA_TBL VALUES (8, 1000, 9.4, 2);
```

```
INSERT INTO PAL_SMOTETOMEK_DATA_TBL VALUES (6, 1000, 0.34, 1);
INSERT INTO PAL_SMOTETOMEK_DATA_TBL VALUES (8, 999, 7.4, 2);
INSERT INTO PAL_SMOTETOMEK_DATA_TBL VALUES (7, 999, 3.5, 1);
INSERT INTO PAL_SMOTETOMEK_DATA_TBL VALUES (6, 1000, 7, 1);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 0.1, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('RANDOM_SEED', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('DEPENDENT_VARIABLE', NULL, NULL, 'TYPE');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MINORITY_CLASS', NULL, NULL, '2');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SMOTE_AMOUNT', 200, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('K_NEAREST_NEIGHBOURS', 2, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('METHOD', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SAMPLING_STRATEGY', 3, NULL, NULL);
DROP TABLE PAL_SMOTETOMEK_RESULT_TBL;
CREATE COLUMN TABLE PAL_SMOTETOMEK_RESULT_TBL (
"X1" INTEGER,
"X2" DOUBLE,
"X3" DOUBLE,
"TYPE" INTEGER
);
CALL _SYS_AFL.PAL_SMOTETOMEK(PAL_SMOTETOMEK_DATA_TBL, "#PAL_PARAMETER_TBL",
PAL_SMOTETOMEK_RESULT_TBL) WITH OVERVIEW;
SELECT * FROM PAL_SMOTETOMEK_RESULT_TBL;
```

### Expected Results

RESULT:

| | X1 | X2 | X3 | TYPE |
|---|---|---|---|---|
| 1 | 2 | 1 | 3.5 | 1 |
| 2 | 3 | 10 | 7.6 | 1 |
| 3 | 7 | 1,000 | 8.5 | 1 |
| 4 | 8 | 1,000 | 9.4 | 2 |
| 5 | 6 | 1,000 | 0.34 | 1 |
| 6 | 8 | 999 | 7.4 | 2 |
| 7 | 7 | 999 | 3.5 | 1 |
| 8 | 6 | 1,000 | 7 | 1 |
| 9 | 8 | 973.1... | 7.350... | 2 |
| 10 | 7 | 888.0... | 8.959... | 2 |
| 11 | 8 | 999.0... | 7.513... | 2 |
| 12 | 8 | 999.5... | 8.424... | 2 |
| 13 | 4 | 131.5... | 5.733... | 2 |
| 14 | 5 | 340.7... | 6.135... | 2 |

## 5.8.20 Tomek's Links

TomekLinks detects the so-called Tomek's link. A Tomek's link between two samples of different class x and y is defined such that for any sample z:

`d(x,y) < d (x,z)` and `d(x,y) < d(y,z)`

where is the distance between the two samples. In some other words, a Tomek's link exist if the two samples are the nearest neighbors of each other.

### Prerequisites

- No missing or null data in input data.

### _SYS_AFL.PAL_TOMEKLINKS

This function performs under-sampling by removing Tomek's links.

**Overview of All Tables**

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<Result table>` | OUT |

**Input Table**

| Table | Column | Data Type | Description | Constraint |
|---|---|---|---|---|
| DATA | Feature columns | INTEGER, DOUBLE, VARCHAR, or NVARCHAR | Feature data can be continuous or categorical. | If the HAS_ID parameter is 1, the first column is ID column and can be INTEGER, VARCHAR, or NVARCHAR. |
| | Dependent variable column | INTEGER, VARCHAR or NVARCHAR | Dependent variabel | |

**Parameter Table**

Mandatory Parameters

None

Optional Parameters

The following parameter is optional. If it is not specified. PAL will use its default value.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| THREAD_RATIO | DOUBLE | 0 | Specifies the ratio of total number of threads that can be used by this function. The value range is from 0 to 1, where 0 means only using 1 thread, and 1 means using at most all the currently available threads. Values outside the range will be ignored and this function heuristically determines the number of threads to use. | |
| HAS_ID | INTEGER | 0 | Specifies whether the input data table has an ID column. If it exists, the ID column must be the first column.<br>• 0: indicates there is no ID column in the data table.<br>• 1: indicates that the first column of the data table is ID column. The algorithm will ignore the first column. | |
| DEPENDENT_VARIABLE | NVARCHAR | No default value | Specifies the dependent variable by name, or:<br>• HAS_ID = 0: the first column in data table is considered to be the dependent variable.<br>• HAS_ID = 1: the second column in data table is considered to be the dependent variable. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| SAMPLING_STRATEGY | INTEGER | 0 | Specify the class targeted by the resampling.<br>• 0: Resamples only the majority class.<br>• 1: Resamples all classes but the minority class.<br>• 2: Resamples all classes but the majority class.<br>• 3: Resamples all classes. | |
| CATEGORICAL_VARIABLE | NVARCHAR | No default value | Indicates whether or not a column data is actually corresponding to a category variable even the data type of this column is INTEGER.<br>By default, VARCHAR or NVARCHAR is category variable, and INTEGER or DOUBLE is continuous variable. | |
| VARIABLE_WEIGHT | NVARCHAR for variable name and DOUBLE for weight value | No default value | Specifies the weight of a variable participating in distance calculation. The value must be greater or equal to 0. Defaults to 1 for variables not specified. | |
| CATEGORY_WEIGHTS | DOUBLE | 0.707 | Represents the weight of category attributes. The value must be greater or equal to 0. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| DISTANCE_LEVEL | INTEGER | 2 | Computes the distance between the train data and the test data point.<br>• 1: Manhattan distance<br>• 2: Euclidean distance<br>• 3: Minkowski distance<br>• 4: Chebyshev distance<br>• 6: Cosine distance | |
| MINKOWSKI_POWER | DOUBLE | 3 | When you use the Minkowski distance, this parameter controls the value of power. | Only valid when DISTANCE_LEVEL is 3. |
| METHOD | INTEGER | 0 | Searching method when finding K nearest neighbor.<br>• 0: Brute force searching<br>• 1: KD-tree searching | |

### Output Tables

| Name | Column | Data Type | Description |
|---|---|---|---|
| RESULT | Columns | DATA_FEATURES (in Input Table) data type | The Output Table has the same structure as defined in the Input Table. |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

Example 1: No ID column and under-sampling strategy = 0

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_TOMEKLINKS_DATA_TBL;
CREATE COLUMN TABLE PAL_TOMEKLINKS_DATA_TBL (
    "X1" INTEGER,
    "X2" DOUBLE,
    "X3" DOUBLE,
    "TYPE" INTEGER
);
INSERT INTO PAL_TOMEKLINKS_DATA_TBL VALUES (2, 1, 3.5, 1);
```

```
INSERT INTO PAL_TOMEKLINKS_DATA_TBL VALUES (3, 10, 7.6, 1);
INSERT INTO PAL_TOMEKLINKS_DATA_TBL VALUES (3, 10, 5.5, 2);
INSERT INTO PAL_TOMEKLINKS_DATA_TBL VALUES (3, 10, 4.7, 1);
INSERT INTO PAL_TOMEKLINKS_DATA_TBL VALUES (7, 1000, 8.5, 1);
INSERT INTO PAL_TOMEKLINKS_DATA_TBL VALUES (8, 1000, 9.4, 2);
INSERT INTO PAL_TOMEKLINKS_DATA_TBL VALUES (6, 1000, 0.34, 1);
INSERT INTO PAL_TOMEKLINKS_DATA_TBL VALUES (8, 999, 7.4, 2);
INSERT INTO PAL_TOMEKLINKS_DATA_TBL VALUES (7, 999, 3.5, 1);
INSERT INTO PAL_TOMEKLINKS_DATA_TBL VALUES (6, 1000, 7, 1);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 0.1, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('DEPENDENT_VARIABLE', NULL, NULL, 'TYPE');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('METHOD', 1, NULL, NULL);
DROP TABLE PAL_TOMEKLINKS_RESULT_TBL;
CREATE COLUMN TABLE PAL_TOMEKLINKS_RESULT_TBL (
"X1" INTEGER,
"X2" DOUBLE,
"X3" DOUBLE,
"TYPE" INTEGER
);
CALL _SYS_AFL.PAL_TOMEKLINKS(PAL_TOMEKLINKS_DATA_TBL, "#PAL_PARAMETER_TBL",
PAL_TOMEKLINKS_RESULT_TBL) WITH OVERVIEW;
SELECT * FROM PAL_TOMEKLINKS_RESULT_TBL;
```

### Expected Results

RESULT:

| | X1 | X2 | X3 | TYPE |
|---|---|---|---|---|
| 1 | 2 | 1 | 3.5 | 1 |
| 2 | 3 | 10 | 7.6 | 1 |
| 3 | 3 | 10 | 5.5 | 2 |
| 4 | 8 | 1,000 | 9.4 | 2 |
| 5 | 6 | 1,000 | 0.34 | 1 |
| 6 | 8 | 999 | 7.4 | 2 |
| 7 | 7 | 999 | 3.5 | 1 |
| 8 | 6 | 1,000 | 7 | 1 |

Example 2: With INTEGER ID column and under-sampling strategy = 3

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_TOMEKLINKS_DATA_TBL;
CREATE COLUMN TABLE PAL_TOMEKLINKS_DATA_TBL (
    "ID" INTEGER,
    "X1" INTEGER,
    "X2" DOUBLE,
    "X3" DOUBLE,
    "TYPE" INTEGER
);
INSERT INTO PAL_TOMEKLINKS_DATA_TBL VALUES (0, 2, 1, 3.5, 1);
INSERT INTO PAL_TOMEKLINKS_DATA_TBL VALUES (1, 3, 10, 7.6, 1);
INSERT INTO PAL_TOMEKLINKS_DATA_TBL VALUES (2, 3, 10, 5.5, 2);
INSERT INTO PAL_TOMEKLINKS_DATA_TBL VALUES (3, 3, 10, 4.7, 1);
INSERT INTO PAL_TOMEKLINKS_DATA_TBL VALUES (4, 7, 1000, 8.5, 1);
INSERT INTO PAL_TOMEKLINKS_DATA_TBL VALUES (5, 8, 1000, 9.4, 2);
INSERT INTO PAL_TOMEKLINKS_DATA_TBL VALUES (6, 6, 1000, 0.34, 1);
```

```
INSERT INTO PAL_TOMEKLINKS_DATA_TBL VALUES (7, 8, 999, 7.4, 2);
INSERT INTO PAL_TOMEKLINKS_DATA_TBL VALUES (8, 7, 999, 3.5, 1);
INSERT INTO PAL_TOMEKLINKS_DATA_TBL VALUES (9, 6, 1000, 7, 1);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 0.1, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('DEPENDENT_VARIABLE', NULL, NULL, 'TYPE');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SAMPLING_STRATEGY', 3, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('HAS_ID', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('METHOD', 1, NULL, NULL);
DROP TABLE PAL_TOMEKLINKS_RESULT_TBL;
CREATE COLUMN TABLE PAL_TOMEKLINKS_RESULT_TBL (
"ID" INTEGER,
"X1" INTEGER,
"X2" DOUBLE,
"X3" DOUBLE,
"TYPE" INTEGER
);
CALL _SYS_AFL.PAL_TOMEKLINKS(PAL_TOMEKLINKS_DATA_TBL, "#PAL_PARAMETER_TBL",
PAL_TOMEKLINKS_RESULT_TBL) WITH OVERVIEW;
SELECT * FROM PAL_TOMEKLINKS_RESULT_TBL;
```

**Expected Results**

RESULT:

| | ID | X1 | X2 | X3 | TYPE |
|---|---|---|---|---|---|
| 1 | 0 | 2 | 1 | 3.5 | 1 |
| 2 | 1 | 3 | 10 | 7.6 | 1 |
| 3 | 6 | 6 | 1,000 | 0.34 | 1 |
| 4 | 7 | 8 | 999 | 7.4 | 2 |
| 5 | 8 | 7 | 999 | 3.5 | 1 |
| 6 | 9 | 6 | 1,000 | 7 | 1 |

Example 3: With NVARCHAR ID column and under-sampling strategy = 1

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_TOMEKLINKS_DATA_TBL;
CREATE COLUMN TABLE PAL_TOMEKLINKS_DATA_TBL (
    "ID" NVARCHAR(256),
    "X1" INTEGER,
    "X2" DOUBLE,
    "X3" DOUBLE,
    "TYPE" INTEGER
);
INSERT INTO PAL_TOMEKLINKS_DATA_TBL VALUES ('0', 2, 1, 3.5, 1);
INSERT INTO PAL_TOMEKLINKS_DATA_TBL VALUES ('1', 3, 10, 7.6, 1);
INSERT INTO PAL_TOMEKLINKS_DATA_TBL VALUES ('2', 3, 10, 5.5, 2);
INSERT INTO PAL_TOMEKLINKS_DATA_TBL VALUES ('3', 3, 10, 4.7, 1);
INSERT INTO PAL_TOMEKLINKS_DATA_TBL VALUES ('4', 7, 1000, 8.5, 1);
INSERT INTO PAL_TOMEKLINKS_DATA_TBL VALUES ('5', 8, 1000, 9.4, 2);
INSERT INTO PAL_TOMEKLINKS_DATA_TBL VALUES ('6', 6, 1000, 0.34, 1);
INSERT INTO PAL_TOMEKLINKS_DATA_TBL VALUES ('7', 8, 999, 7.4, 2);
INSERT INTO PAL_TOMEKLINKS_DATA_TBL VALUES ('8', 7, 999, 3.5, 1);
INSERT INTO PAL_TOMEKLINKS_DATA_TBL VALUES ('9', 6, 1000, 7, 1);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" NVARCHAR(256),
```

```
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 0.1, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('DEPENDENT_VARIABLE', NULL, NULL, 'TYPE');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SAMPLING_STRATEGY', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('HAS_ID', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('METHOD', 1, NULL, NULL);
DROP TABLE PAL_TOMEKLINKS_RESULT_TBL;
CREATE COLUMN TABLE PAL_TOMEKLINKS_RESULT_TBL (
"ID" NVARCHAR(256),
"X1" INTEGER,
"X2" DOUBLE,
"X3" DOUBLE,
"TYPE" INTEGER
);
CALL _SYS_AFL.PAL_TOMEKLINKS(PAL_TOMEKLINKS_DATA_TBL, "#PAL_PARAMETER_TBL",
PAL_TOMEKLINKS_RESULT_TBL) WITH OVERVIEW;
SELECT * FROM PAL_TOMEKLINKS_RESULT_TBL;
```

**Expected Results**

RESULT:

| | ID | X1 | X2 | X3 | TYPE |
|---|---|---|---|---|---|
| 1 | 0 | 2 | 1 | 3.5 | 1 |
| 2 | 1 | 3 | 10 | 7.6 | 1 |
| 3 | 2 | 3 | 10 | 5.5 | 2 |
| 4 | 5 | 8 | 1,000 | 9.4 | 2 |
| 5 | 6 | 6 | 1,000 | 0.34 | 1 |
| 6 | 7 | 8 | 999 | 7.4 | 2 |
| 7 | 8 | 7 | 999 | 3.5 | 1 |
| 8 | 9 | 6 | 1,000 | 7 | 1 |

## 5.8.21 Variance Test

Variance Test is a method to identify the outliers of n number of numeric data $\{x_i\}$ where $0 < i < n+1$, using the mean $\{\mu\}$ and the standard deviation of $\{\sigma\}$ of n number of numeric data $\{x_i\}$.

Below is the algorithm for Variance Test:

1. Calculate the mean ($\mu$) and the standard deviation ($\sigma$):

$$\mu = \frac{1}{n} \times \sum_{i=1}^{n} x_i$$

$$\sigma = \sqrt{\frac{1}{n-1} \times \sum_{i=1}^{n} (x_i - \mu)^2}$$

2. Set the upper and lower bounds as follows:
   `Upper-bound = μ + multiplier * σ`
   `Lower-bound = μ – multiplier * σ`
   Where the multiplier is a DOUBLE type coefficient provided by the user to test whether all the values of a numeric vector are in the range.
   If a value is outside the range, it means it doesn't pass the Variance Test. The value is marked as an outlier.

## Prerequisites

- No missing or null data in the inputs.
- The data is numeric, not categorical.

## _SYS_AFL.PAL_VARIANCE_TEST

This is a variance test function.

### Overview of all Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<RESULT table>` | OUT |
| 4 | `<STATISTICS table>` | OUT |

### Input Table

| Table | Column | Reference for Output Table | Column Data Type | Description |
|---|---|---|---|---|
| DATA | 1st column | DATA_ID | INTEGER, VARCHAR, or NVARCHAR | ID |
| | 2nd column | | INTEGER, DOUBLE, or DECIMAL(p, s) | Raw data |

### Parameter Table

Mandatory Parameter

The following parameter is mandatory and must be given a value.

| Name | Data Type | Description |
|---|---|---|
| SIGMA_NUM | DOUBLE | Multiplier for sigma |

Optional Parameter

The following parameter is optional. If it is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| THREAD_RATIO | DOUBLE | 1.0 | Specifies the ratio of total number of threads that can be used by this function. The value range is from 0 to 1, where 0 means only using 1 thread, and 1 means using at most all the currently available threads. Values outside the range will be ignored and this function heuristically determines the number of threads to use. |

### Output Tables

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| RESULT | 1st column | DATA_ID (in input table) data type | DATA_ID (in input table) column name | ID |
| | 2nd column | INTEGER | IS_OUT_OF_RANGE | Result output:<br>• 0: in bounds<br>• 1: out of bounds |
| STATISTICS | 1st column | NVARCHAR | STAT_NAME | Statistics name |
| | 2nd column | DOUBLE | STAT_VALUE | Statistics value |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_VT_DATA_TBL;
CREATE COLUMN TABLE PAL_VT_DATA_TBL (
    "ID" INTEGER,
```

```
    "X" DOUBLE
);
INSERT INTO PAL_VT_DATA_TBL VALUES (0,25);
INSERT INTO PAL_VT_DATA_TBL VALUES (1,20);
INSERT INTO PAL_VT_DATA_TBL VALUES (2,23);
INSERT INTO PAL_VT_DATA_TBL VALUES (3,29);
INSERT INTO PAL_VT_DATA_TBL VALUES (4,26);
INSERT INTO PAL_VT_DATA_TBL VALUES (5,23);
INSERT INTO PAL_VT_DATA_TBL VALUES (6,22);
INSERT INTO PAL_VT_DATA_TBL VALUES (7,21);
INSERT INTO PAL_VT_DATA_TBL VALUES (8,22);
INSERT INTO PAL_VT_DATA_TBL VALUES (9,25);
INSERT INTO PAL_VT_DATA_TBL VALUES (10,26);
INSERT INTO PAL_VT_DATA_TBL VALUES (11,28);
INSERT INTO PAL_VT_DATA_TBL VALUES (12,29);
INSERT INTO PAL_VT_DATA_TBL VALUES (13,27);
INSERT INTO PAL_VT_DATA_TBL VALUES (14,26);
INSERT INTO PAL_VT_DATA_TBL VALUES (15,23);
INSERT INTO PAL_VT_DATA_TBL VALUES (16,22);
INSERT INTO PAL_VT_DATA_TBL VALUES (17,23);
INSERT INTO PAL_VT_DATA_TBL VALUES (18,25);
INSERT INTO PAL_VT_DATA_TBL VALUES (19,103);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SIGMA_NUM', null, 3.0, null);
CALL "_SYS_AFL"."PAL_VARIANCE_TEST"(PAL_VT_DATA_TBL, #PAL_PARAMETER_TBL, ?, ?);
```

### Expected Result

RESULT:

| | ID | IS_OUT_OF_RANGE |
|---|---|---|
| 1 | 0 | 0 |
| 2 | 1 | 0 |
| 3 | 2 | 0 |
| 4 | 3 | 0 |
| 5 | 4 | 0 |
| 6 | 5 | 0 |
| 7 | 6 | 0 |
| 8 | 7 | 0 |
| 9 | 8 | 0 |
| 10 | 9 | 0 |
| 11 | 10 | 0 |
| 12 | 11 | 0 |
| 13 | 12 | 0 |
| 14 | 13 | 0 |
| 15 | 14 | 0 |
| 16 | 15 | 0 |
| 17 | 16 | 0 |
| 18 | 17 | 0 |
| 19 | 18 | 0 |
| 20 | 19 | 1 |

STATISTICS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | mean | 28.4 |
| 2 | standard deviation | 17.747942801468863 |

> **Note**
>
> The wrapper procedure generation approach to call PAL functions is still supported. For more information, refer to the older versions (prior to SAP HANA Platform 2.0 SPS 02) of this document.

## 5.9 Statistics Algorithms

This section describes the statistics functions that are provided by the Predictive Analysis Library.

## 5.9.1 Anova

Analysis of variance (ANOVA) is a collection of statistical models used to analyze the differences among group means and their associated procedures (such as "variation" among and between groups).

It is useful for comparing (testing) three or more means (groups or variables) for statistical significance. ANOVA is conceptually similar to multiple two-sample t-tests, but is more conservative (results in less type I error) and is therefore suitable for a wide range of practical problems. The current release of PAL supports one-way ANOVA and one-way repeated measures ANOVA.

### One-Way Analysis of Variance

The purpose of one-way ANOVA is to determine whether there is any statistically significant difference between the means of three or more independent groups. It tests the null hypothesis that all group means are equal:

$H_0: \mu_1=\mu_2=\ldots=\mu_k,$

where $\mu_j$ is the mean of the $j$-th group and $k$ is the number of groups. The fundamental technique of ANOVA is partitioning of the total sum of squares $SS_{Total}$ into components related to different effects. In one-way ANOVA, the total sum of squares is divided into two components: variation between groups and variation within groups,

$SS_{Total}=SS_{between}+SS_{within}=SS_{groups}+SS_{error},$

where

$$SS_{Total}=\sum_i\sum_j\left(y_{ij}-\bar{y}\right)^2,$$

$$SS_{groups}=SS_{between}=\sum_j n_j\left(\bar{y}_{.j}-\bar{y}\right)^2,$$

$$SS_{error}=SS_{within}=\sum_i\sum_j\left(y_{ij}-\bar{y}_{.j}\right)^2,$$

$y_{ij}$ is the $i$-th sample of the $j$-th group, $\bar{y}$ is the overall sample mean, $\bar{y}_{.j}$) and $n_j$ is the sample mean and sample size of the $j$-th group, respectively. The statistical significance is tested for by comparing the $F$ test statistic

$$F=\frac{MS_{groups}}{MS_{error}}=\frac{SS_{groups}/df_{groups}}{SS_{error}/df_{error}}=\frac{SS_{between}/(k-1)}{SS_{within}/(n-k)},$$

where $MS_{groups}$, $MS_{error}$ are mean squares, $df_{groups}=k-1$, $df_{error}=n-k$ are degrees of freedom and $n$ is the total number of samples, to the $F$-distribution with $df_{groups}$, $df_{error}$ degrees of freedom. The null hypothesis is rejected if the $p$-value of $F$ is less than or equal to a certain significance level $\alpha$ (conventional it is set to $\alpha=0.05$).

### One-Way Repeated Measures Analysis of Variance

One-way repeated measures ANOVA is used to compare three or more group means where the subjects are the same in each group, and therefore, different groups represent repeated measures of same subjects. The procedure to perform one-way repeated measures ANOVA is similar as in the original one-way ANOVA except that with repeated measures, variation within groups is further divided into two parts: variation from subject and variation from error,

$SS_{Total}=SS_{between}+SS_{within}=SS_{groups}+SS_{subjects}+SS_{error},$

Where

$$SS_{Total}=\sum_i\sum_j\left(y_{ij}-\bar{y}\right)^2,$$

$$SS_{groups}=SS_{between}=n_S\sum_j\left(\bar{y}_{.j}-\bar{y}\right)^2,$$

$$SS_{error}=SS_{within}-SS_{subjects}=\sum_i\sum_j\left(y_{ij}-\bar{y}_{.j}\right)^2-k\sum_i\left(\bar{y}_{i.}-\bar{y}\right)^2,$$

$n_S$ is the number of subjects in each group (notice that equal number of subjects in each group, i.e. balanced design, is always assumed with repeated measures) and $\bar{y}_{i.}$ is the sample mean of repeated measures of subject `i`. The corresponding F test statistic is calculated as

$$F=\frac{MS_{groups}}{MS_{error}}=\frac{SS_{groups}/df_{groups}}{SS_{error}/df_{error}}=\frac{SS_{between}/(k-1)}{(SS_{within}-SS_{subjects})/\left((n_S-1)(k-1)\right)},$$

where $df_{groups}=k-1, df_{error}=(n_S-1)(k-1)$, and compared to the F-distribution with $df_{groups}$, $df_{error}$ degrees of freedom. With repeated measures, the test is generally more powerful due to the reduction of $SS_{error}$ by subtracting $SS_{subjects}$ from the original error term $SS_{within}$.

## Mauchly's Test of Sphericity

Sphericity, i.e. equality of variances of the differences between all possible pairs of groups, is an important assumption of one-way repeated measures ANOVA. Mauchly's Test of Sphericity tests the null hypothesis that sphericity is satisfied. If Mauchly's test is significant, then the assumption of sphericity is identified as violated and modifications need to be made to the degrees of freedom so that a valid F-ratio can be obtained. In PAL, three corrections are generated: the Greenhouse-Geisser, the Huynh-Feldt, and the lower-bound. For each correction, an estimation sphericity $\varepsilon\leq 1$ ($\varepsilon=1$ indicates that the assumption of sphericity is exactly met) is calculated and used to adjust the degrees of freedom

$$\widetilde{df}_{groups}=\varepsilon*df_{groups},\ \widetilde{df}_{error}=\varepsilon*df_{error},$$

when calculating the F test statistic (in fact F remains the same since $\varepsilon$ in the numerator and denominator cancel each other) and the corresponding p-value. In PAL, one-way repeated measures ANOVA automatically performs Mauchly's Test of Sphericity, $\varepsilon$ of each correction and the corrected p-values are presented in the Mauchly test table and ANOVA table of one-way repeated measures ANOVA output tables, respectively.

## Multiple Comparison Test

When the null hypothesis of equal mean in ANOVA is rejected, one can conclude that at least one group mean is different from another. To further identify which means are different, additional tests are needed. Usually, pairwise comparison tests are performed to measure the difference between each pair of group means. Since these tests are not independent due to the usage of same dataset, the likelihood of incorrectly rejecting a null hypothesis (i.e. making a Type I error) increases when multiple hypotheses are tested. Therefore, special multiple comparison procedures are needed to compensate for these tests.

In PAL, several methods to perform multiple comparison tests are provided. For the `i`-th and `j`-th groups, the `t` test statistic is computed as

$$t=\frac{\bar{y}_{.i}-\bar{y}_{.j}}{s_{ij}},$$

where $s_{ij}$ is the corresponding standard error (which will be explained later). The comparison test rejects the null hypothesis $H_0:\mu_i=\mu_j$ if

- Tukey-Kramer method:

  $$|t|>\frac{1}{\sqrt{2}}q_{\alpha,k,v},$$

  where $q_{\alpha,k,v}$ is the `100*(1-α)` percentage point of the studentized range distribution with parameter `k` and `v`. `k` is the number of groups and `v` is the degrees of freedom corresponding to the standard error $s_{ij}$. This method is optimal for balanced one-way ANOVA and it is proven to be conservative for unequal sample sizes, i.e. it guarantees a confidence coefficient of at least `(1-α)`.
- Bonferroni method:

  $$|t|>t_{\frac{\alpha}{2\binom{k}{2}},\ v}.$$

  This method is a conservative method based on the Bonferroni inequality. It compensates each test by simply dividing the significance level for testing α by the total number of tests $\binom{k}{2}$.
- Dunn-Sidak method:

  $$|t|>t_{\eta/2,\ v},$$

  Where $\eta=1-(1-\alpha)^{1/\binom{k}{2}}$, $t_{\eta/2,v}$ is the $100*(1-\frac{\eta}{2})$ p percentage point of the student's `t` distribution. This method is based on an improved Bonferroni inequality and it is less conservative than the Bonferroni method.
- Scheffe method:

  $$|t|>\sqrt{(k-1)F_{\alpha,\ k-1,\ v}},$$

  where $F_{\alpha,k-1,v}$ is `100*(1-α)` percentage point of the `F` distribution with `k-1` and `v` degrees of freedom. This method can give simultaneous confidence interval for comparisons of all linear combinations of the means.
- Fisher's LSD method:

  $$|t|>t_{\alpha/2,\ v}.$$

  This method provides no protection against the multiple comparison problems and it is usually only appropriate for single test planned before the ANOVA.

For each method, the `p`-value is corrected accordingly and meant to be compared with the original significance level α.

PAL supports two types of standard error:

- For both one-way ANOVA and one-way repeated measures ANOVA: compute the standard error from all data, i.e. from the mean squares of error (in the ANOVA table) and the corresponding sample size

  $$s_{ij}=\sqrt{MS_{error}\left(\frac{1}{n_i}+\frac{1}{n_j}\right)},$$

  the corresponding degrees of freedom $v=df_{error}$, where $MS_{error}$ and $df_{error}$ are the same terms as in the ANOVA table, $n_i$, $n_j$ are the sample sizes of the groups;
- For one-way repeated measures ANOVA only: compute the standard error from only the two groups being compared, just the same as when performing a paired `t` test,

  $$s_{ij}=\frac{s_{ij}^{(d)}}{\sqrt{n_s}},$$

  the corresponding degrees of freedom $v=n_s-1$, where $n_s$ is the number of subjects, $s_{ij}^{(d)}$ is the standard deviation of the mean differences $d_l=y_{li}-y_{lj}$, $l=1,\ldots n_s$.

The first type is standard and the only option for ordinary one-way ANOVA. However, with repeated measures, it may give misleading results if the assumption of sphericity is not true, while the second type does not

assume sphericity but analyzes only a subset of data, therefore has less power than the first type if the assumption of sphericity is true, especially with small number of subjects. By default, one-way repeated measures ANOVA uses the second type of standard error.

## Prerequisites

- PAL_ONEWAY_ANOVA: There must be at least 2 groups and the total number of valid (non-null) samples must be larger than the number of groups.
- PAL_ONEWAY_REPEATED_MEASURES_ANOVA: There must be at least 2 groups and 2 subjects. The input data table does not contain null values, otherwise the algorithm issues errors.

## _SYS_AFL.PAL_ONEWAY_ANOVA

This function performs one-way analysis of variance, along with post hoc multiple comparison tests.

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<STATISTICS table>` | OUT |
| 4 | `<ANOVA table>` | OUT |
| 5 | `<MULTIPLE_COMPARISON table>` | OUT |

### Input Table

| Table | Column | Data Type | Description |
|---|---|---|---|
| DATA | 1st column | INTEGER, VARCHAR, or NVARCHAR | Group |
| | 2nd column | INTEGER, DOUBLE, or DECIMAL(p,s) | Data |

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| MULTCOMP_METHOD | INTEGER | 0 | Specifies the multiple comparison method:<br>• 0: Tukey-Kramer method<br>• 1: Bonferroni method<br>• 2: Dunn-Sidak method<br>• 3: Scheffe method<br>• 4: Fisher's LSD method |
| SIGNIFICANCE_LEVEL | DOUBLE | 0.05 | Specifies the significance level when the function calculates the confidence interval in multiple comparison tests. Note that this parameter is for multiple comparison tests, but not for ANOVA.<br>Value range: Larger than 0 and smaller than 1. |

### Output Tables

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| STATISTICS | 1st column | NVARCHAR (256) | GROUP | Group name |
| | 2nd column | INTEGER | VALID_SAMPLES | Number of valid samples |
| | 3rd column | DOUBLE | MEAN | Mean |
| | 4th column | DOUBLE | SD | Standard deviation |
| ANOVA | 1st column | NVARCHAR(100) | VARIABILITY_SOURCE | Source of variability |
| | 2nd column | DOUBLE | SUM_OF_SQUARES | Sum of squares |
| | 3rd column | DOUBLE | DEGREES_OF_FREEDOM | Degrees of freedom |
| | 4th column | DOUBLE | MEAN_SQUARES | Mean squares |
| | 5th column | DOUBLE | F_RATIO | F-ratio |
| | 6th column | DOUBLE | P_VALUE | p-value |

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| MULTIPLE_COMPARISON | 1st column | NVARCHAR(256) | FIRST_GROUP | Name of the first group |
| | 2nd column | NVARCHAR(256) | SECOND_GROUP | Name of the second group |
| | 3rd column | DOUBLE | MEAN_DIFFERENCE | Mean difference |
| | 4th column | DOUBLE | SE | Standard error |
| | 5th column | DOUBLE | P_VALUE | p-value |
| | 6th column | DOUBLE | CI_LOWER | Lower limit of confidence interval |
| | 7th column | DOUBLE | CI_UPPER | Upper limit of confidence interval |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_ONEWAY_ANOVA_DATA_TBL;
CREATE COLUMN TABLE PAL_ONEWAY_ANOVA_DATA_TBL ("GROUP" NVARCHAR(256), "DATA"
DOUBLE);
INSERT INTO PAL_ONEWAY_ANOVA_DATA_TBL VALUES ('A', 4);
INSERT INTO PAL_ONEWAY_ANOVA_DATA_TBL VALUES ('A', 5);
INSERT INTO PAL_ONEWAY_ANOVA_DATA_TBL VALUES ('A', 4);
INSERT INTO PAL_ONEWAY_ANOVA_DATA_TBL VALUES ('A', 3);
INSERT INTO PAL_ONEWAY_ANOVA_DATA_TBL VALUES ('A', 2);
INSERT INTO PAL_ONEWAY_ANOVA_DATA_TBL VALUES ('A', 4);
INSERT INTO PAL_ONEWAY_ANOVA_DATA_TBL VALUES ('A', 3);
INSERT INTO PAL_ONEWAY_ANOVA_DATA_TBL VALUES ('A', 4);
INSERT INTO PAL_ONEWAY_ANOVA_DATA_TBL VALUES ('B', 6);
INSERT INTO PAL_ONEWAY_ANOVA_DATA_TBL VALUES ('B', 8);
INSERT INTO PAL_ONEWAY_ANOVA_DATA_TBL VALUES ('B', 4);
INSERT INTO PAL_ONEWAY_ANOVA_DATA_TBL VALUES ('B', 5);
INSERT INTO PAL_ONEWAY_ANOVA_DATA_TBL VALUES ('B', 4);
INSERT INTO PAL_ONEWAY_ANOVA_DATA_TBL VALUES ('B', 6);
INSERT INTO PAL_ONEWAY_ANOVA_DATA_TBL VALUES ('B', 5);
INSERT INTO PAL_ONEWAY_ANOVA_DATA_TBL VALUES ('B', 8);
INSERT INTO PAL_ONEWAY_ANOVA_DATA_TBL VALUES ('C', 6);
INSERT INTO PAL_ONEWAY_ANOVA_DATA_TBL VALUES ('C', 7);
INSERT INTO PAL_ONEWAY_ANOVA_DATA_TBL VALUES ('C', 6);
INSERT INTO PAL_ONEWAY_ANOVA_DATA_TBL VALUES ('C', 6);
INSERT INTO PAL_ONEWAY_ANOVA_DATA_TBL VALUES ('C', 7);
INSERT INTO PAL_ONEWAY_ANOVA_DATA_TBL VALUES ('C', 5);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL ("PARAM_NAME"
VARCHAR(256), "INT_VALUE" INTEGER, "DOUBLE_VALUE" DOUBLE, "STRING_VALUE"
VARCHAR(1000));
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SIGNIFICANCE_LEVEL', NULL, 0.05, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MULTCOMP_METHOD', 0, NULL, NULL);
```

```
CALL _SYS_AFL.PAL_ONEWAY_ANOVA (PAL_ONEWAY_ANOVA_DATA_TBL,
#PAL_PARAMETER_TBL, ?, ?, ?);
```

### Expected Result

STATISTICS:

| | GROUP | VALID_SAMPLES | MEAN | SD |
|---|---|---|---|---|
| 1 | A | 8 | 3.625 | 0.9161253813129043 |
| 2 | B | 8 | 5.75 | 1.5811388300841898 |
| 3 | C | 6 | 6.16... | 0.752772652709081 |
| 4 | Total | 22 | 5.09... | 1.6008655667389946 |

ANOVA:

| | VARIABILITY_SOURCE | SUM_OF_SQUARES | DEGREES_OF_FREEDOM | MEAN_SQUARES | F_RATIO | P_VALUE |
|---|---|---|---|---|---|---|
| 1 | Group | 27.609848484848484 | 2 | 13.804924242424242 | 10.008021390374331 | 0.0010748519052363736 |
| 2 | Error | 26.208333333333332 | 19 | 1.3793859649122806 | ? | ? |
| 3 | Total | 53.81818181818181 | 21 | ? | ? | ? |

MULTIPLE_COMPARISON:

| | FIRST_GROUP | SECOND_GROUP | MEAN_DIFFERENCE | SE | P_VALUE | CI_LOWER | CI_UPPER |
|---|---|---|---|---|---|---|---|
| 1 | A | B | -2.125 | 0.5872363163395722 | 0.00495991630546033 2 | -3.6168446200703315 | -0.6331553799296687 |
| 2 | A | C | -2.541666666666667 | 0.6342877158141631 | 0.0020766249150763194 | -4.153043024165101 | -0.9302903091682324 |
| 3 | B | C | -0.4166666666666... | 0.6342877158141631 | 0.7907648749130427 | -2.0280430241651013 | 1.1947096908317678 |

## _SYS_AFL.PAL_ONEWAY_REPEATED_MEASURES_ANOVA

This function performs one-way repeated measures analysis of variance, along with Mauchly's Test of Sphericity and post hoc multiple comparison tests.

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<STATISTICS table>` | OUT |
| 4 | `<MAUCHLY_TEST table>` | OUT |
| 5 | `<ANOVA table>` | OUT |
| 6 | `<MULTIPLE_COMPARISON table>` | OUT |

### Input Table

| Table | Column | Data Type | Description |
|---|---|---|---|
| DATA | 1st column | INTEGER, VARCHAR, or NVARCHAR | Subject ID.<br>The algorithm treats each row of the data table as a different subject. Hence there should be no duplicate subject IDs in this column. |
| | Data columns | INTEGER or DOUBLE | Data of different groups (measures). One column for each group. |

**Parameter Table**

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| MULTCOMP_METHOD | INTEGER | 1 | Specifies the multiple comparison method:<br>• 0: Tukey-Kramer method<br>• 1: Bonferroni method<br>• 2: Dunn-Sidak method<br>• 3: Scheffe method<br>• 4: Fisher's LSD method |
| SIGNIFICANCE_LEVEL | DOUBLE | 0.05 | Specifies the significance level when calculating the confidence interval in multiple comparison tests. Note that this parameter is for multiple comparison tests, but not for ANOVA.<br>Value range: Larger than 0 and smaller than 1. |

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| SE_TYPE | INTEGER | 1 | Specifies the type of standard error used in multiple comparison tests:<br>• 0: Computes the standard error from all data. This type has more power if assumption of sphericity is true, especially with small data sets.<br>• 1: Computes the standard error from only the two groups being compared. This type does not assume sphericity. |

### Output Tables

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| STATISTICS | 1st column | NVARCHAR(256) | GROUP | Group name |
| | 2nd column | INTEGER | VALID_SAMPLES | Number of valid samples |
| | 3rd column | DOUBLE | MEAN | Mean |
| | 4th column | DOUBLE | SD | Standard deviation |
| MAUCHLY_TEST | 1st column | NVARCHAR (100) | STAT_NAME | Names of test result quantities |
| | 2nd column | DOUBLE | STAT_VALUE | Values of test result quantities |
| ANOVA | 1st column | NVARCHAR(100) | VARIABILITY_SOURCE | Source of variability |
| | 2nd column | DOUBLE | SUM_OF_SQUARES | Sum of squares |
| | 3rd column | DOUBLE | DEGREES_OF_FREEDOM | Degrees of freedom |
| | 4th column | DOUBLE | MEAN_SQUARES | Mean squares |
| | 5th column | DOUBLE | F_RATIO | F-ratio |
| | 6th column | DOUBLE | P_VALUE | p-value |

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| | 7th column | DOUBLE | P_VALUE_GG | p-value of Greenhouse-Geisser correction |
| | 8th column | DOUBLE | P_VALUE_HF | p-value of Huynh-Feldt correction |
| | 9th column | DOUBLE | P_VALUE_LB | p-value of lower bound correction |
| MULTIPLE_COMPARISON | 1st column | NVARCHAR(256) | FIRST_GROUP | Name of the first group |
| | 2nd column | NVARCHAR(256) | SECOND_GROUP | Name of the second group |
| | 3rd column | DOUBLE | MEAN_DIFFERENCE | Mean difference |
| | 4th column | DOUBLE | SE | Standard error |
| | 5th column | DOUBLE | P_VALUE | p-value |
| | 6th column | DOUBLE | CI_LOWER | Lower limit of confidence interval |
| | 7th column | DOUBLE | CI_UPPER | Upper limit of confidence interval |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_ONEWAY_REPEATED_MEASURES_ANOVA_DATA_TBL;
CREATE COLUMN TABLE PAL_ONEWAY_REPEATED_MEASURES_ANOVA_DATA_TBL ("ID"
NVARCHAR(100), "MEASURE1" DOUBLE, "MEASURE2" DOUBLE, "MEASURE3" DOUBLE,
"MEASURE4" DOUBLE);
INSERT INTO PAL_ONEWAY_REPEATED_MEASURES_ANOVA_DATA_TBL VALUES ('1', 8, 7, 1, 6);
INSERT INTO PAL_ONEWAY_REPEATED_MEASURES_ANOVA_DATA_TBL VALUES ('2', 9, 5, 2, 5);
INSERT INTO PAL_ONEWAY_REPEATED_MEASURES_ANOVA_DATA_TBL VALUES ('3', 6, 2, 3, 8);
INSERT INTO PAL_ONEWAY_REPEATED_MEASURES_ANOVA_DATA_TBL VALUES ('4', 5, 3, 1, 9);
INSERT INTO PAL_ONEWAY_REPEATED_MEASURES_ANOVA_DATA_TBL VALUES ('5', 8, 4, 5, 8);
INSERT INTO PAL_ONEWAY_REPEATED_MEASURES_ANOVA_DATA_TBL VALUES ('6', 7, 5, 6, 7);
INSERT INTO PAL_ONEWAY_REPEATED_MEASURES_ANOVA_DATA_TBL VALUES ('7', 10, 2, 7,
2);
INSERT INTO PAL_ONEWAY_REPEATED_MEASURES_ANOVA_DATA_TBL VALUES ('8', 12, 6, 8,
1);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL ("PARAM_NAME"
VARCHAR(256), "INT_VALUE" INTEGER, "DOUBLE_VALUE" DOUBLE, "STRING_VALUE"
VARCHAR(1000));
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SIGNIFICANCE_LEVEL', NULL, 0.05, NULL);
```

```
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MULTCOMP_METHOD', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SE_TYPE', 1, NULL, NULL);
CALL _SYS_AFL.PAL_ONEWAY_REPEATED_MEASURES_ANOVA
(PAL_ONEWAY_REPEATED_MEASURES_ANOVA_DATA_TBL, #PAL_PARAMETER_TBL, ?, ?, ?, ?);
```

**Expected Result**

STATISTICS:

| | GROUP | VALID_SAMPLES | MEAN | SD |
|---|---|---|---|---|
| 1 | MEASURE1 | 8 | 8.125 | 2.2320714274285347 |
| 2 | MEASURE2 | 8 | 4.25 | 1.8322507626258087 |
| 3 | MEASURE3 | 8 | 4.125 | 2.748376143938713 |
| 4 | MEASURE4 | 8 | 5.75 | 2.9154759474226504 |

MAUCHLY_TEST:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | Mauchly's W | 0.13624802937253486 |
| 2 | Chi-Square | 11.40598143405639 |
| 3 | df | 5 |
| 4 | pValue | 0.046773348555553... |
| 5 | Greenhous... | 0.5328455527984736 |
| 6 | Huynh-Fel... | 0.66576361409737 |
| 7 | Lower bou... | 0.3333333333333333 |

ANOVA:

| | VARIABILITY_SOURCE | SUM_OF_SQUARES | DEGREES_OF_FREEDOM | MEAN_SQUARES | F_RATIO | P_VALUE | P_VALUE_GG | P_VALUE_HF | P_VALUE_LB |
|---|---|---|---|---|---|---|---|---|---|
| 1 | Group | 83.125 | 3 | 27.708333333333332 | 3.793806030969845 | 0.025570296863039355 | 0.06258412068696351 | 0.04833061355194368 | 0.09247085191468019 |
| 2 | Subject | 17.375 | 7 | 2.482142857142857 | ? | ? | ? | ? | ? |
| 3 | Error | 153.375 | 21 | 7.303571428571429 | ? | ? | ? | ? | ? |

MULTIPLE_COMPARISON:

| | FIRST_GROUP | SECOND_GROUP | MEAN_DIFFERENCE | SE | P_VALUE | CI_LOWER | CI_UPPER |
|---|---|---|---|---|---|---|---|
| 1 | MEASURE1 | MEASURE2 | 3.875 | 0.8114691262501259 | 0.0121396972553236 | 0.9246545126361698 | 6.82534548736383 |
| 2 | MEASURE1 | MEASURE3 | 4 | 0.7319250547113998 | 0.005644859835075344 | 1.3388614398242882 | 6.661138560175711 |
| 3 | MEASURE1 | MEASURE4 | 2.375 | 1.7922202909878495 | 1 | -4.141167869888903 | 8.891167869888903 |
| 4 | MEASURE2 | MEASURE3 | 0.125 | 1.2017472398850884 | 1 | -4.244321556978162 | 4.494321556978162 |
| 5 | MEASURE2 | MEASURE4 | -1.5 | 1.3363062095621219 | 1 | -6.358552060183521 | 3.358552060183521 |
| 6 | MEASURE3 | MEASURE4 | -1.625 | 1.8218661939262795 | 1 | -8.248954664387679 | 4.9989546643876... |

> **Note**
>
> The wrapper procedure generation approach to call PAL functions is still supported. For more information, refer to the older versions (prior to SAP HANA Platform 2.0 SPS 02) of this document.

## 5.9.2 Chi-Square Goodness-of-Fit Test

The chi-square goodness-of fit test tells whether or not an observed distribution differs from an expected chi-squared distribution.

The chi-squared value is defined as:

$$\chi^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i}$$

where $O_i$ is an observed frequency and $E_i$ is an expected frequency.

The chi-squared value $\chi^2$ will be used to calculate a p-value by comparing the value of chi-squared to a chi-squared distribution. The degree of freedom is set to n - p where p is the reduction in degrees of freedom.

### Prerequisites

- The input data has three columns. The first column is ID column with INTEGER, VARCHAR, or NVARCHAR type; the second column is the observed data with INTEGER or DOUBLE type; the third column is the expected frequency.
- The input data does not contain null value. The algorithm will issue errors when encountering null values.

### _SYS_AFL.PAL_CHISQUARED_GOF_TEST

This function does the Pearson's chi-squared test for the goodness-of-fit according to the user's input.

**Overview of All Tables**

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<INPUT table>` | IN |
| 2 | `<Result table>` | OUT |
| 3 | `<STATISTICS table>` | OUT |

**Input Table**

| Table | Column | Reference for Output Table | Column Data Type | Description |
|---|---|---|---|---|
| Data | 1st column | ID | INTEGER, VARCHAR, or NVARCHAR | ID.<br>This must be the first column. |
| | 2nd column | OBSERVED DATA | INTEGER, DOUBLE, or DECIMAL(p, s) | Observed data |
| | 3rd column | EXPECTED FREQUENCY | DOUBLE or DECIMAL(p, s) | Expected frequency P.<br>The sum of expected frequency must be 1. |

### Output Tables

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| RESULT | 1st column | ID (in input table) type | ID (in input table) column name | ID |
| | 2nd column | OBSERVED DATA (in input table) type; widening data type from INTEGER to DOUBLE. | OBSERVED DATA (in input table) column name | Observed data |
| | 3rd column | DOUBLE | EXPECTED | Expected data |
| | 4th column | DOUBLE | RESIDUAL | The residual |
| STATISTICS | 1st column | NVARCHAR(100) | STAT_NAME | Name of statistics |
| | 2nd column | DOUBLE | STAT_VALUE | Value of statistics |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_CHISQTESTFIT_DATA_TBL;
CREATE COLUMN TABLE PAL_CHISQTESTFIT_DATA_TBL ("ID" INTEGER, "OBSERVED" DOUBLE,
"P"  DOUBLE);
INSERT INTO PAL_CHISQTESTFIT_DATA_TBL VALUES (0,519,0.3);
INSERT INTO PAL_CHISQTESTFIT_DATA_TBL VALUES (1,364,0.2);
INSERT INTO PAL_CHISQTESTFIT_DATA_TBL VALUES (2,363,0.2);
INSERT INTO PAL_CHISQTESTFIT_DATA_TBL VALUES (3,200,0.1);
INSERT INTO PAL_CHISQTESTFIT_DATA_TBL VALUES (4,212,0.1);
INSERT INTO PAL_CHISQTESTFIT_DATA_TBL VALUES (5,193,0.1);
CALL _SYS_AFL.PAL_CHISQUARED_GOF_TEST(PAL_CHISQTESTFIT_DATA_TBL, ?, ?);
```

**Expected Result**

RESULT:

| | ID | OBSERVED | EXPECTED | RESIDUAL |
|---|---|---|---|---|
| 1 | 0 | 519 | 555.3 | -36.299999999999955 |
| 2 | 1 | 364 | 370.20000000000005 | -6.2000000000000455 |
| 3 | 2 | 363 | 370.20000000000005 | -7.2000000000000455 |
| 4 | 3 | 200 | 185.10000000000002 | 14.899999999999977 |
| 5 | 4 | 212 | 185.10000000000002 | 26.899999999999977 |
| 6 | 5 | 193 | 185.10000000000002 | 7.899999999999977 |

STATISTICS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | Chi-squared Value | 8.062668827660708 |
| 2 | degree of freedom | 5 |
| 3 | p-value | 0.1528154775508... |

> **Note**
>
> The wrapper procedure generation approach to call PAL functions is still supported. For more information, refer to the older versions (prior to SAP HANA Platform 2.0 SPS 02) of this document.

## 5.9.3 Chi-Squared Test of Independence

The chi-squared test of independence tells whether observations of two variables are independent from each other.

The chi-squared value is defined as:

$$\chi^2 = \sum_{i=1}^{r} \sum_{j=1}^{c} \frac{(O_{i,j} - E_{i,j})^2}{E_{i,j}}$$

where $O_{i,j}$ is an observed frequency and $E_{i,j}$ is an expected frequency. Then $\chi^2$ will be used to calculate a p-value by comparing the value of chi-squared to a chi-squared distribution. The degree of freedom is set to `(r-1)*(c-1)`.

### Prerequisites

- The input data contains an ID column in the first column with the type of INTEGER, VARCHAR, or NVARCHAR and the other columns are of INTEGER or DOUBLE data type.

- The input data does not contain null value. The algorithm will issue errors when encountering null values.

## _SYS_AFL.PAL_CHISQUARED_IND_TEST

This function does the Pearson's chi-squared test for independent according to the user's input.

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<RESULT table>` | OUT |
| 4 | `<STATISTICS table>` | OUT |

### Input Table

| Table | Column | Reference for Output Table | Data Type | Description |
|---|---|---|---|---|
| DATA | 1st column | ID | INTEGER, VARCHAR, or NVARCHAR | ID.<br>This must be the first column. |
| | Other columns | OBSERVED DATA | INTEGER, DOUBLE, or DECIMAL(p, s) | Observed data |

### Parameter Table

Mandatory Parameters

None.

Optional Parameter

The following parameter is optional. If it is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| CORRECTION_TYPE | INTEGER | 0 | Controls whether to perform the Yates's correction for continuity.<br>• 0: Does not perform the Yates's correction<br>• 1: Performs the Yates's correction |

### Output Tables

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| RESULT | 1st column | ID (in input table) data type | ID (in input table) column name | ID |
| | Other columns | OBSERVED DATA (in input table) data type; widening data type from INTEGER to DOUBLE. | EXPECTED+OBSERVED DATA (in input table) column name | Expected data |
| STATISTICS | 1st column | NVARCHAR(100) | STAT_NAME | Name of statistics |
| | 2nd column | DOUBLE | STAT_VALUE | Value of statistics |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_CHISQTESTIND_DATA_TBL;
CREATE COLUMN TABLE PAL_CHISQTESTIND_DATA_TBL ("ID" VARCHAR(100), "X1" INTEGER,
"X2" DOUBLE, "X3" INTEGER, "X4" DOUBLE);
INSERT INTO PAL_CHISQTESTIND_DATA_TBL VALUES ('male',25,23,11,14);
INSERT INTO PAL_CHISQTESTIND_DATA_TBL VALUES ('female',41,20,18,6);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE
    #PAL_PARAMETER_TBL
    ("PARAM_NAME" VARCHAR(256), "INT_VALUE" INTEGER, "DOUBLE_VALUE" DOUBLE,
"STRING_VALUE" VARCHAR(1000));
INSERT INTO #PAL_PARAMETER_TBL VALUES ('CORRECTION_TYPE',0,NULL,NULL); //default
value is 0, it can be {0,1}
CALL _SYS_AFL.PAL_CHISQUARED_IND_TEST(PAL_CHISQTESTIND_DATA_TBL,
#PAL_PARAMETER_TBL, ?, ?) ;
```

### Expected Result

RESULT:

| | ID | EXPECTED_X1 | EXPECTED_X2 | EXPECTED_X3 | EXPECTED_X4 |
|---|---|---|---|---|---|
| 1 | male | 30.49367088607595 | 19.867088607594937 | 13.39873417721519 | 9.240506329113924 |
| 2 | female | 35.50632911392405 | 23.132911392405063 | 15.60126582278481 | 10.759493670886076 |

STATISTICS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | Chi-squared Value | 8.113152118695105 |
| 2 | degree of freedom | 3 |
| 3 | p-value | 0.0437301693095... |

> **Note**
>
> The wrapper procedure generation approach to call PAL functions is still supported. For more information, refer to the older versions (prior to SAP HANA Platform 2.0 SPS 02) of this document.

## 5.9.4 Condition Index

Condition index is used to detect collinearity problem between independent variables which are later used as predictors in a multiple linear regression model.

This method firstly employs the principle component analysis (PCA) to find out the eigenvalues and the corresponding eigenvectors of the matrix formed by independent variables, then calculates the condition index and variance decomposition proportion. For example, if you feed in a data matrix $(X_{ij})_{n\times p}$, this function gets singular values $\sigma_i$ $(i=1,\ldots,p)$ and the V matrix $(V_{kj})_{p\times p}$ from the singular value decomposition, then proceeds to calculate condition index

$CI_i=\sigma_{max}/\sigma_i$,

and the condition number

$CN=\sigma_{max}/\sigma_{min}$,

which is the largest value of condition indices. Note that a diagonal matrix $D=diag(\sigma_1,\ldots,\sigma_p)$, you can calculate variance decomposition proportions $\pi_{jk}=\varphi_{jk}/\varphi_k$, where $\phi_{kj}=V_{kj}^2/\sigma_j^2$ and $\phi_k=\sum_{j=1}^{p} V_{kj}^2/\sigma_j^2$. This quantity illustrates how much variance of the estimated coefficient for a variable can be explained by the k-th principle component.

Generally speaking, a dataset with condition number larger than 30 indicates the existence of a possible collinearity. Variables which are involved in collinearity have variance decomposition proportions greater than 0.5.

### Prerequisites

- The input data columns are of INTEGER or DOUBLE data type.

- The input data has no null value.

## _SYS_AFL.PAL_CONDITION_INDEX

This function reads input data and calculates the condition index and variance decomposition proportion.

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<INPUT table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<RESULT table>` | OUT |
| 4 | `<HELPER table>` | OUT |

### Input Table

| Table | Column | Reference for Output Table | Data Type | Description |
|---|---|---|---|---|
| INPUT | 1st column | ID | INTEGER, VARCHAR, or NVARCHAR | |
| | Feature columns | FEATURES | DOUBLE or DECIMAL(p,s) | Independent variables that will be used in multiple linear regression. Continuous variable only. |

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| THREAD_RATIO | DOUBLE | 0 | Specifies the ratio of total number of threads that can be used by this function. The value range is from 0 to 1, where 0 means only using 1 thread, and 1 means using at most all the currently available threads. Values outside the range will be ignored and this function heuristically determines the number of threads to use. |
| INCLUDE_INTERCEPT | INTEGER | 1 | Specifies whether the algorithm considers intercept during the calculation.<br>• 0: No<br>• 1: Yes |
| SCALING | INTEGER | 1 | Specifies whether the input data are scaled to have unit variance before the analysis.<br>• 0: No<br>• 1: Yes |
| SELECTED_FEATURES | VARCHAR | No default value | A string to specify the features that will be processed.<br>The pattern is "X1,...,Xn", where Xi is the corresponding column name in the data table.<br>If this parameter is not specified, all features will be processed, and the data in all columns except for the first and second columns will be processed as independent variables. |

## Output Tables

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| RESULT | 1st column | NVARCHAR(100) | COMPONENT_ID | Principal component ID. |

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| | 2nd column | DOUBLE | EIGENVALUE | Eigenvalue. |
| | 3rd column | DOUBLE | CONDITION_INDEX | Condition index. |
| | Other columns | FEATURES (in input table) data type | FEATURES (in input table) column name | Variance decomposition proportion for each variable. |
| | Last column | DOUBLE | INTERCEPT | Variance decomposition proportion for the intercept term. |
| HELPER | 1st column | NVARCHAR(100) | STAT_NAME | Name for the values, including condition number, and the name of variables which are involved in collinearity problem.<br>This table is empty if collinearity problem has not been detected. |
| | 2nd column | DOUBLE | STAT_VALUE | Values of the corresponding name, either condition number or variance decomposition proportions for variables. |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA "DM_PAL";
DROP TABLE PAL_CI_DATA_TBL;
CREATE COLUMN TABLE PAL_CI_DATA_TBL
    ("ID" INTEGER, "X1" DOUBLE, "X2" DOUBLE, "X3" DOUBLE, "X4" DOUBLE);
INSERT INTO PAL_CI_DATA_TBL VALUES (1, 12, 52, 20, 44);
INSERT INTO PAL_CI_DATA_TBL VALUES (2, 12, 57, 25, 45);
INSERT INTO PAL_CI_DATA_TBL VALUES (3, 12, 54, 21, 45);
INSERT INTO PAL_CI_DATA_TBL VALUES (4, 13, 52, 21, 46);
INSERT INTO PAL_CI_DATA_TBL VALUES (5, 14, 54, 24, 46);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL
("PARAM_NAME" VARCHAR(50256), "INT_VALUE" INTEGER, "DOUBLE_VALUE" DOUBLE,
"STRING_VALUE" VARCHAR(1000));
INSERT INTO #PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 0.1, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('INCLUDE_INTERCEPT', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SCALING', 1, NULL, NULL);
```

```
CALL _SYS_AFL.PAL_CONDITION_INDEX(PAL_CI_DATA_TBL,"#PAL_PARAMETER_TBL", ?, ?);
```

**Expected Result**

RESULT:

| | COMPONENT_ID | EIGENVALUE | CONDITION_INDEX | X1 | X2 | X3 | X4 | INTERCEPT |
|---|---|---|---|---|---|---|---|---|
| 1 | Comp_1 | 19.966688092070346 | 1 | 0.000011857606279048819 | 0.000001556872173788015 5 | 0.000009911148151911021 | 0.000003175777767443220 7 | 0.000002173804609841471 |
| 2 | Comp_2 | 0.020735848860972662 | 31.0307382148988 | 0.008776373701414155 | 0.0002098206196380496 | 0.031062754113955482 | 0.00125108721916143 | 0.0009070816379272149 |
| 3 | Comp_3 | 0.01226013461397973 | 40.35574763495319 | 0.053471983177132296 | 0.002570866013111457 | 0.005314573321664645 | 0.0006389341283516 75 | 0.0027104870706757133 |
| 4 | Comp_4 | 0.0002295285079620... | 294.9406955200734 | 0.205665614112024 41 | 0.015224312605729503 | 0.006578587820129992 | 0.9311208208813717 | 0.24686206444251055 |
| 5 | Comp_5 | 0.0000863959467271... | 480.7356538429136 | 0.7320741714031501 | 0.9819934438893472 | 0.9570341735960979 | 0.06698598199334779 | 0.7495181930442767 |

HELPER:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | CONDITION_NUMBER | 480.7356538429136 |
| 2 | X1 | 0.7320741714031501 |
| 3 | X2 | 0.9819934438893472 |
| 4 | X3 | 0.9570341735960979 |
| 5 | INTERCEPT | 0.7495181930442767 |

> **Note**
>
> The wrapper procedure generation approach to call PAL functions is still supported. For more information, refer to the older versions (prior to SAP HANA Platform 2.0 SPS 02) of this document.

## 5.9.5 Correlation Function

A correlation function gives the statistical correlation between random variables.

If one considers the correlation function between random variables and itself at different time points, then this is often referred to as an auto-correlation function (ACF). Correlation functions of different random variables are sometimes called cross-correlation functions (CCF). Correlation functions used in astronomy, financial analysis, econometrics, and statistical mechanics differ only in the particular stochastic processes they are applied to.

PAL considers the sample correlation function only. Given a variable with observations $x_1, x_2, \ldots, x_n$, the sample auto-covariance function (ACVF) at lag h is

$$\hat{\gamma}(h) := \frac{1}{n}\sum_{t=1}^{n-|h|}(x_{t+|h|}-\bar{x})(x_t-\bar{x}), \quad -n<h<n$$

And its corresponding auto-correlation function is:

$$\hat{\rho}(h) = \frac{\hat{\gamma}(h)}{\hat{\gamma}(0)}, \quad -n<h<n$$

Evidently, $\hat{\gamma}(h)=\hat{\gamma}(-h)$, and $\hat{\rho}(h)=\hat{\rho}(-h)$.

In contrast with auto-correlation function, partial auto-correlation function (PACF) measures the relationship between $x_t$ and $x_{t+k}$ after removing the effects of other time lags $1, 2, \ldots, k-1$, which is very useful in time series forecast. PACF can be solved iteratively with Durbin-Levinson algorithm.

The cross-covariance function (CCVF) and cross-correlation function between series x and y, likewise, has definitions

$\gamma_{XY}(h)=E[(x_t-\mu_X)(y_{t+h}-\mu_Y)]$

$\rho_{XY}(h)=E[(x_t-\mu_X)(y_{t+h}-\mu_Y)]/(\sigma_X\sigma_Y)=\gamma_{XY}(h)/(\sigma_X\sigma_Y)$

where $\mu_X$ and $\sigma_X$ are the mean and the standard deviation of the process $x_t$, which are constant over time due to stationary; and similarly for $y_t$, respectively.

Unlike auto-covariance and auto-correlation, clearly, cross-covariance and cross-correlation are asymmetric, i.e. $\gamma_{XY}(h)\neq\gamma_{XY}(-h), \rho_{XY}(h)\neq\rho_{XY}(-h)$. Thus, we need to compute the functions from lag -h to lag h.

Similar to convolution, correlation functions can be solved using fast Fourier transform (FFT), which is also implemented in PAL.

## Prerequisites

- An ID column must be provided, and the column does not contain null value.
- If one sequence is provided, auto-correlation function is calculated. If two sequences are provided, cross-correlation function is calculated.
- Null value in sequence indicates the end of data, and the longer sequence for cross-correlation is truncated to the same length of the shorter one.

## _SYS_AFL.PAL_CORRELATION_FUNCTION

This function calculates the auto-correlation function of a series, or the cross-correlation function between two series.

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<RESULT table>` | OUT |

### Input Table

| Table | Column | Data Type | Description |
|---|---|---|---|
| DATA | 1st column | INTEGER | The ID column that indicates the order of the sequence.<br>This must be the first column. |
| | 2nd column | INTEGER, DOUBLE, or DECIMAL(p, s) | First series |
| | 3rd column (optional) | INTEGER, DOUBLE, or DECIMAL(p, s) | Second series |

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| THREAD_RATIO | DOUBLE | -1 | The ratio of available threads.<br>• 0: single thread<br>• 0~1: percentage<br>• Others: heuristically determined | Valid only when the brute-force method is used. |
| USE_FFT | INTEGER | -1 | Indicates the method to be used to calculate the correlation function.<br>• -1: Automatically determined<br>• 0: Brute-force<br>• 1: FFT | |
| MAX_LAG | INTEGER | sqrt(n) | Maximum lag for the correlation function | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| CALCULATE_PACF | INTEGER | 1 | Controls whether to calculate PACF or not.<br>• 0: Does not calculate PACF<br>• 1: Calculates PACF | Valid only when only one series is provided (the Data table consists of 2 columns), i.e. auto-correlation is calculated. |

### Output Table

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| RESULT | 1st column | INTEGER | LAG | ID column.<br>Non-negative for ACF; symmetric for CCF. |
| | 2nd column | DOUBLE | CV | ACV/CCV |
| | 3rd column | DOUBLE | CF | ACF/CCF |
| | 4th column | DOUBLE | PACF | PACF. Null if cross-correlation is calculated. |

### Examples

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

### Example 1: Auto-correlation

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_CORR_DATA_TBL;
CREATE COLUMN TABLE PAL_CORR_DATA_TBL (
    "ID" INTEGER,
    "X" DOUBLE
);
INSERT INTO PAL_CORR_DATA_TBL VALUES(1, 88);
INSERT INTO PAL_CORR_DATA_TBL VALUES(2, 84);
INSERT INTO PAL_CORR_DATA_TBL VALUES(3, 85);
INSERT INTO PAL_CORR_DATA_TBL VALUES(4, 85);
INSERT INTO PAL_CORR_DATA_TBL VALUES(5, 84);
INSERT INTO PAL_CORR_DATA_TBL VALUES(6, 85);
INSERT INTO PAL_CORR_DATA_TBL VALUES(7, 83);
INSERT INTO PAL_CORR_DATA_TBL VALUES(8, 85);
INSERT INTO PAL_CORR_DATA_TBL VALUES(9, 88);
INSERT INTO PAL_CORR_DATA_TBL VALUES(10, 89);
INSERT INTO PAL_CORR_DATA_TBL VALUES(11, 91);
INSERT INTO PAL_CORR_DATA_TBL VALUES(12, 99);
INSERT INTO PAL_CORR_DATA_TBL VALUES(13, 104);
INSERT INTO PAL_CORR_DATA_TBL VALUES(14, 112);
```

```
INSERT INTO PAL_CORR_DATA_TBL VALUES(15, 126);
INSERT INTO PAL_CORR_DATA_TBL VALUES(16, 138);
INSERT INTO PAL_CORR_DATA_TBL VALUES(17, 146);
INSERT INTO PAL_CORR_DATA_TBL VALUES(18, 151);
INSERT INTO PAL_CORR_DATA_TBL VALUES(19, 150);
INSERT INTO PAL_CORR_DATA_TBL VALUES(20, 148);
INSERT INTO PAL_CORR_DATA_TBL VALUES(21, 147);
INSERT INTO PAL_CORR_DATA_TBL VALUES(22, 149);
INSERT INTO PAL_CORR_DATA_TBL VALUES(23, 143);
INSERT INTO PAL_CORR_DATA_TBL VALUES(24, 132);
INSERT INTO PAL_CORR_DATA_TBL VALUES(25, 131);
INSERT INTO PAL_CORR_DATA_TBL VALUES(26, 139);
INSERT INTO PAL_CORR_DATA_TBL VALUES(27, 147);
INSERT INTO PAL_CORR_DATA_TBL VALUES(28, 150);
INSERT INTO PAL_CORR_DATA_TBL VALUES(29, 148);
INSERT INTO PAL_CORR_DATA_TBL VALUES(30, 145);
INSERT INTO PAL_CORR_DATA_TBL VALUES(31, 140);
INSERT INTO PAL_CORR_DATA_TBL VALUES(32, 134);
INSERT INTO PAL_CORR_DATA_TBL VALUES(33, 131);
INSERT INTO PAL_CORR_DATA_TBL VALUES(34, 131);
INSERT INTO PAL_CORR_DATA_TBL VALUES(35, 129);
INSERT INTO PAL_CORR_DATA_TBL VALUES(36, 126);
INSERT INTO PAL_CORR_DATA_TBL VALUES(37, 126);
INSERT INTO PAL_CORR_DATA_TBL VALUES(38, 132);
INSERT INTO PAL_CORR_DATA_TBL VALUES(39, 137);
INSERT INTO PAL_CORR_DATA_TBL VALUES(40, 140);
INSERT INTO PAL_CORR_DATA_TBL VALUES(41, 142);
INSERT INTO PAL_CORR_DATA_TBL VALUES(42, 150);
INSERT INTO PAL_CORR_DATA_TBL VALUES(43, 159);
INSERT INTO PAL_CORR_DATA_TBL VALUES(44, 167);
INSERT INTO PAL_CORR_DATA_TBL VALUES(45, 170);
INSERT INTO PAL_CORR_DATA_TBL VALUES(46, 171);
INSERT INTO PAL_CORR_DATA_TBL VALUES(47, 172);
INSERT INTO PAL_CORR_DATA_TBL VALUES(48, 172);
INSERT INTO PAL_CORR_DATA_TBL VALUES(49, 174);
INSERT INTO PAL_CORR_DATA_TBL VALUES(50, 175);
INSERT INTO PAL_CORR_DATA_TBL VALUES(51, 172);
INSERT INTO PAL_CORR_DATA_TBL VALUES(52, 172);
INSERT INTO PAL_CORR_DATA_TBL VALUES(53, 174);
INSERT INTO PAL_CORR_DATA_TBL VALUES(54, 174);
INSERT INTO PAL_CORR_DATA_TBL VALUES(55, 169);
INSERT INTO PAL_CORR_DATA_TBL VALUES(56, 165);
INSERT INTO PAL_CORR_DATA_TBL VALUES(57, 156);
INSERT INTO PAL_CORR_DATA_TBL VALUES(58, 142);
INSERT INTO PAL_CORR_DATA_TBL VALUES(59, 131);
INSERT INTO PAL_CORR_DATA_TBL VALUES(60, 121);
INSERT INTO PAL_CORR_DATA_TBL VALUES(61, 112);
INSERT INTO PAL_CORR_DATA_TBL VALUES(62, 104);
INSERT INTO PAL_CORR_DATA_TBL VALUES(63, 102);
INSERT INTO PAL_CORR_DATA_TBL VALUES(64, 99);
INSERT INTO PAL_CORR_DATA_TBL VALUES(65, 99);
INSERT INTO PAL_CORR_DATA_TBL VALUES(66, 95);
INSERT INTO PAL_CORR_DATA_TBL VALUES(67, 88);
INSERT INTO PAL_CORR_DATA_TBL VALUES(68, 84);
INSERT INTO PAL_CORR_DATA_TBL VALUES(69, 84);
INSERT INTO PAL_CORR_DATA_TBL VALUES(70, 87);
INSERT INTO PAL_CORR_DATA_TBL VALUES(71, 89);
INSERT INTO PAL_CORR_DATA_TBL VALUES(72, 88);
INSERT INTO PAL_CORR_DATA_TBL VALUES(73, 85);
INSERT INTO PAL_CORR_DATA_TBL VALUES(74, 86);
INSERT INTO PAL_CORR_DATA_TBL VALUES(75, 89);
INSERT INTO PAL_CORR_DATA_TBL VALUES(76, 91);
INSERT INTO PAL_CORR_DATA_TBL VALUES(77, 91);
INSERT INTO PAL_CORR_DATA_TBL VALUES(78, 94);
INSERT INTO PAL_CORR_DATA_TBL VALUES(79, 101);
INSERT INTO PAL_CORR_DATA_TBL VALUES(80, 110);
INSERT INTO PAL_CORR_DATA_TBL VALUES(81, 121);
INSERT INTO PAL_CORR_DATA_TBL VALUES(82, 135);
```

```
INSERT INTO PAL_CORR_DATA_TBL VALUES(83, 145);
INSERT INTO PAL_CORR_DATA_TBL VALUES(84, 149);
INSERT INTO PAL_CORR_DATA_TBL VALUES(85, 156);
INSERT INTO PAL_CORR_DATA_TBL VALUES(86, 165);
INSERT INTO PAL_CORR_DATA_TBL VALUES(87, 171);
INSERT INTO PAL_CORR_DATA_TBL VALUES(88, 175);
INSERT INTO PAL_CORR_DATA_TBL VALUES(89, 177);
INSERT INTO PAL_CORR_DATA_TBL VALUES(90, 182);
INSERT INTO PAL_CORR_DATA_TBL VALUES(91, 193);
INSERT INTO PAL_CORR_DATA_TBL VALUES(92, 204);
INSERT INTO PAL_CORR_DATA_TBL VALUES(93, 208);
INSERT INTO PAL_CORR_DATA_TBL VALUES(94, 210);
INSERT INTO PAL_CORR_DATA_TBL VALUES(95, 215);
INSERT INTO PAL_CORR_DATA_TBL VALUES(96, 222);
INSERT INTO PAL_CORR_DATA_TBL VALUES(97, 228);
INSERT INTO PAL_CORR_DATA_TBL VALUES(98, 226);
INSERT INTO PAL_CORR_DATA_TBL VALUES(99, 222);
INSERT INTO PAL_CORR_DATA_TBL VALUES(100, 220);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR(100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR(100)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 0.4, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('USE_FFT', -1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('CALCULATE_PACF', 1, NULL, NULL);
CALL _SYS_AFL.PAL_CORRELATION_FUNCTION (PAL_CORR_DATA_TBL,
"#PAL_PARAMETER_TBL", ?);
```

### Expected Result

RESULT:

| | LAG | CV | CF | PACF |
|---|---|---|---|---|
| 1 | 0 | 1,583.9536 | 1 | 1 |
| 2 | 1 | 1,520.880736 | 0.9601801062859416 | 0.9601801062859417 |
| 3 | 2 | 1,427.356272 | 0.9011351544641205 | -0.2666184750683... |
| 4 | 3 | 1,312.6958... | 0.8287463774191363 | -0.1544168868600... |
| 5 | 4 | 1,181.6069... | 0.7459858319082071 | -0.1201758111837... |
| 6 | 5 | 1,041.0424... | 0.6572430404527 | -0.0715456170771... |
| 7 | 6 | 894.493216... | 0.5647218554886962 | -0.0650646083170... |
| 8 | 7 | 742.178351... | 0.4685606649083659 | -0.0836864209235... |
| 9 | 8 | 587.453488... | 0.3708779651121094 | -0.0652131828823... |
| 10 | 9 | 434.287823... | 0.2741796375853433 | -0.0455008583489... |
| 11 | 10 | 286.464160... | 0.1808538835986105 | -0.0295857367228... |

### Example 2: Cross-correlation

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_CORR_DATA_TBL;
CREATE COLUMN TABLE PAL_CORR_DATA_TBL (
    "ID" INTEGER,
    "X" DOUBLE,
    "Y" DOUBLE
);
INSERT INTO PAL_CORR_DATA_TBL VALUES(1, 1, 3);
INSERT INTO PAL_CORR_DATA_TBL VALUES(2, 2, 8);
INSERT INTO PAL_CORR_DATA_TBL VALUES(3, 3, 0);
```

```
INSERT INTO PAL_CORR_DATA_TBL VALUES(5, 5, 8);
INSERT INTO PAL_CORR_DATA_TBL VALUES(6, 6, 4);
INSERT INTO PAL_CORR_DATA_TBL VALUES(7, NULL, 3);
INSERT INTO PAL_CORR_DATA_TBL VALUES(4, 4, 16);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR(100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (100)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 0.4, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('USE_FFT', -1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('CALCULATE_PACF', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MAX_LAG', 5, NULL, NULL);
CALL _SYS_AFL.PAL_CORRELATION_FUNCTION (PAL_CORR_DATA_TBL,
"#PAL_PARAMETER_TBL", ?);
```

**Expected Result**

RESULT:

| | LAG | CV | CF | PACF |
|---|---|---|---|---|
| 1 | -5 | -1.4583333333333333 | -0.1677350536754654 | ? |
| 2 | -4 | -0.25 | -0.028754580630079782 | ? |
| 3 | -3 | -2.625 | -0.3019230966158377 | ? |
| 4 | -2 | 2.75 | 0.3163003869308776 | ? |
| 5 | -1 | 3.2083333333333335 | 0.3690171180860239 | ? |
| 6 | 0 | 1.75 | 0.2012820644105585 | ? |
| 7 | 1 | -0.2916666666666667 | -0.03354701073509308 | ? |
| 8 | 2 | 0 | 0 | ? |
| 9 | 3 | -4.125 | -0.4744505803963164 | ? |
| 10 | 4 | 0 | 0 | ? |
| 11 | 5 | 1.0416666666666667 | 0.11981075262533243 | ? |

## _SYS_AFL.PAL_CORRELATION_FUNCTION_WITH_CONFIDENCE

This function extends the outputs of auto-correlation by adding the confidence bound for ACF and PACF. Particularly, the confidence bound of ACF can be generated by the Bartlett's formula.

**Input Table**

| Table | Column | Data Type | Description |
|---|---|---|---|
| DATA | 1st column | INTEGER | The ID column that indicates the order of the sequence.<br>This must be the first column. |
| | 2nd column | INTEGER, DOUBLE, or DECIMAL(p,s) | Auto-correlation time series data |

### Parameter Table

This function uses all the parameters listed above as well as the exclusive optional parameters below.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| ALPHA | DOUBLE | 0.05 | Confidence bound for the given level is returned. For instance if alpha = 0.05, 95% confidence bound is returned. | |
| BARTLETT | INTEGER | 1 | • 0: using standard error to calculate the confidence bound<br>• 1: using Bartlett's formula to calculate the confidence bound | |

### Examples

Assume that:

- DM_PAL is a schema belonging to USER1;
- USER1 has been assigned the AFL__SYS_AFL_AFLPAL_EXECUTE or AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION role.

**Example: Auto-correlation with confidence bound using Bartlett's formula**

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_CORR_DATA_TBL;
CREATE COLUMN TABLE PAL_CORR_DATA_TBL (
    "ID" INTEGER,
    "X" DOUBLE
);
INSERT INTO PAL_CORR_DATA_TBL VALUES(1, 88);
INSERT INTO PAL_CORR_DATA_TBL VALUES(2, 84);
INSERT INTO PAL_CORR_DATA_TBL VALUES(3, 85);
INSERT INTO PAL_CORR_DATA_TBL VALUES(4, 85);
INSERT INTO PAL_CORR_DATA_TBL VALUES(5, 84);
INSERT INTO PAL_CORR_DATA_TBL VALUES(6, 85);
INSERT INTO PAL_CORR_DATA_TBL VALUES(7, 83);
INSERT INTO PAL_CORR_DATA_TBL VALUES(8, 85);
INSERT INTO PAL_CORR_DATA_TBL VALUES(9, 88);
INSERT INTO PAL_CORR_DATA_TBL VALUES(10, 89);
INSERT INTO PAL_CORR_DATA_TBL VALUES(11, 91);
INSERT INTO PAL_CORR_DATA_TBL VALUES(12, 99);
INSERT INTO PAL_CORR_DATA_TBL VALUES(13, 104);
INSERT INTO PAL_CORR_DATA_TBL VALUES(14, 112);
INSERT INTO PAL_CORR_DATA_TBL VALUES(15, 126);
INSERT INTO PAL_CORR_DATA_TBL VALUES(16, 138);
INSERT INTO PAL_CORR_DATA_TBL VALUES(17, 146);
INSERT INTO PAL_CORR_DATA_TBL VALUES(18, 151);
INSERT INTO PAL_CORR_DATA_TBL VALUES(19, 150);
INSERT INTO PAL_CORR_DATA_TBL VALUES(20, 148);
INSERT INTO PAL_CORR_DATA_TBL VALUES(21, 147);
INSERT INTO PAL_CORR_DATA_TBL VALUES(22, 149);
INSERT INTO PAL_CORR_DATA_TBL VALUES(23, 143);
```

```
INSERT INTO PAL_CORR_DATA_TBL VALUES(24, 132);
INSERT INTO PAL_CORR_DATA_TBL VALUES(25, 131);
INSERT INTO PAL_CORR_DATA_TBL VALUES(26, 139);
INSERT INTO PAL_CORR_DATA_TBL VALUES(27, 147);
INSERT INTO PAL_CORR_DATA_TBL VALUES(28, 150);
INSERT INTO PAL_CORR_DATA_TBL VALUES(29, 148);
INSERT INTO PAL_CORR_DATA_TBL VALUES(30, 145);
INSERT INTO PAL_CORR_DATA_TBL VALUES(31, 140);
INSERT INTO PAL_CORR_DATA_TBL VALUES(32, 134);
INSERT INTO PAL_CORR_DATA_TBL VALUES(33, 131);
INSERT INTO PAL_CORR_DATA_TBL VALUES(34, 131);
INSERT INTO PAL_CORR_DATA_TBL VALUES(35, 129);
INSERT INTO PAL_CORR_DATA_TBL VALUES(36, 126);
INSERT INTO PAL_CORR_DATA_TBL VALUES(37, 126);
INSERT INTO PAL_CORR_DATA_TBL VALUES(38, 132);
INSERT INTO PAL_CORR_DATA_TBL VALUES(39, 137);
INSERT INTO PAL_CORR_DATA_TBL VALUES(40, 140);
INSERT INTO PAL_CORR_DATA_TBL VALUES(41, 142);
INSERT INTO PAL_CORR_DATA_TBL VALUES(42, 150);
INSERT INTO PAL_CORR_DATA_TBL VALUES(43, 159);
INSERT INTO PAL_CORR_DATA_TBL VALUES(44, 167);
INSERT INTO PAL_CORR_DATA_TBL VALUES(45, 170);
INSERT INTO PAL_CORR_DATA_TBL VALUES(46, 171);
INSERT INTO PAL_CORR_DATA_TBL VALUES(47, 172);
INSERT INTO PAL_CORR_DATA_TBL VALUES(48, 172);
INSERT INTO PAL_CORR_DATA_TBL VALUES(49, 174);
INSERT INTO PAL_CORR_DATA_TBL VALUES(50, 175);
INSERT INTO PAL_CORR_DATA_TBL VALUES(51, 172);
INSERT INTO PAL_CORR_DATA_TBL VALUES(52, 172);
INSERT INTO PAL_CORR_DATA_TBL VALUES(53, 174);
INSERT INTO PAL_CORR_DATA_TBL VALUES(54, 174);
INSERT INTO PAL_CORR_DATA_TBL VALUES(55, 169);
INSERT INTO PAL_CORR_DATA_TBL VALUES(56, 165);
INSERT INTO PAL_CORR_DATA_TBL VALUES(57, 156);
INSERT INTO PAL_CORR_DATA_TBL VALUES(58, 142);
INSERT INTO PAL_CORR_DATA_TBL VALUES(59, 131);
INSERT INTO PAL_CORR_DATA_TBL VALUES(60, 121);
INSERT INTO PAL_CORR_DATA_TBL VALUES(61, 112);
INSERT INTO PAL_CORR_DATA_TBL VALUES(62, 104);
INSERT INTO PAL_CORR_DATA_TBL VALUES(63, 102);
INSERT INTO PAL_CORR_DATA_TBL VALUES(64, 99);
INSERT INTO PAL_CORR_DATA_TBL VALUES(65, 99);
INSERT INTO PAL_CORR_DATA_TBL VALUES(66, 95);
INSERT INTO PAL_CORR_DATA_TBL VALUES(67, 88);
INSERT INTO PAL_CORR_DATA_TBL VALUES(68, 84);
INSERT INTO PAL_CORR_DATA_TBL VALUES(69, 84);
INSERT INTO PAL_CORR_DATA_TBL VALUES(70, 87);
INSERT INTO PAL_CORR_DATA_TBL VALUES(71, 89);
INSERT INTO PAL_CORR_DATA_TBL VALUES(72, 88);
INSERT INTO PAL_CORR_DATA_TBL VALUES(73, 85);
INSERT INTO PAL_CORR_DATA_TBL VALUES(74, 86);
INSERT INTO PAL_CORR_DATA_TBL VALUES(75, 89);
INSERT INTO PAL_CORR_DATA_TBL VALUES(76, 91);
INSERT INTO PAL_CORR_DATA_TBL VALUES(77, 91);
INSERT INTO PAL_CORR_DATA_TBL VALUES(78, 94);
INSERT INTO PAL_CORR_DATA_TBL VALUES(79, 101);
INSERT INTO PAL_CORR_DATA_TBL VALUES(80, 110);
INSERT INTO PAL_CORR_DATA_TBL VALUES(81, 121);
INSERT INTO PAL_CORR_DATA_TBL VALUES(82, 135);
INSERT INTO PAL_CORR_DATA_TBL VALUES(83, 145);
INSERT INTO PAL_CORR_DATA_TBL VALUES(84, 149);
INSERT INTO PAL_CORR_DATA_TBL VALUES(85, 156);
INSERT INTO PAL_CORR_DATA_TBL VALUES(86, 165);
INSERT INTO PAL_CORR_DATA_TBL VALUES(87, 171);
INSERT INTO PAL_CORR_DATA_TBL VALUES(88, 175);
INSERT INTO PAL_CORR_DATA_TBL VALUES(89, 177);
INSERT INTO PAL_CORR_DATA_TBL VALUES(90, 182);
INSERT INTO PAL_CORR_DATA_TBL VALUES(91, 193);
```

```
INSERT INTO PAL_CORR_DATA_TBL VALUES(92, 204);
INSERT INTO PAL_CORR_DATA_TBL VALUES(93, 208);
INSERT INTO PAL_CORR_DATA_TBL VALUES(94, 210);
INSERT INTO PAL_CORR_DATA_TBL VALUES(95, 215);
INSERT INTO PAL_CORR_DATA_TBL VALUES(96, 222);
INSERT INTO PAL_CORR_DATA_TBL VALUES(97, 228);
INSERT INTO PAL_CORR_DATA_TBL VALUES(98, 226);
INSERT INTO PAL_CORR_DATA_TBL VALUES(99, 222);
INSERT INTO PAL_CORR_DATA_TBL VALUES(100, 220);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR(100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR(100)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 0.4, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('USE_FFT', -1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('CALCULATE_PACF', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('BARTLETT', 1, NULL, NULL);
CALL _SYS_AFL.PAL_CORRELATION_FUNCTION_WITH_CONFIDENCE (PAL_CORR_DATA_TBL,
"#PAL_PARAMETER_TBL", ?);
```

**Expected Result**

RESULT:

| | LAG | CV | CF | PACF | ACF_CONFIDENCE_BOUND | PACF_CONFIDENCE_BOUND |
|---|---|---|---|---|---|---|
| 1 | 0 | 1583.9536 | 1 | 1 | *NULL* | *NULL* |
| 2 | 1 | 1520.8807360000003 | 0.9601801062859419 | 0.9601801062859419 | 0.1959963984540054 | 0.1959963984540054 |
| 3 | 2 | 1427.3562720000002 | 0.9011351544641207 | -0.2666184750683225 | 0.3305252297843596 | 0.1959963984540054 |
| 4 | 3 | 1312.6958079999997 | 0.8287463774191363 | -0.15441688686009125 | 0.4142893225610585 | 0.1959963984540054 |
| 5 | 4 | 1181.6069439999999 | 0.7459858319082072 | -0.12017581118371048 | 0.47371246617533797 | 0.1959963984540054 |
| 6 | 5 | 1041.04248 | 0.6572430404527001 | -0.07154561707715221 | 0.5168738167971348 | 0.1959963984540054 |
| 7 | 6 | 894.4932160000004 | 0.5647218554886964 | -0.065064608317029 | 0.5480386117711427 | 0.1959963984540054 |
| 8 | 7 | 742.178352 | 0.46856066490836595 | -0.08368642092350012 | 0.5699543596754845 | 0.1959963984540054 |
| 9 | 8 | 587.4534880000001 | 0.37087796511210935 | -0.06521318288237364 | 0.5845645801747793 | 0.1959963984540054 |
| 10 | 9 | 434.28782399999994 | 0.27417963758534336 | -0.045500858348922577 | 0.593534866150802 | 0.1959963984540054 |
| 11 | 10 | 286.46416000000005 | 0.1808538835986105 | -0.029585736722815954 | 0.5983805062297646 | 0.1959963984540054 |

## Related Information

Fast Fourier Transform [page 758]

# 5.9.6 Cumulative Distribution Function

This algorithm evaluates the probability of a variable x from the cumulative distribution function (CDF) or complementary cumulative distribution function (CCDF) for a given probability distribution.

**CDF**

CDF describes the lower tail probability of a probability distribution. It is the probability that a random variable X with a given probability distribution takes a value less than or equal to x. The CDF `F(x)` of a real-valued random variable X is given by:

$F(x)=P[X\leq x]$

### CCDF

CCDF describes the upper tail probability of a probability distribution. It is the probability that a random variable X with a given probability distribution takes a value greater than x. The CCDF $\bar{F}(x)$ of a real-valued random variable X is given by:

$\bar{F}(x)$=P[X>x]=1-F(x)

## Prerequisites

- The input data is numeric, not categorical.
- The input data and distribution parameters do not contain null data. The algorithm will issue errors when encountering null values.

## _SYS_AFL.PAL_DISTRIBUTION_FUNCTION

This function calculates the value of CDF or CCDF (depending on the parameter given by user) for a given distribution.

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<DISTRIBUTION PARAMETER table>` | IN |
| 3 | `<PARAMETER table>` | IN |
| 4 | `<RESULT table>` | OUT |

### Input Tables

| Table | Column | Reference for Output Table | Data Type | Description |
|---|---|---|---|---|
| DATA | 1st column | INPUT_DATA | DOUBLE or DECIMAL(p,s) | Input data |
| DISTRIBUTION PARAMETER | 1st column | | VARCHAR or NVARCHAR | Names of the distribution parameters. See Distribution Parameters Definition table for details. |

| Table | Column | Reference for Output Table | Data Type | Description |
|---|---|---|---|---|
| | 2nd column | | VARCHAR or NVARCHAR | Values of distribution parameters. See Distribution Parameters Definition table for details. |

Distribution Parameters Definition Table

The following parameters are mandatory and must be given a value.

| Distribution | Parameter Name | Parameter Value | Constraint |
|---|---|---|---|
| Uniform | "DistributionName" | "Uniform" | |
| | "Min" | "0.0" | Min < Max |
| | "Max" | "1.0" | |
| Normal | "DistributionName" | "Normal" | |
| | "Mean" | "0.0" | |
| | "Variance" | "1.0" | Variance > 0 |
| Weibull | "DistributionName" | "Weibull" | |
| | "Shape" | "1.0" | Shape > 0 |
| | "Scale" | "1.0" | Scale > 0 |
| Gamma | "DistributionName" | "Gamma" | |
| | "Shape" | "1.0" | Shape > 0 |
| | "Scale" | "1.0" | Scale > 0 |

> **Note**
>
> The names and values of the distribution parameters are not case sensitive.

### Parameter Table

The following parameter is optional. If it is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| LOWER_UPPER | INTEGER | 0 | • 0: Calculates the value of CDF<br>• 1: Calculates the value of CCDF |

### Output Table

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| RESULT | 1st column | INPUT_DATA (in input table) type; widening data type from INTEGER to DOUBLE. | INPUT_DATA (in input table) column name | Input data |
| | 2nd column | DOUBLE | PROBABILITY | Probabilities |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_DISTRPROB_DATA_TBL;
CREATE COLUMN TABLE PAL_DISTRPROB_DATA_TBL ("DATACOL" DOUBLE);
INSERT INTO PAL_DISTRPROB_DATA_TBL VALUES (37.4);
INSERT INTO PAL_DISTRPROB_DATA_TBL VALUES (277.9);
INSERT INTO PAL_DISTRPROB_DATA_TBL VALUES (463.2);
DROP TABLE PAL_DISTRPROB_DISTRPARAM_TBL;
CREATE COLUMN TABLE PAL_DISTRPROB_DISTRPARAM_TBL ("NAME" VARCHAR(50),"VALUE"
VARCHAR(50));;
INSERT INTO PAL_DISTRPROB_DISTRPARAM_TBL VALUES ('DistributionName', 'Weibull');
INSERT INTO PAL_DISTRPROB_DISTRPARAM_TBL VALUES ('Shape', '2.11995');
INSERT INTO PAL_DISTRPROB_DISTRPARAM_TBL VALUES ('Scale', '277.698');
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE
    #PAL_PARAMETER_TBL
    ("PARAM_NAME" VARCHAR(256), "INT_VALUE" INTEGER, "DOUBLE_VALUE" DOUBLE,
"STRING_VALUE" VARCHAR(1000));
INSERT INTO #PAL_PARAMETER_TBL VALUES ('LOWER_UPPER',0,NULL,NULL);
CALL _SYS_AFL.PAL_DISTRIBUTION_FUNCTION(PAL_DISTRPROB_DATA_TBL,
PAL_DISTRPROB_DISTRPARAM_TBL, #PAL_PARAMETER_TBL, ?);
```

### Expected Result

RESULT:

| | DATACOL | PROBABILITY |
|---|---|---|
| 1 | 37.4 | 0.014160024030305474 |
| 2 | 277.9 | 0.6326876484179044 |
| 3 | 463.2 | 0.9480935416104985 |

> **Note**
>
> The wrapper procedure generation approach to call PAL functions is still supported. For more information, refer to the older versions (prior to SAP HANA Platform 2.0 SPS 02) of this document.

## 5.9.7 Distribution Fitting

This algorithm aims to fit a probability distribution for a variable according to a series of measurements to the variable. There are many probability distributions of which some can be fitted more closely to the observed variable than others.

In PAL, you can choose one probability distribution type from a supported list (Normal, Gamma, Weibull, Exponential, Poisson and Uniform) and then PAL will calculate the optimized parameters of this distribution which fit the observed variable best.

PAL supports two distribution fitting interfaces: PAL_DISTRIBUTION_FIT and PAL_DISTRIBUTION_FIT_CENSORED. PAL_DISTRIBUTION_FIT fits un-censored data while PAL_DISTRIBUTION_FIT_CENSORED fits censored data.

Maximum likelihood and median rank are two estimation methods for finding the optimized parameters. In PAL, the maximum likelihood method supports all distribution types in the supporting list for un-censored data and supports Weibull distribution fitting for a mixture of left, right, and interval censored data. The median rank method supports the Weibull distribution fitting for both right-censored and un-censored data but it does not support other distribution types.

### Prerequisites

- The input data is numeric, not categorical.
- For PAL_DISTRIBUTION_FIT, the input data does not contain null data. The algorithm will issue errors when encountering null values.

### _SYS_AFL.PAL_DISTRIBUTION_FIT

This is a distribution fitting function with un-censored data.

**Overview of all Tables**

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |

| Position | Table Name | Parameter Type |
| --- | --- | --- |
| 3 | `<RESULT table>` | OUT |
| 4 | `<STATISTICS table>` | OUT |

### Input Table

| Table | Column | Data Type | Description |
| --- | --- | --- | --- |
| DATA | 1st column | DOUBLE, DECIMAL(p, s) | Input data |

### Parameter Table

Mandatory Parameter

The following parameter is mandatory and must be given a value.

| Name | Data Type | Description |
| --- | --- | --- |
| DISTRIBUTIONNAME | VARCHAR | Distribution name:<br>• Exponential<br>• Gamma<br>• Normal<br>• Poisson<br>• Uniform<br>• Weibull |

Optional Parameter

The following parameter is optional. If it is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description |
| --- | --- | --- | --- |
| OPTIMAL_METHOD | INTEGER | 1 (for Weibull distribution)<br>0 (for other distributions) | Estimation method:<br>• 0: Maximum likelihood<br>• 1: Median rank (only for Weibull distribution) |

> **Note**
>
> For Gamma or Weibull distribution, you can specify the start values of parameters for optimization. If the start values are not specified, the algorithm will calculate them automatically.

### Output Tables

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| RESULT | 1st column | NVARCHAR(100) | NAME | Name of distribution parameters |
| | 2nd column | NVARCHAR(100) | VALUE | Value of distribution parameters |
| STATISTICS | 1st column | NVARCHAR(100) | STAT_NAME | Name of statistics |
| | 2nd column | DOUBLE | STAT_VALUE | Value of statistics |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_DISTRIBUTION_FIT_DATA_TBL;
CREATE COLUMN TABLE PAL_DISTRIBUTION_FIT_DATA_TBL ("DATA" DOUBLE);
INSERT INTO PAL_DISTRIBUTION_FIT_DATA_TBL VALUES (71);
INSERT INTO PAL_DISTRIBUTION_FIT_DATA_TBL VALUES (83);
INSERT INTO PAL_DISTRIBUTION_FIT_DATA_TBL VALUES (92);
INSERT INTO PAL_DISTRIBUTION_FIT_DATA_TBL VALUES (104);
INSERT INTO PAL_DISTRIBUTION_FIT_DATA_TBL VALUES (120);
INSERT INTO PAL_DISTRIBUTION_FIT_DATA_TBL VALUES (134);
INSERT INTO PAL_DISTRIBUTION_FIT_DATA_TBL VALUES (138);
INSERT INTO PAL_DISTRIBUTION_FIT_DATA_TBL VALUES (146);
INSERT INTO PAL_DISTRIBUTION_FIT_DATA_TBL VALUES (181);
INSERT INTO PAL_DISTRIBUTION_FIT_DATA_TBL VALUES (191);
INSERT INTO PAL_DISTRIBUTION_FIT_DATA_TBL VALUES (206);
INSERT INTO PAL_DISTRIBUTION_FIT_DATA_TBL VALUES (226);
INSERT INTO PAL_DISTRIBUTION_FIT_DATA_TBL VALUES (276);
INSERT INTO PAL_DISTRIBUTION_FIT_DATA_TBL VALUES (283);
INSERT INTO PAL_DISTRIBUTION_FIT_DATA_TBL VALUES (291);
INSERT INTO PAL_DISTRIBUTION_FIT_DATA_TBL VALUES (332);
INSERT INTO PAL_DISTRIBUTION_FIT_DATA_TBL VALUES (351);
INSERT INTO PAL_DISTRIBUTION_FIT_DATA_TBL VALUES (401);
INSERT INTO PAL_DISTRIBUTION_FIT_DATA_TBL VALUES (466);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL ("PARAM_NAME"
VARCHAR(256), "INT_VALUE" INTEGER, "DOUBLE_VALUE" DOUBLE, "STRING_VALUE"
VARCHAR(1000));
INSERT INTO #PAL_PARAMETER_TBL VALUES ('DISTRIBUTIONNAME', NULL, NULL,
'WEIBULL');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('OPTIMAL_METHOD', 0, NULL, NULL);
CALL _SYS_AFL.PAL_DISTRIBUTION_FIT (PAL_DISTRIBUTION_FIT_DATA_TBL,
#PAL_PARAMETER_TBL, ?, ?);
```

### Expected Result

RESULT:

| | NAME | VALUE |
|---|---|---|
| 1 | DISTRIBUTIONNAME | WEIBULL |
| 2 | SCALE | 244.401 |
| 3 | SHAPE | 2.06698 |

STATISTICS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | LOGLIKELIHOOD | -115.11379382624872 |

## _SYS_AFL.PAL_DISTRIBUTION_FIT_CENSORED

This is a Weibull distribution fitting function with censored data. This release of PAL only supports the maximum likelihood estimation method on a mixture of left, right, and interval censored data and the median rank estimation method on right censored data.

### Overview of all Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<RESULT table>` | OUT |
| 4 | `<STATISTICS table>` | OUT |

### Input Table

| Table | Column | Column Data Type | Description |
|---|---|---|---|
| DATA | 1st column | DOUBLE, DECIMAL(p, s) | • For un-censored data, both 1st column and 2nd column should be observed data. Example: (17, 17)<br>• For right censored data, 1st column should be lower bound and 2nd column should be NULL. Example: (19, NULL)<br>• For left censored data, 1st column should be |

| Table | Column | Column Data Type | Description |
|---|---|---|---|
| | 2nd column | DOUBLE, DECIMAL(p, s) | NULL and 2nd column should be upper bound. Example: (NULL, 21)<br>• For interval censored data, 1st column should be lower bound and 2nd column should be upper bound. Example: (23, 50)<br>**Note**: The current release of PAL supports the following:<br>• Using the maximum likelihood method on uncensored data to estimate all distribution types in the supporting list;<br>• Using the maximum likelihood method on a mixture of left, right, and interval censored data to estimate weibull distribution;<br>• Using the media rank estimation method to estimate Weibull distribution on uncensored and right censored data. |

### Parameter Table

Mandatory Parameter

The following parameter is mandatory and must be given a value.

| Name | Data Type | Description |
|---|---|---|
| DISTRIBUTIONNAME | VARCHAR | Distribution name:<br>• Weibull |

Optional Parameter

The following parameter is optional. If it is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| OPTIMAL_METHOD | INTEGER | 0 | Estimation method:<br>• 0: Maximum likelihood<br>• 1: Median rank (only for weibull distribution) |

### Output Tables

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| RESULT | 1st column | NVARCHAR(100) | NAME | Name of distribution parameters |
| | 2nd column | NVARCHAR(100) | VALUE | Value of distribution parameters |
| STATISTICS | 1st column | NVARCHAR(100) | STAT_NAME | Name of statistics |
| | 2nd column | DOUBLE | STAT_VALUE | Value of statistics |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

#### Example 1: Median Rank Estimation Method

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_DISTRIBUTION_FIT_CENSORED_DATA_TBL;
CREATE COLUMN TABLE PAL_DISTRIBUTION_FIT_CENSORED_DATA_TBL ("LEFT" DOUBLE,
"RIGHT" DOUBLE);
INSERT INTO PAL_DISTRIBUTION_FIT_CENSORED_DATA_TBL VALUES (17, NULL); --right
censored data.
INSERT INTO PAL_DISTRIBUTION_FIT_CENSORED_DATA_TBL VALUES (55, NULL);
INSERT INTO PAL_DISTRIBUTION_FIT_CENSORED_DATA_TBL VALUES (55, NULL);
INSERT INTO PAL_DISTRIBUTION_FIT_CENSORED_DATA_TBL VALUES (71, 71); --uncensored
data
INSERT INTO PAL_DISTRIBUTION_FIT_CENSORED_DATA_TBL VALUES (83, 83);
INSERT INTO PAL_DISTRIBUTION_FIT_CENSORED_DATA_TBL VALUES (85, NULL);
INSERT INTO PAL_DISTRIBUTION_FIT_CENSORED_DATA_TBL VALUES (92, 92);
INSERT INTO PAL_DISTRIBUTION_FIT_CENSORED_DATA_TBL VALUES (104, 104);
INSERT INTO PAL_DISTRIBUTION_FIT_CENSORED_DATA_TBL VALUES (115, NULL);
INSERT INTO PAL_DISTRIBUTION_FIT_CENSORED_DATA_TBL VALUES (120, 120);
INSERT INTO PAL_DISTRIBUTION_FIT_CENSORED_DATA_TBL VALUES (125, NULL);
INSERT INTO PAL_DISTRIBUTION_FIT_CENSORED_DATA_TBL VALUES (134, 134);
INSERT INTO PAL_DISTRIBUTION_FIT_CENSORED_DATA_TBL VALUES (138, 138);
INSERT INTO PAL_DISTRIBUTION_FIT_CENSORED_DATA_TBL VALUES (146, 146);
INSERT INTO PAL_DISTRIBUTION_FIT_CENSORED_DATA_TBL VALUES (163, NULL);
INSERT INTO PAL_DISTRIBUTION_FIT_CENSORED_DATA_TBL VALUES (171, NULL);
INSERT INTO PAL_DISTRIBUTION_FIT_CENSORED_DATA_TBL VALUES (181, 181);
INSERT INTO PAL_DISTRIBUTION_FIT_CENSORED_DATA_TBL VALUES (191, 191);
INSERT INTO PAL_DISTRIBUTION_FIT_CENSORED_DATA_TBL VALUES (195, NULL);
INSERT INTO PAL_DISTRIBUTION_FIT_CENSORED_DATA_TBL VALUES (206, 206);
INSERT INTO PAL_DISTRIBUTION_FIT_CENSORED_DATA_TBL VALUES (226, 226);
```

```
INSERT INTO PAL_DISTRIBUTION_FIT_CENSORED_DATA_TBL VALUES (260, NULL);
INSERT INTO PAL_DISTRIBUTION_FIT_CENSORED_DATA_TBL VALUES (276, 276);
INSERT INTO PAL_DISTRIBUTION_FIT_CENSORED_DATA_TBL VALUES (283, 283);
INSERT INTO PAL_DISTRIBUTION_FIT_CENSORED_DATA_TBL VALUES (291, 291);
INSERT INTO PAL_DISTRIBUTION_FIT_CENSORED_DATA_TBL VALUES (332, 332);
INSERT INTO PAL_DISTRIBUTION_FIT_CENSORED_DATA_TBL VALUES (351, 351);
INSERT INTO PAL_DISTRIBUTION_FIT_CENSORED_DATA_TBL VALUES (384, NULL);
INSERT INTO PAL_DISTRIBUTION_FIT_CENSORED_DATA_TBL VALUES (401, 401);
INSERT INTO PAL_DISTRIBUTION_FIT_CENSORED_DATA_TBL VALUES (466, 466);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL ("PARAM_NAME"
VARCHAR(256), "INT_VALUE" INTEGER, "DOUBLE_VALUE" DOUBLE, "STRING_VALUE"
VARCHAR(1000));
INSERT INTO #PAL_PARAMETER_TBL VALUES ('DISTRIBUTIONNAME', NULL, NULL,
'WEIBULL');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('OPTIMAL_METHOD', 1, NULL, NULL);
CALL _SYS_AFL.PAL_DISTRIBUTION_FIT_CENSORED
(PAL_DISTRIBUTION_FIT_CENSORED_DATA_TBL, #PAL_PARAMETER_TBL, ?, ?);
```

**Expected Result**

RESULT:

| | NAME | VALUE |
|---|---|---|
| 1 | DISTRIBUTIONNAME | WEIBULL |
| 2 | SCALE | 277.697 |
| 3 | SHAPE | 2.11995 |

STATISTICS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| | | |
| | | |

**Note**: The median rank estimation method only supports Weibull distribution, and this example does not have statistics output.

**Example 2: Maximum Likelihood Estimation Method**

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_DISTRIBUTION_FIT_CENSORED_DATA_TBL;
CREATE COLUMN TABLE PAL_DISTRIBUTION_FIT_CENSORED_DATA_TBL ("LEFT" DOUBLE,
"RIGHT" DOUBLE);
INSERT INTO PAL_DISTRIBUTION_FIT_CENSORED_DATA_TBL VALUES (NULL, 0.04);
INSERT INTO PAL_DISTRIBUTION_FIT_CENSORED_DATA_TBL VALUES (100, 100);
INSERT INTO PAL_DISTRIBUTION_FIT_CENSORED_DATA_TBL VALUES (0.04, NULL);
INSERT INTO PAL_DISTRIBUTION_FIT_CENSORED_DATA_TBL VALUES (15, 30);
INSERT INTO PAL_DISTRIBUTION_FIT_CENSORED_DATA_TBL VALUES (0.04, 0.04);
INSERT INTO PAL_DISTRIBUTION_FIT_CENSORED_DATA_TBL VALUES (100, 100);
INSERT INTO PAL_DISTRIBUTION_FIT_CENSORED_DATA_TBL VALUES (1, 1);
INSERT INTO PAL_DISTRIBUTION_FIT_CENSORED_DATA_TBL VALUES (1, 1);
INSERT INTO PAL_DISTRIBUTION_FIT_CENSORED_DATA_TBL VALUES (0.04, 0.04);
INSERT INTO PAL_DISTRIBUTION_FIT_CENSORED_DATA_TBL VALUES (1, NULL);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL ("PARAM_NAME"
VARCHAR(256), "INT_VALUE" INTEGER, "DOUBLE_VALUE" DOUBLE, "STRING_VALUE"
VARCHAR(1000));
INSERT INTO #PAL_PARAMETER_TBL VALUES ('DISTRIBUTIONNAME', NULL, NULL,
'WEIBULL');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('OPTIMAL_METHOD', 0, NULL, NULL);
CALL _SYS_AFL.PAL_DISTRIBUTION_FIT_CENSORED
(PAL_DISTRIBUTION_FIT_CENSORED_DATA_TBL, #PAL_PARAMETER_TBL, ?, ?);
```

**Expected Result**

RESULT:

| | NAME | VALUE |
|---|---|---|
| 1 | DISTRIBUTIONNAME | WEIBULL |
| 2 | SCALE | 8.42129 |
| 3 | SHAPE | 0.335718 |

STATISTICS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | LOGLIKELIHOOD | -23.62960158005567 |

> **Note**
>
> The wrapper procedure generation approach to call PAL functions is still supported. For more information, refer to the older versions (prior to SAP HANA Platform 2.0 SPS 02) of this document.

## 5.9.8 Distribution Quantile

This algorithm evaluates the inverse $F^{-1}(x)$ of the cumulative distribution function (CDF) or the inverse $\bar{F}^{-1}(x)$ of the complementary cumulative distribution function (CCDF) for a given probability p and probability distribution.

The CDF `F(x)` of a real-valued random variable X is given by

`F(x)=P[X≤x]`

The CCDF $\bar{F}(x)$ of a real-valued random variable X is given by

$\bar{F}(x)$`=P[X>x]=1-F(x)`

### Prerequisites

- The input data is numeric, not categorical.
- The input data and distribution parameters do not contain null data. The algorithm will issue errors when encountering null values.

### _SYS_AFL.PAL_DISTRIBUTION_QUANTILE

This function calculates quantiles for a given distribution.

### Overview of all Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<DISTRIBUTION_PARAMETER table>` | IN |
| 3 | `<PARAMETER table>` | IN |
| 4 | `<RESULT table>` | OUT |

### Input Tables

| Table | Column | Data Type | Description |
|---|---|---|---|
| DATA | 1st column | DOUBLE or DECIMAL(p, s) | Probabilities. Must be in the open interval (0.0, 1.0). |
| DISTRIBUTION_PARAMETER | 1st column | VARCHAR or NVARCHAR | Names of the distribution parameters. See Distribution Parameters Definition table for details. |
| | 2nd column | VARCHAR or NVARCHAR | Values of the distribution parameters. See Distribution Parameters Definition table for details. |

Distribution Parameters Definition

The following parameters are mandatory and must be given a value.

| Distribution | Parameter Name | Parameter Value | Constraint |
|---|---|---|---|
| Uniform | "DistributionName" | "Uniform" | |
| | "Min" | "0.0" | Min < Max |
| | "Max" | "1.0" | |
| Normal | "DistributionName" | "Normal" | |
| | "Mean" | "0.0" | |
| | "Variance" | "1.0" | Variance > 0 |
| Weibull | "DistributionName" | "Weibull" | |
| | "Shape" | "1.0" | Shape > 0 |

| Distribution | Parameter Name | Parameter Value | Constraint |
|---|---|---|---|
| | "Scale" | "1.0" | Scale > 0 |
| Gamma | "DistributionName" | "Gamma" | |
| | "Shape" | "1.0" | Shape > 0 |
| | "Scale" | "1.0" | Scale > 0 |

> **Note**
>
> The names and values of the distribution parameters are not case sensitive.

### Parameter Table

The following parameter is optional. If it is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| LOWER_UPPER | INTEGER | 0 | • 0: Calculates the inverse of CDF<br>• 1: Calculates the inverse of CCDF |

Output Table

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| RESULT | 1st column | DOUBLE | INPUT_DATA | Input probabilities |
| | 2nd column | DOUBLE | QUANTILE | Quantiles |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_DISTRQUANTILE_DATA_TBL;
CREATE COLUMN TABLE PAL_DISTRQUANTILE_DATA_TBL (
    "DATACOL" DOUBLE
);
INSERT INTO PAL_DISTRQUANTILE_DATA_TBL VALUES (0.3);
INSERT INTO PAL_DISTRQUANTILE_DATA_TBL VALUES (0.5);
INSERT INTO PAL_DISTRQUANTILE_DATA_TBL VALUES (0.632);
INSERT INTO PAL_DISTRQUANTILE_DATA_TBL VALUES (0.8);
DROP TABLE PAL_DISTRQUANTILE_DISTRPARAM_TBL;
CREATE COLUMN TABLE PAL_DISTRQUANTILE_DISTRPARAM_TBL (
    "NAME" VARCHAR(50),
    "VALUE" VARCHAR(50)
);
```

```
INSERT INTO PAL_DISTRQUANTILE_DISTRPARAM_TBL VALUES ('DistributionName',
'Weibull');
INSERT INTO PAL_DISTRQUANTILE_DISTRPARAM_TBL VALUES ('Shape', '2.11995');
INSERT INTO PAL_DISTRQUANTILE_DISTRPARAM_TBL VALUES ('Scale', '277.698');
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('LOWER_UPPER',0,null,null);
CALL "_SYS_AFL"."PAL_DISTRIBUTION_QUANTILE"(PAL_DISTRQUANTILE_DATA_TBL,
PAL_DISTRQUANTILE_DISTRPARAM_TBL, #PAL_PARAMETER_TBL, ?);
```

**Expected Result**

| | INPUT_DATA | QUANTILE |
|---|---|---|
| 1 | 0.3 | 170.7558541487521 |
| 2 | 0.5 | 233.60850568865035 |
| 3 | 0.632 | 277.6550753190647 |
| 4 | 0.8 | 347.5864951691035 |

> **Note**
>
> The wrapper procedure generation approach to call PAL functions is still supported. For more information, refer to the older versions (prior to SAP HANA Platform 2.0 SPS 02) of this document.

## 5.9.9 Entropy

This function is used to calculate the information entropy of attributes.

Assume an attribute x has n distinct values. The definition of entropy H is:

$$H=-\sum_{i=1}^{n} p(x_i)\log p(x_i)$$

where $p_i$ is the probability of the ith value $x_i$

### Prerequisites

- Only attributes with data type VARCHAR, NVARCHAR, and INTEGER are not processed.

### _SYS_AFL.PAL_ENTROPY

This function calculates the information entropy of attributes.

**Overview of All Tables**

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<INPUT_1 table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<RESULT_1 table>` | OUT |
| 4 | `<RESULT_2 table>` | OUT |

### Input Table

| Table | Column | Column Data Type | Description |
|---|---|---|---|
| INPUT_1 | All columns | VARCHAR, NVARCHAR, INTEGER, or DOUBLE | Attribute data. Attributes with continuous data type are ignored. |

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If the parameter value is not specified, PAL uses its default value.

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| COLUMN | VARCHAR or NVARCHAR | No default value | Names of the columns to be processed. There could be multiple occurrences of this parameter. If this parameter is not used, all columns are processed. If this parameter is used, only the specified columns are processed. |
| DISTINCT_VALUE_COUNT_DETAIL | NTEGER | 1 | Indicates whether to output the details of distinct value counts:<br>• 0: Does not output detailed distinct value count.<br>• 1: Outputs detailed distinct value count. |

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| THREAD_RATIO | DOUBLE | 0 | Specifies the ratio of total number of threads that can be used by this function. The value range is from 0 to 1, where 0 means only using 1 thread, and 1 means using at most all the currently available threads. Values outside the range are ignored and this function heuristically determines the number of threads to use. |

### Output Table

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| | 1st column | NVARCHAR | COLUMN_NAME | Name of columns |
| | 2nd column | DOUBLE | ENTROPY | Entropy of columns |
| | 3rd column | INTEGER | COUNT_OF_DISTINCT_VALUES | Count of distinct values |

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| RESULT_1 | 1st column | NVARCHAR | COLUMN_NAME | Name of columns |
| | 2nd column | NVARCHAR | DISTINCT_VALUE | Distinct values of columns |
| | 3rd column | INTEGER | COUNT | Count of each distinct value |

## Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA DM_PAL;
DROP TABLE  PAL_ENTROPY_DATA_TBL;
CREATE COLUMN TABLE PAL_ENTROPY_DATA_TBL (
    "OUTLOOK" VARCHAR(20),
    "TEMP" INTEGER,
```

```
    "HUMIDITY" DOUBLE,
    "WINDY" VARCHAR(10),
    "CLASS" VARCHAR(20)
);
INSERT INTO PAL_ENTROPY_DATA_TBL VALUES ('Sunny', 75, 70.0, 'Yes', 'Play');
INSERT INTO PAL_ENTROPY_DATA_TBL VALUES ('Sunny', NULL, 90.0, 'Yes', 'Do not
Play');
INSERT INTO PAL_ENTROPY_DATA_TBL VALUES ('Sunny', 85, NULL, 'No', 'Do not Play');
INSERT INTO PAL_ENTROPY_DATA_TBL VALUES ('Sunny', 72, 95.0, 'No', 'Do not Play');
INSERT INTO PAL_ENTROPY_DATA_TBL VALUES (NULL, NULL, 70.0, NULL, 'Play');
INSERT INTO PAL_ENTROPY_DATA_TBL VALUES ('Overcast', 72.0, 90, 'Yes', 'Play');
INSERT INTO PAL_ENTROPY_DATA_TBL VALUES ('Overcast', 83.0, 78, 'No', 'Play');
INSERT INTO PAL_ENTROPY_DATA_TBL VALUES ('Overcast', 64.0, 65, 'Yes', 'Play');
INSERT INTO PAL_ENTROPY_DATA_TBL VALUES ('Overcast', 81.0, 75, 'No', 'Play');
INSERT INTO PAL_ENTROPY_DATA_TBL VALUES (NULL, 71, 80.0, 'Yes', 'Do not Play');
INSERT INTO PAL_ENTROPY_DATA_TBL VALUES ('Rain', 65, 70.0, 'Yes', 'Do not Play');
INSERT INTO PAL_ENTROPY_DATA_TBL VALUES ('Rain', 75, 80.0, 'No', 'Play');
INSERT INTO PAL_ENTROPY_DATA_TBL VALUES ('Rain', 68, 80.0, 'No', 'Play');
INSERT INTO PAL_ENTROPY_DATA_TBL VALUES ('Rain', 70, 96.0, 'No', 'Play');
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (100)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('COLUMN', NULL, NULL, 'TEMP');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('COLUMN', NULL, NULL, 'WINDY');
--INSERT INTO #PAL_PARAMETER_TBL VALUES ('DISTINCT_VALUE_COUNT_DETAIL', 0, NULL,
NULL);
CALL _SYS_AFL.PAL_ENTROPY(PAL_ENTROPY_DATA_TBL, #PAL_PARAMETER_TBL, ?, ?);
```

## Expected Result

Entropy output table:

| | COLUMN_NAME | ENTROPY | COUNT_OF_DISTINCT_VALUES |
|---|---|---|---|
| 1 | TEMP | 2.2538575896013526 | 10 |
| 2 | WINDY | 0.6901856760188042 | 2 |

Detailed distinct value count output table:

| | COLUMN_NAME | DISTINCT_VALUE | COUNT |
|---|---|---|---|
| 1 | TEMP | 75 | 2 |
| 2 | TEMP | 85 | 1 |
| 3 | TEMP | 72 | 2 |
| 4 | TEMP | 83 | 1 |
| 5 | TEMP | 64 | 1 |
| 6 | TEMP | 81 | 1 |
| 7 | TEMP | 71 | 1 |
| 8 | TEMP | 65 | 1 |
| 9 | TEMP | 68 | 1 |
| 10 | TEMP | 70 | 1 |
| 11 | WINDY | Yes | 6 |
| 12 | WINDY | No | 7 |

> **Note**
>
> The wrapper procedure generation approach to call PAL functions is still supported. For more information, refer to the older versions (prior to SAP HANA Platform 2.0 SPS 02) of this document.

## 5.9.10 Equal Variance Test

This function is used to test the equality of two random variances using F-test. The null hypothesis is that two independent normal variances are equal. The observed sums of some selected squares are then examined to see whether their ratio is significantly incompatible with this null hypothesis.

Let $x_1, x_2, \ldots, x_n$ and $y_1, y_2, \ldots, y_n$ be independent and identically distributed samples from two populations.

Let the sample mean of x and y be:

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n} \quad \text{and} \quad \bar{y} = \frac{\sum_{i=1}^{m} y_i}{m}$$

Therefore, the sample variance of x and y are:

$$s_x^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2 \quad \text{and} \quad s_y^2 = \frac{1}{m-1}\sum_{i=1}^{m}(y_i - \bar{y})^2$$

Then the test statistics is:

$$F = \frac{s_x^2}{s_y^2}$$

The F value will be used to calculate a p-value by comparing the value of F to an F-distribution. The degree of freedom is set to `(n-1)` and `(m-1)`.

## Prerequisites

- The input data has two tables and each table has only one column with the type of INTEGER or DOUBLE.
- The input data does not contain null value. The algorithm will issue errors when encountering null values.
- The minimum number of valid data of each input data table is two.

## _SYS_AFL.PAL_EQUAL_VARIANCE_TEST

This function tests the equality of variances between two input data.

### Overview of all Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<INPUT_1 table>` | IN |
| 2 | `<INPUT_2 table>` | IN |
| 3 | `<PARAMETER table>` | IN |
| 4 | `<STATISTICS table>` | OUT |

### Input Tables

| Table | Column | Data Type | Description |
|---|---|---|---|
| INPUT_1 | 1st column | INTEGER, DOUBLE, or DECIMAL(p, s) | Attribute data |
| INPUT_2 | 1st column | INTEGER, DOUBLE, or DECIMAL(p, s) | Attribute data |

### Parameter Table

Mandatory Parameters

None.

Optional Parameter

The following parameter is optional. If it is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| TEST_TYPE | INTEGER | 0 | The alternative hypothesis type.<br>• 0: Two sides<br>• 1: Less<br>• 2: Greater |

### Output Table

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| STATISTICS | 1st column | NVARCHAR | STAT_NAME | Name of statistics |
| | 2nd column | DOUBLE | STAT_VALUE | Value of statistics |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_VAREQUALTEST_DATA1_TBL;
CREATE COLUMN TABLE PAL_VAREQUALTEST_DATA1_TBL (
    "X" INTEGER
);
INSERT INTO PAL_VAREQUALTEST_DATA1_TBL VALUES (1);
INSERT INTO PAL_VAREQUALTEST_DATA1_TBL VALUES (2);
INSERT INTO PAL_VAREQUALTEST_DATA1_TBL VALUES (4);
INSERT INTO PAL_VAREQUALTEST_DATA1_TBL VALUES (7);
INSERT INTO PAL_VAREQUALTEST_DATA1_TBL VALUES (3);
DROP TABLE PAL_VAREQUALTEST_DATA2_TBL;
CREATE COLUMN TABLE PAL_VAREQUALTEST_DATA2_TBL (
    "Y" DOUBLE
);
INSERT INTO PAL_VAREQUALTEST_DATA2_TBL VALUES (10);
INSERT INTO PAL_VAREQUALTEST_DATA2_TBL VALUES (15);
INSERT INTO PAL_VAREQUALTEST_DATA2_TBL VALUES (12);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('TEST_TYPE',0,null,null); //default value
is 0, it can be {0,1,2}
CALL "_SYS_AFL"."PAL_EQUAL_VARIANCE_TEST"(PAL_VAREQUALTEST_DATA1_TBL,
PAL_VAREQUALTEST_DATA2_TBL, #PAL_PARAMETER_TBL, ?);
```

### Expected Result

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | F Value | 0.8368421052631579 |
| 2 | numerator degree of freedom | 4 |
| 3 | denominator degree of free... | 2 |
| 4 | p-value | 0.7837125674251348 |

> **Note**
>
> The wrapper procedure generation approach to call PAL functions is still supported. For more information, refer to the older versions (prior to SAP HANA Platform 2.0 SPS 02) of this document.

## 5.9.11 Factor Analysis

Factor analysis is a statistical method that tries to extract a low number of unobserved variables – factors – that can best describe the covariance pattern of a larger set of observed variables.

It can be used to reduce dimension of the data as well as to reveal the underlying relationships between the observed variables. It is related to PCA, but they are not the same.

Suppose we have a set of observed variables $x_1,\ldots,x_p$, let $x=(x_1,\ldots,x_p)^T$, then the factor analysis model can be written in form as

$x=\mu+Lf+\varepsilon$.

In the above form, $\mu=(\mu_1,\ldots,\mu_p)^T$, where $\mu_i$ denotes the population mean of variable $x_i$. $L$ is the matrix of factor loadings and $f$ is the vector of factors. Suppose the number of unobserved variables is $k$ $(k\le p)$, then $L=(l_{ij})$ is a matrix of size $p\times k$, where $l_{ij}$ is the loading of the $i$th observed variable on the $j$th factor and $f=(f_1,\ldots,f_k)^T$, where $f_j$ is the $j$th factor. And finally, the last term $\varepsilon=(\varepsilon_1,\ldots,\varepsilon_p)^T$ denotes the independent specific factors. In general, we want $k\ll p$ so that we can use a relatively low number of latent factors to explain the covariance structure of the whole dataset.

Several assumptions are imposed on the model to make sure of its uniqueness:

- $E(f_j)=0, E(\varepsilon_i)=0$;
- $Cov(f)=I$, $Cov(\varepsilon)=\Psi$, where $\Psi$ is a $p\times p$ diagonal matrix;
- $f$ and $\varepsilon$ are independent.

With the above assumptions, the factor analysis model can be alternatively specified as

$Cov(x)=LL^T+\Psi$

Note that for any orthogonal matrix $U$, the model and its assumptions also hold for $L'=LU$ and $f'=U^T f$. Hence the factors and factor loadings are only unique up to an orthogonal transformation. This fact makes it practicable to perform factor rotation, which will be discussed later, to obtain more interpretable results.

There are different methods to estimate the parameters of the model. PAL currently supports the principal component method.

**Principal Component Method**

Suppose the eigendecomposition of matrix $\Sigma=Cov(x)$ (if x should be standardized, $\Sigma=Cor(x)$ is used) is

$$\Sigma = U\Lambda U^T = \sum_{i=1}^{p} \lambda_i u_i u_i^T,$$

where $U=(u_1,\ldots,u_p)$, $\Lambda=diag(\lambda_1,\ldots,\lambda_p)$, $\lambda_1 \geq \ldots \lambda_p \geq 0$. Principal component method is to approximate the covariance structure $\Sigma$ with only first and main few terms of the sum:

$$\Sigma = \sum_{i=1}^{p} \lambda_i u_i u_i^T \approx \sum_{i=1}^{k} \lambda_i u_i u_i^T = LL^T,$$

which yields the estimation of the factor loadings $(L)_{ij} = \sqrt{\lambda_j} u_{ij}$.

The unrotated factor analysis forces the factors to be orthogonal and usually has many observed variables loaded substantially on more than one factor. This makes the result hard to interpret. Hence factor rotation is often used to facilitate the interpretation by rotating the axes so that each observed variable loads mainly on only one factor. There are two main types of rotation: orthogonal rotation where the new axes are still orthogonal and oblique rotation where orthogonal is not required.

For orthogonal rotation, PAL currently supports varimax rotation. For oblique rotation, PAL currently supports promax rotation.

## _SYS_AFL.PAL_FACTOR_ANALYSIS

This procedure performs factor analysis with given data matrix.

### Overview of all Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<EIGENVALUES table>` | OUT |
| 4 | `<VARIANCE_EXPLANATION table>` | OUT |
| 5 | `<COMMUNALITIES table>` | OUT |
| 6 | `<LOADINGS table>` | OUT |
| 7 | `<ROTATED_LOADINGS table>` | OUT |
| 8 | `<STRUCTURE table>` | OUT |
| 9 | `<ROTATION table>` | OUT |
| 10 | `<FACTOR_CORRELATION table>` | OUT |
| 11 | `<SCORE_MODEL table>` | OUT |

| Position | Table Name | Parameter Type |
|---|---|---|
| 12 | `<SCORES table>` | OUT |
| 13 | `<STATISTICS table>` | OUT |

### Input Table

| Table | Column | Reference for Output Table | Data Type | Description |
|---|---|---|---|---|
| DATA | 1st column | ID | INTEGER, VARCHAR, or NVARCHAR | ID |
| | Observed variable column | OBSERVED_VARS | INTEGER, DOUBLE, or DECIMAL(p, s) | Observed variables |

### Parameter Table

Mandatory Parameters

The following parameter is mandatory and must be given a value.

| Name | Data Type | Description |
|---|---|---|
| FACTOR_NUMBER | INTEGER | Number of factors. |

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| METHOD | INTEGER | 0 | Specifies the method used for factor analysis:<br>• 0: Principal component method<br>Currently PAL only supports the principal component method. |
| ROTATION | INTEGER | 1 | Specifies the rotation to be performed on loadings:<br>• 0: None<br>• 1: Varimax rotation<br>• 2: Promax rotation |

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| SCORE | INTEGER | 1 | Specifies the method to compute factor scores:<br>• 0: None<br>• 1: Regression method |
| COR | INTEGER | 1 | • 0: Uses covariance matrix to perform factor analysis.<br>• 1: Uses correlation matrix instead. |
| KAPPA | DOUBLE | 4 | Power of promax rotation. (Only valid when ROTATION is 2.) |

### Output Table

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| EIGENVALUES | 1st column | NVARCHAR(100) | FACTOR_ID | Factor ID |
| | 2nd column | DOUBLE | EIGENVALUE | Eigenvalue (i.e. variance explained) |
| | 3rd column | DOUBLE | VAR_PROP | Variance proportion to the total variance explained. |
| | 4th column | DOUBLE | CUM_VAR_PROP | Cumulative variance proportion to the total variance explained. |
| VARIANCE_EXPLANATION | 1st column | NVARCHAR(100) | FACTOR_ID | Factor ID |
| | 2nd column | DOUBLE | VAR | Variance explained without rotation. |
| | 3rd column | DOUBLE | VAR_PROP | Variance proportion to the total variance explained without rotation. |
| | 4th column | DOUBLE | CUM_VAR_PROP | Cumulative variance proportion to the total variance explained without rotation. |

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| | 5th column | DOUBLE | ROT_VAR | Variance explained with rotation. |
| | 6th column | DOUBLE | ROT_VAR_PROP | Variance proportion to the total variance explained with rotation.<br>Note that there is no rotated variance proportion when performing oblique rotation since the rotated factors are correlated. |
| | 7th column | DOUBLE | ROT_CUM_VAR_PROP | Cumulative variance proportion to the total variance explained with rotation. |
| COMMUNALITIES | 1st column | NVARCHAR(100) | NAME | Name |
| | Communalities columns | DOUBLE | OBSERVED_VARS (in input table) column name | Communalities |
| LOADINGS | 1st column | NVARCHAR(100) | FACTOR_ID | Factor ID |
| | Loadings columns | DOUBLE | LOADINGS_ + OBSERVED_VARS (in input table) column name | Loadings |
| ROTATED_LOADINGS | 1st column | NVARCHAR(100) | FACTOR_ID | Factor ID |
| | Rotated loadings columns | DOUBLE | ROT_LOADINGS_ + OBSERVED_VARS (in input table) column name | Rotated loadings |
| STRUCTURE | 1st column | NVARCHAR(100) | FACTOR_ID | Factor ID |
| | Structure columns | DOUBLE | STRUCTURE_ + OBSERVED_VARS (in input table) column name | Structure matrix. It is empty when rotation is not oblique. |
| ROTATION | 1st column | NVARCHAR(100) | ROTATION | Rotation |

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| | Rotation columns | DOUBLE | ROTATION_ + i (i sequences from 1 to number of columns in OBSERVED_VARS (in input table) | Rotation matrix |
| FACTOR_CORRELATION | 1st column | NVARCHAR(100) | FACTOR_ID | Factor ID |
| | Factor correlation columns | DOUBLE | FACTOR_ + i (i sequences from 1 to number of columns in OBSERVED_VARS (in input table) | Factor correlation matrix. It is empty when rotation is not oblique. |
| SCORE_MODEL | 1st column | NVARCHAR(100) | NAME | Factor ID, MEAN and SD |
| | Other columns | DOUBLE | OBSERVED_VARS (in input table) column name | Score coefficients, and means and standard deviations of observed variables. |
| SCORES | 1st column | ID (in input data) data type | ID (in input table) column name | ID |
| | Other columns | DOUBLE | FACTOR_ + i (i sequences from 1 to number of rows in OBSERVED_VARS (in input table)) | Scores |
| STATISTICS | 1st column | NVARCHAR(100) | STAT_NAME | Placeholder for future features |
| | 2nd column | DOUBLE | STAT_VALUE | |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_FACTOR_ANALYSIS_DATA_TBL;
CREATE COLUMN TABLE PAL_FACTOR_ANALYSIS_DATA_TBL ("ID" INTEGER, "X1" DOUBLE,
"X2" DOUBLE, "X3" DOUBLE, "X4" DOUBLE, "X5" DOUBLE, "X6" DOUBLE);
INSERT INTO PAL_FACTOR_ANALYSIS_DATA_TBL VALUES (1, 1, 1, 3, 3, 1, 1);
INSERT INTO PAL_FACTOR_ANALYSIS_DATA_TBL VALUES (2, 1, 2, 3, 3, 1, 1);
INSERT INTO PAL_FACTOR_ANALYSIS_DATA_TBL VALUES (3, 1, 1, 3, 4, 1, 1);
INSERT INTO PAL_FACTOR_ANALYSIS_DATA_TBL VALUES (4, 1, 1, 3, 3, 1, 2);
INSERT INTO PAL_FACTOR_ANALYSIS_DATA_TBL VALUES (5, 1, 1, 3, 3, 1, 1);
INSERT INTO PAL_FACTOR_ANALYSIS_DATA_TBL VALUES (6, 1, 1, 1, 1, 3, 3);
```

```
INSERT INTO PAL_FACTOR_ANALYSIS_DATA_TBL VALUES (7, 1, 2, 1, 1, 3, 3);
INSERT INTO PAL_FACTOR_ANALYSIS_DATA_TBL VALUES (8, 1, 1, 1, 2, 3, 3);
INSERT INTO PAL_FACTOR_ANALYSIS_DATA_TBL VALUES (9, 1, 2, 1, 1, 3, 4);
INSERT INTO PAL_FACTOR_ANALYSIS_DATA_TBL VALUES (10, 1, 1, 1, 1, 3, 3);
INSERT INTO PAL_FACTOR_ANALYSIS_DATA_TBL VALUES (11, 3, 3, 1, 1, 1, 1);
INSERT INTO PAL_FACTOR_ANALYSIS_DATA_TBL VALUES (12, 3, 4, 1, 1, 1, 1);
INSERT INTO PAL_FACTOR_ANALYSIS_DATA_TBL VALUES (13, 3, 3, 1, 2, 1, 1);
INSERT INTO PAL_FACTOR_ANALYSIS_DATA_TBL VALUES (14, 3, 3, 1, 1, 1, 2);
INSERT INTO PAL_FACTOR_ANALYSIS_DATA_TBL VALUES (15, 3, 3, 1, 1, 1, 1);
INSERT INTO PAL_FACTOR_ANALYSIS_DATA_TBL VALUES (16, 4, 4, 5, 5, 6, 6);
INSERT INTO PAL_FACTOR_ANALYSIS_DATA_TBL VALUES (17, 5, 6, 4, 6, 4, 5);
INSERT INTO PAL_FACTOR_ANALYSIS_DATA_TBL VALUES (18, 6, 5, 6, 4, 5, 4);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL ("PARAM_NAME"
VARCHAR(256), "INT_VALUE" INTEGER, "DOUBLE_VALUE" DOUBLE, "STRING_VALUE"
VARCHAR(1000));
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FACTOR_NUMBER', 2, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('METHOD', 0, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('ROTATION', 2, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SCORE', 1, NULL, NULL);
CALL _SYS_AFL.PAL_FACTOR_ANALYSIS (PAL_FACTOR_ANALYSIS_DATA_TBL,
#PAL_PARAMETER_TBL, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?);
```

**Expected Result**

EIGENVALUES:

| | FACTOR_ID | EIGENVALUE | VAR_PROP | CUM_VAR_PROP |
|---|---|---|---|---|
| 1 | FACTOR_1 | 3.696030774653954 | 0.6160051291089922 | 0.6160051291089922 |
| 2 | FACTOR_2 | 1.0731144798253218 | 0.17885241330422028 | 0.7948575424132125 |
| 3 | FACTOR_3 | 1.0007740939363208 | 0.1667956823227201 | 0.9616532247359326 |
| 4 | FACTOR_4 | 0.16100348082835... | 0.0268339134713932 | 0.9884871382073258 |
| 5 | FACTOR_5 | 0.04096115935357... | 0.006826859892263... | 0.9953139980995891 |
| 6 | FACTOR_6 | 0.02811601140246... | 0.004686001900410... | 1 |

VARIANCE_EXPLANATION:

| | FACTOR_ID | VAR | VAR_PROP | CUM_VAR_PROP | ROT_VAR | ROT_VAR_PROP | ROT_CUM_VAR_PROP |
|---|---|---|---|---|---|---|---|
| 1 | FACTOR_1 | 3.696030774653954 | 0.6160051291089922 | 0.6160051291089922 | 3.2785366400710743 | ? | ? |
| 2 | FACTOR_2 | 1.0731144798253223 | 0.17885241330422036 | 0.7948575424132125 | 3.017653052173039 | ? | ? |

COMMUNALITIES:

| | NAME | X1 | X2 | X3 | X4 | X5 | X6 |
|---|---|---|---|---|---|---|---|
| 1 | COMMUNALITIES | 0.9405221775107266 | 0.9089737086968994 | 0.6375374175418147 | 0.5815488822758216 | 0.8621429881660707 | 0.8384200802879433 |

LOADINGS:

| | FACTOR_ID | LOADINGS_X1 | LOADINGS_X2 | LOADINGS_X3 | LOADINGS_X4 | LOADINGS_X5 | LOADINGS_X6 |
|---|---|---|---|---|---|---|---|
| 1 | FACTOR_1 | 0.7987985386143936 | 0.7703593833516583 | 0.7947521836188668 | 0.7575727801606499 | 0.8087763527340572 | 0.7777090397407108 |
| 2 | FACTOR_2 | 0.549948245036054 | 0.5617117847962174 | -0.07685300368142912 | -0.08736340787472313 | -0.45609626113822344 | -0.4833101796915982 |

ROTATED_LOADINGS:

| | FACTOR_ID | ROT_LOADINGS_X1 | ROT_LOADINGS_X2 | ROT_LOADINGS_X3 | ROT_LOADINGS_X4 | ROT_LOADINGS_X5 | ROT_LOADINGS_X6 |
|---|---|---|---|---|---|---|---|
| 1 | FACTOR_1 | -0.08307847100356597 | -0.11344423886800542 | 0.5807888046113657 | 0.5686158486782273 | 0.992802844774049 | 1.002227085116613 |
| 2 | FACTOR_2 | 1.0162052583004346 | 1.0155145102828198 | 0.30450672125242334 | 0.2742901912227176 | -0.11798833596026007 | -0.1641066269323464 |

STRUCTURE:

| | FACTOR_ID | STRUCTURE_X1 | STRUCTURE_X2 | STRUCTURE_X3 | STRUCTURE_X4 | STRUCTURE_X5 | STRUCTURE_X6 |
|---|---|---|---|---|---|---|---|
| 1 | FACTOR_1 | 0.5130626268914917 | 0.48229164238580174 | 0.7594229630563283 | 0.7295239471177395 | 0.9235868123203324 | 0.9059564705412448 |
| 2 | FACTOR_2 | 0.9674685581987842 | 0.9489642021081224 | 0.645217490851508 | 0.607859884493634 | 0.4644240979871751 | 0.42383439185437527 |

ROTATION:

| | ROTATION | ROTATION_1 | ROTATION_2 | ROTATION_3 | ROTATION_4 | ROTATION_5 | ROTATION_6 |
|---|---|---|---|---|---|---|---|
| 1 | ROTATION_1 | 0.6279687411021506 | 0.49263745334125136 | ? | ? | ? | ? |
| 2 | ROTATION_2 | -1.0631891072825448 | 1.1322650560761474 | ? | ? | ? | ? |

FACTOR_CORRELATION:

| | FACTOR_ID | FACTOR_1 | FACTOR_2 | FACTOR_3 | FACTOR_4 | FACTOR_5 | FACTOR_6 |
|---|---|---|---|---|---|---|---|
| 1 | FACTOR_1 | 1 | 0.5866345337477209 | ? | ? | ? | ? |
| 2 | FACTOR_2 | 0.5866345337477209 | 1 | ? | ? | ? | ? |

SCORE_MODEL:

| | NAME | X1 | X2 | X3 | X4 | X5 | X6 |
|---|---|---|---|---|---|---|---|
| 1 | FACTOR_1 | -0.0062822288906809465 | -0.01771129993750234 | 0.22574653932261313 | 0.2204300800548703 | 0.3702212150591374 | 0.37263118046314 |
| 2 | FACTOR_2 | 0.44671469466731883 | 0.44566440258856954 | 0.1487235525377457 | 0.13508125155124953 | -0.027737236622774664 | -0.047871660349739636 |
| 3 | _MEAN | 2.2222222222222228 | 2.4444444444444446 | 2.2222222222222223 | 2.388888888888889 | 2.2222222222222223 | 2.388888888888889 |
| 4 | _SD | 1.5924659219017219 | 1.5424282697487952 | 1.5924659219017219 | 1.5769997160986313 | 1.5924659219017219 | 1.5769997160986313 |

SCORES:

| | ID | FACTOR_1 | FACTOR_2 | FACTOR_3 | FACTOR_4 | FACTOR_5 | FACTOR_6 |
|---|---|---|---|---|---|---|---|
| 1 | 1 | -0.39524324522751914 | -0.5717741144841599 | ? | ? | ? | ? |
| 2 | 2 | -0.4067259833780503 | -0.28283724051130926 | ? | ? | ? | ? |
| 3 | 3 | -0.25546510969291275 | -0.48611699598737945 | ? | ? | ? | ? |
| 4 | 4 | -0.15895202928171093 | -0.6021302774314623 | ? | ? | ? | ? |
| 5 | 5 | -0.39524324522751914 | -0.5717741144841599 | ? | ? | ? | ? |
| 6 | 6 | -0.02076933880564577 | -1.0254202256886762 | ? | ? | ? | ? |
| 7 | 7 | -0.03225207695617696 | -0.7364833517158256 | ? | ? | ? | ? |
| 8 | 8 | 0.11900879672896061 | -0.9397631071918957 | ? | ? | ? | ? |
| 9 | 9 | 0.20403913898963125 | -0.7668395146631279 | ? | ? | ? | ? |
| 10 | 10 | -0.02076933880564577 | -1.0254202256886762 | ? | ? | ? | ? |
| 11 | 11 | -0.9891731351798415 | 0.20903659774167851 | ? | ? | ? | ? |
| 12 | 12 | -1.0006558733303725 | 0.497973471714529 | ? | ? | ? | ? |
| 13 | 13 | -0.8493949996452351 | 0.29469371623845886 | ? | ? | ? | ? |
| 14 | 14 | -0.7528819192340332 | 0.17868043479437612 | ? | ? | ? | ? |
| 15 | 15 | -0.9891731351798415 | 0.20903659774167851 | ? | ? | ? | ? |
| 16 | 16 | 2.465419062929069 | 1.2558177048234718 | ? | ? | ? | ? |
| 17 | 17 | 1.7352704849894467 | 2.1716659143639836 | ? | ? | ? | ? |
| 18 | 18 | 1.7429619463073975 | 2.1916547304284926 | ? | ? | ? | ? |

STATISTICS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| | | |

## 5.9.12 Grubbs' Test

Grubbs' test is used to detect outliers from a given univariate data set $Y=\{Y_1, Y_2, ..., Y_n\}$. The algorithm assumes that Y comes from Gaussian distribution.

The basic steps of the algorithm are as follows:

1. Define the hypothesis.
   H0: There are no outliers in the data set Y.
   H1: There is at least one outlier in data set Y.
2. Calculate Grubbs' test statistic.

$$G = \frac{\max_{i=1,..,n} |Y_i - \overline{Y}|}{s}$$

   Here $\overline{Y} = \frac{1}{n}\sum_{i=1}^{n} Y_i, s = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(Y_i - \overline{Y})^2}$
3. Given the significance level α, if

$$G > T = \frac{n-1}{\sqrt{n}}\sqrt{\frac{(t_{\frac{\alpha}{2n},n-2})^2}{n-2+(t_{\frac{\alpha}{2n},n-2})^2}}$$

   The algorithm will reject the hypothesis at the significance level α, which means that the data set contains outlier. Here $t_{\frac{\alpha}{2n},n-2}$ denotes the quantile value of t-distribution with n-2 degrees and a significance level $\frac{\alpha}{2n}$.

The above is called two-sided test. There is another version called one-sided test for minimum value or maximum value.

- For minimum value: $G = \frac{\overline{Y} - Y_{min}}{s}$
- For maximum value: $G = \frac{Y_{max} - \overline{Y}}{s}$

Note that you must replace $\frac{\alpha}{2n}$ with $\frac{\alpha}{n}$ for one-sided test.

Suppose $Y_{max}$ is an outlier from Grubbs' test, you can calculate the statistic value U as shown below:

1. Remove $Y_{max}$ from original data and get $Z=\{Z_1, Z_2, ..., Z_{n-1}$.
2. Calculate $$U = \frac{\sum_{i=1}^{n-1}(Z_i - \bar{Z})^2}{\sum_{i=1}^{n}(Y_i - \bar{Y})^2}$$

PAL also supports the repeat version of two-sided test. The steps are as follows:

1. Perform two-sided test for data Y.
2. If there is outlier in Y, remove it from Y and go to Step 1; Otherwise go to Step 3.
3. Return all the outliers.

## Prerequisites

- The length of the input data must not be less than 5.
- The input data does not contain null value.

## _SYS_AFL.PAL_GRUBBS_TEST

This function performs Grubbs' test for identifying outliers from input data.

### Overview of all Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<OUTLIER table>` | OUT |
| 4 | `<STATISTICS table>` | OUT |

### Input Table

| Table | Column | Reference for Output Table | Data Type | Description |
|---|---|---|---|---|
| DATA | 1st column | SOURCE_ID | INTEGER, VARCHAR, or NVARCHAR | ID |

| Table | Column | Reference for Output Table | Data Type | Description |
|---|---|---|---|---|
| | 2nd column | RAW_DATA | INTEGER, DOUBLE, or DECIMAL(p, s) | Raw data |

### Parameter Table

Mandatory Parameter

None.

Optional Parameter

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| METHOD | INTEGER | 1 | • 1: Two-sided test<br>• 2: One-sided test for minimum value<br>• 3: One-sided test for maximum value<br>• 4: Repeat two-sided test |
| ALPHA | DOUBLE | 0.05 | Significance level |

### Output Tables

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| OUTLIER | 1st column | SOURCE_ID (in input table) data type | SOURCE_ID (in input table) column name | ID of outlier data |
| | 2nd column | RAW_DATA (in input table) data type | RAW_DATA (in input table) column name | Value of original data |
| STATISTICS | 1st column | SOURCE_ID (in input table) data type | SOURCE_ID (in input table) column name | ID of outlier data |
| | 2nd column | NVARCHAR | STAT_NAME | Statistic name |
| | 3rd column | DOUBLE | STAT_VALUE | Statistic value |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA DM_PAL;
```

```
DROP TABLE PAL_GRUBBS_DATA_TBL;
CREATE COLUMN TABLE PAL_GRUBBS_DATA_TBL (
    "ID" INTEGER,
    "VAL" DOUBLE
);
INSERT INTO PAL_GRUBBS_DATA_TBL VALUES (100, 4.254843);
INSERT INTO PAL_GRUBBS_DATA_TBL VALUES (200, 0.135000);
INSERT INTO PAL_GRUBBS_DATA_TBL VALUES (300, 11.072257);
INSERT INTO PAL_GRUBBS_DATA_TBL VALUES (400, 14.797838);
INSERT INTO PAL_GRUBBS_DATA_TBL VALUES (500, 12.125133);
INSERT INTO PAL_GRUBBS_DATA_TBL VALUES (600, 14.265839);
INSERT INTO PAL_GRUBBS_DATA_TBL VALUES (700, 7.731352);
INSERT INTO PAL_GRUBBS_DATA_TBL VALUES (800, 6.856739);
INSERT INTO PAL_GRUBBS_DATA_TBL VALUES (900, 15.094403);
INSERT INTO PAL_GRUBBS_DATA_TBL VALUES (101,  8.149382);
INSERT INTO PAL_GRUBBS_DATA_TBL VALUES (201, 9.160144);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('ALPHA',  null,  0.2, null);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('METHOD',    2, null, null);
CALL "_SYS_AFL"."PAL_GRUBBS_TEST"(PAL_GRUBBS_DATA_TBL, #PAL_PARAMETER_TBL, ?, ?);
```

**Expected Result**

OUTLIER:

| | ID | VAL |
|---|---|---|
| 1 | 200 | 0.135 |

STATISTICS:

| | ID | STAT_NAME | STAT_VALUE |
|---|---|---|---|
| 1 | 200 | MEAN | 9.422084545454545 |
| 2 | 200 | STANDARD_SAMPLE_VARIANCE | 4.675935228561435 |
| 3 | 200 | T | 1.9102192033966785 |
| 4 | 200 | G | 1.9861448226928808 |
| 5 | 200 | U | 0.5660751617619288 |

> **Note**
>
> The wrapper procedure generation approach to call PAL functions is still supported. For more information, refer to the older versions (prior to SAP HANA Platform 2.0 SPS 02) of this document.

## 5.9.13 Kaplan-Meier Survival Analysis

The Kaplan-Meier estimator is a non-parametric statistic used to estimate the survival function from lifetime data. It is often used to measure the time-to-death of patients after treatment or time-to-failure of machine parts.

Sometimes subjects under study are refused to remain in the study or some of the subjects may not experience the event before the end of the study, or you lose touch with them midway in the study. These situations are labeled as censored observations.

The Kaplan-Meier estimator of survival function is calculated as:

$$\hat{S}(t) = \prod_{t_i \le t} \frac{n_i - d_i}{n_i}$$

Where $n_i$ is the number of subjects at risk and $d_i$ is the number of subjects who fail, both at time $t_i$.

The Kaplan-Meier estimator $\hat{S}(t)$ can be regarded as a point estimate of the survival function `S(t)` at any time t. We can construct 95% confidence intervals around each of these estimates. To compute the confidence intervals, Greenwood's Formula gives an asymptotic estimate of the variance of $\hat{S}(t)$ for large groups:

$$\widehat{Var}[\hat{S}(t)] = \hat{S}(t)^2 \sum_{t_i \le t} \frac{d_i}{n_i(n_i - d_i)}$$

So the Greenwood's confidence interval is:

$$\hat{S}(t) \pm z_{\alpha/2} \sqrt{\widehat{Var}[\hat{S}(t)]}$$

Where $Z_{\alpha/2}$ is the $\alpha/2$–th quantile of the normal distribution.

However the endpoints of Greenwood's confidence interval can be negative or greater than one. Here we use another confidence interval based on the large sample normal distribution of `log(-log(`$\hat{S}(t)$`))` with:

$$\widehat{Var}[\log(-\log(\hat{S}(t)))] = \frac{1}{(\log(\hat{S}(t)))^2} \sum_{t_i \le t} \frac{d_i}{n_i(n_i - d_i)}$$

So we get:

$$\texttt{ci_lower} = \log(-\log(\hat{S}(t))) + z_{\alpha/2} \sqrt{\widehat{Var}[\log(-\log(\hat{S}(t)))]}$$

$$\texttt{ci_upper} = \log(-\log(\hat{S}(t))) - z_{\alpha/2} \sqrt{\widehat{Var}[\log(-\log(\hat{S}(t)))]}$$

Transform endpoints (`ci_lower`, `ci_upper`) back to obtain confidence interval:

```
(exp(-exp(ci_lower)), exp(-exp(ci_upper)))
```

Equality comparison of two or more Kaplan-Meier survival functions can be done using a statistical hypothesis test called the log rank test by weighting all time points the same. It is used to test the null hypothesis where there is no difference between the population survival functions.

For `i=1, 2, ... , g` and `j=1, 2, … , k`, where `g` is number of groups and `k` is number of distinct failure times.

$n_{ij}$ = number at risk in $i_{th}$ group at $j_{th}$ ordered failure time

$o_{ij}$ = observed number of failures in $i_{th}$ group at $j_{th}$ ordered failure time

$e_{ij}$ = expected number of failures in $i_{th}$ group at $j_{th}$ ordered failure time = $= \frac{n_{ij}}{\sum_{i=1}^{g} n_{ij}} \left( \sum_{i=1}^{g} o_{ij} \right)$

$$n_j = \sum_{i=1}^{g} n_{ij}$$

$$o_j = \sum_{i=1}^{g} o_{ij}$$

$$O_i - E_i = \sum_{j=1}^{k} o_{ij} - e_{ij}$$

$$\widehat{Var}(O_i - E_i) = \sum_{j=1}^{k} \frac{n_{ij}(n_j - n_{ij}) o_j (n_j - o_j)}{n_j^2 (n_j - 1)}$$

$$\widehat{Cov}(O_i - E_i, O_p - E_p) = \sum_{j=1}^{k} \frac{-n_{ij} n_{pj} o_j (n_j - o_j)}{n_j^2 (n_j - 1)}$$

$$d = (O_1 - E_1, \quad O_2 - E_2, \ \ldots, \quad O_{g-1} - E_{g-1})$$

$V = ((v_{ip}))'$ where $v_{ii} = \widehat{Var}(O_i - E_i)$ and $v_{ip} = \widehat{Cov}(O_i - E_i, O_p - E_p)$ for $i=1, 2, \ldots, g-1$; $p = 1, 2, \ldots, g-1$.

The, the log rank statistics is given by the matrix product formula:

$$d' V^{-1} d$$

Which has approximately a chi-squared distribution with `g-1` degrees of freedom under the null hypothesis that all `g` groups have a common survival function.

Comparison of two Kaplan-Meier survival functions can be simplified as:

$$\frac{(O_1 - E_1)^2}{\widehat{Var}(O_1 - E_1)}$$

Where $\widehat{Var}(O_i - E_i) = \sum_{j=1}^{k} \frac{n_{1j} n_{2j} (o_{1j} + o_{2j}) (n_{1j} + n_{2j} - o_{1j} - o_{2j})}{(n_{1j} + n_{2j})^2 (n_{1j} + n2_j - 1)}$

## Prerequisite

No missing or null data in the inputs.

# _SYS_AFL.PAL_KMSURV

This function estimates the probability of surviving time `t` (`t` is an event time) using Kaplan-Meier estimator and compares several groups of survival functions using log rank test. This function does log rank test if the lifetime data comes from multiple groups, otherwise it skips doing it.

### Overview of all Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<SURVIVAL ESTIMATES table>` | OUT |
| 4 | `<LOG RANK TEST STATISTICS 1 table>` | OUT |
| 5 | `<LOG RANK TEST STATISTICS 2 table>` | OUT |

### Input Table

| Table | Column | Data Type | Description |
|---|---|---|---|
| DATA | 1st column | INTEGER, DOUBLE, or DECIMAL(p, s) | Follow-up time |
| | 2nd column | INTEGER | Status indicator |
| | 3rd column | INTEGER | Occurrence number of events or censoring at the follow-up time. It also allows for multiple rows for one follow-up time. |
| | 4th column | INTEGER or VARCHAR | Group |

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| EVENT_INDICATOR | INTEGER | 1 | Specifies one value to indicate an event has occurred. |
| CONF_LEVEL | DOUBLE | 0.95 | Specifies confidence level for a two-sided confidence interval on the survival estimate. |

### Output Tables

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| SURVIVAL ESTIMATES | 1st column | VARCHAR | GROUP | Group |
| | 2nd column | DOUBLE or DECIMAL(p, s) | TIME | Event occurrence time. Survival estimates at all event times are output. |
| | 3rd column | INTEGER | RISK_NUMBER | Number at risk (total number of survivors at the beginning of each period) |
| | 4th column | INTEGER | EVENT_NUMBER | Number of event occurrences |
| | 5th column | DOUBLE | PROBABILITY | Probability of surviving beyond event occurrence time |
| | 6th column | DOUBLE | SE | Standard error for the survivor estimate |
| | 7th column | DOUBLE | CI_LOWER | Lower bound of confidence interval |
| | 8th column | DOUBLE | CI_UPPER | Upper bound of confidence interval |
| LOG RANK TEST STATISTICS 1 | 1st column | NVARCHAR | GROUP | Group |
| | 2nd column | INTEGER | TOTAL_RISK | All individuals in the lifetime study |
| | 3rd column | INTEGER | OBSERVED | Observed event number |

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| | 4th column | DOUBLE | EXPECTED | Expected event number |
| | 5th column | DOUBLE | LOGRANK_STAT | Log rank test statistics $\left(\frac{(O_1-E_1)^2}{\widehat{Var}(O_1-E_1)}\right)$ |
| LOG RANK TEST STATISTICS 2 | 1st column | NVARCHAR | STAT_NAME | Statistics name<br>Examples: Chisq, df, p-value |
| | 2nd column | DOUBLE | STAT_VALUE | Statistics value |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_KMSURV_DATA_TBL;
CREATE COLUMN TABLE PAL_KMSURV_DATA_TBL (
    "TIME" INTEGER,
    "STATUS" INTEGER,
    "OCCURRENCES" INTEGER,
    "GROUP" INTEGER
);
INSERT INTO PAL_KMSURV_DATA_TBL VALUES(9,   1, 1, 2);
INSERT INTO PAL_KMSURV_DATA_TBL VALUES(10,  1, 1, 1);
INSERT INTO PAL_KMSURV_DATA_TBL VALUES(1,   1, 2, 0);
INSERT INTO PAL_KMSURV_DATA_TBL VALUES(31,  0, 1, 1);
INSERT INTO PAL_KMSURV_DATA_TBL VALUES(2,   1, 1, 0);
INSERT INTO PAL_KMSURV_DATA_TBL VALUES(25,  1, 3, 1);
INSERT INTO PAL_KMSURV_DATA_TBL VALUES(255, 0, 1, 0);
INSERT INTO PAL_KMSURV_DATA_TBL VALUES(90,  1, 1, 0);
INSERT INTO PAL_KMSURV_DATA_TBL VALUES(22,  1, 1, 1);
INSERT INTO PAL_KMSURV_DATA_TBL VALUES(100, 0, 1, 1);
INSERT INTO PAL_KMSURV_DATA_TBL VALUES(28,  0, 1, 0);
INSERT INTO PAL_KMSURV_DATA_TBL VALUES(5,   1, 1, 1);
INSERT INTO PAL_KMSURV_DATA_TBL VALUES(7,   1, 1, 1);
INSERT INTO PAL_KMSURV_DATA_TBL VALUES(11,  0, 1, 0);
INSERT INTO PAL_KMSURV_DATA_TBL VALUES(20,  0, 1, 0);
INSERT INTO PAL_KMSURV_DATA_TBL VALUES(30,  1, 2, 2);
INSERT INTO PAL_KMSURV_DATA_TBL VALUES(101, 0, 1, 2);
INSERT INTO PAL_KMSURV_DATA_TBL VALUES(8,   0, 1, 1);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
CALL "_SYS_AFL"."PAL_KMSURV"(PAL_KMSURV_DATA_TBL, #PAL_PARAMETER_TBL, ?, ?, ?);
```

**Expected Result**

SURVIVAL ESTIMATES:

| | GROUP | TIME | RISK_NUMBER | EVENT_NUMBER | PROBABILITY | SE | CI_LOWER | CI_UPPER |
|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 8 | 2 | 0.75 | 0.15309310892394862 | 0.3148071137055191 | 0.9308983170906274 |
| 2 | 0 | 2 | 6 | 1 | 0.625 | 0.17116329922036438 | 0.22933332242913176 | 0.8606982769135386 |
| 3 | 0 | 90 | 2 | 1 | 0.3125 | 0.23696485762661093 | 0.015390059783199... | 0.7231578986482249 |
| 4 | 1 | 5 | 10 | 1 | 0.9 | 0.09486832980505137 | 0.4730092713620503 | 0.9852813933673431 |
| 5 | 1 | 7 | 9 | 1 | 0.7999999999999999 | 0.12649110640673517 | 0.4086907816334044 | 0.9458726458389116 |
| 6 | 1 | 10 | 7 | 1 | 0.6857142857142857 | 0.1514940174323206 | 0.30458507420184733 | 0.8871484950277075 |
| 7 | 1 | 22 | 6 | 1 | 0.5714285714285715 | 0.163774715491746 | 0.21719270238228391 | 0.8145721379207795 |
| 8 | 1 | 25 | 5 | 3 | 0.22857142857142862 | 0.14129760994885754 | 0.034939475461901... | 0.5223399669753022 |
| 9 | 2 | 9 | 4 | 1 | 0.75 | 0.21650635094610965 | 0.1279469175951459 | 0.9605486422850784 |
| 10 | 2 | 30 | 3 | 2 | 0.25 | 0.21650635094610965 | 0.008947824416603... | 0.6653253427888653 |

LOG RANK TEST STATISTICS 1:

| | GROUP | TOTAL_RISK | OBSERVED | EXPECTED | LOGRANK_STAT |
|---|---|---|---|---|---|
| 1 | 0 | 8 | 4 | 4.353652122731 07 | 0.0457121614698249 44 |
| 2 | 1 | 10 | 7 | 6.0246383002961945 | 0.30795141540683535 |
| 3 | 2 | 4 | 3 | 3.621709576972735 | 0.1617863617896038 |

LOG RANK TEST STATISTICS 2:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | chiSqr | 0.32797068374749033 |
| 2 | df | 2 |
| 3 | p-value | 0.8487544629391057 |

> **Note**
>
> The wrapper procedure generation approach to call PAL functions is still supported. For more information, refer to the older versions (prior to SAP HANA Platform 2.0 SPS 02) of this document.

## 5.9.14 Kernel Density

Kernel Density Estimation (KDE) is quite popular in unsupervised learning, feature engineering, and data modeling. Its concept is analogue with histograms whereas getting rid of its defects.

Mathematically, a kernel is a positive function `K(x; h)`, which is controlled by the parameter bandwidth `h`. Given this kernel form, the density estimates at a point `y` within a group of points $x_i$`;i = 1 ... N` is given by:

$$\rho K(y) = \sum_{i=1}^{N} K((y - x_i)/h)$$

Here bandwidth acts as a smooth factor which controls the tradeoff between bias and variance in the result. Generally, a small bandwidth often leads to unsmooth density distribution, a big bandwidth could provide a quite smooth density distribution with higher bias.

In this release, PAL supports six different kernel functions. The form of these kernels is as follows:

- Gaussian kernel (kernel = 'gaussian')

  $K(x;h) \propto \exp\left(-\frac{|x|^2}{2h^2}\right)$

- Tophat kernel (kernel = 'tophat')

  $K(x;h) \propto 1 \text{ if } |x| < h$

- Epanechnikov kernel (kernel = 'epanechnikov')

  $K(x;h) \propto 1 - \frac{|x|^2}{h^2} \text{ if } |x| < h$

- Exponential kernel (kernel = 'exponential')

  $K(x;h) \propto \exp\left(-\frac{x}{h}\right)$

- Linear kernel (kernel = 'linear')

  $K(x;h) \propto 1 - \frac{|x|}{h} \text{ if } |x| < h$

- Cosine kernel (kernel = 'cosine')

  $K(x;h) \propto \cos\left(\frac{\pi|x|}{2h}\right) \text{ if } |x| < h$

All of them are valid no matter the sample space is one-dimensional or not. However, in multidimensional case, we only consider those kernel densities that are isotropic, which is the basis to do following calculation. All features should be real-valued. To speed up the calculation process, data structure based on KD-tree and Ball-tree are provided.

## Prerequisites

- The first column of the training data and input data is an ID column. Other data columns are of integer or double type.
- The input data does not contain any null value.

## _SYS_AFL.PAL_KDE

This is a statistics function using KDE algorithm.

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<TRAINING DATA table>` | IN |
| | `<PREDICT DATA tabl>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<RESULT table>` | OUT |

| Position | Table Name | Parameter Type |
|---|---|---|
| 4 | `<STATISTIC table>` | OUT |

### Input Table

| Table | Column | Reference | Column Data Type | Description |
|---|---|---|---|---|
| FITTING DATA | 1st column | ID | INTEGER, BIGINT, VARCHAR, or NVARCHAR | ID |
| | Feature columns | None | INTEGER, DOUBLE, or DECIMAL(p, s) | Feature data |

## Parameter table

Mandatory table

None.

Optional parameters

The following parameters are optional. If a parameter is not specified, PAL uses its default value.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| THREAD_RATIO | DOUBLE | 0.0 | Specifies the ratio of total number of threads that can be used by this function. The value range is from 0 to 1, where 0 means only using 1 thread, and 1 means using at most all the currently available threads. Values outside the range will be ignored and this function heuristically determines the number of threads to use. | |
| BUCKET_SIZE | INTEGER | 30 | Number of samples in a KD tree or Ball tree leaf node. | Only Valid when Method is 1 or 2. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| KERNEL | INTEGER | 0 | Kernel function type:<br>• 0: Gaussian<br>• 1: Tophat<br>• 2: Epanechnikov<br>• 3: Exponential<br>• 4: Linear<br>• 5: Cosine | |
| METHOD | INTEGER | 0 | Searching method.<br>• 0: Brute force searching<br>• 1: KD-tree searching<br>• 2: Ball-tree searching | |
| BANDWIDTH | DOUBLE | 0 | Bandwidth used during density calculation,<br>0 means providing by optimizer inside. Otherwise bandwidth is provided by end users. | Only valid when data is one dimension. |
| DISTANCE_LEVEL | INTEGER | 2 | Computes the distance between the train data and the test data point.<br>• 1: Manhattan distance<br>• 2: Euclidean distance<br>• 3: Minkowski distance<br>• 4: Chebyshev distance | |
| MINKOWSKI_POWER | DOUBLE | 3.0 | When you use the Minkowski distance, this parameter controls the value of power. | Only valid when DISTANCE_LEVEL is 3. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| ABSOLUTE_RESULT_TOLERANCE | DOUBLE | 0 | The desired absolute tolerance of the result. A larger tolerance generally leads to faster execution. | It can assist to speed up the process if some precise could be tolerated. |
| RELATIVE_RESULT_TOLERANCE | DOUBLE | 1E-8 | The desired relative tolerance of the result. A larger tolerance generally leads to faster execution. | |
| STAT_INFO | INTEGER | 0 | • 0: STATISTIC table is empty<br>• 1: Statistic information is displayed in the STATISTIC table. | |

**Output table**

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| RESULT | 1st column | ID data type | ID | ID |
| | 2nd column | DOUBLE | DENSITY_VALUE | Log Density Value |
| STATISTICS | 1st column | ID data type | TEST_ID | Test ID |
| | 2nd column | NVARCHAR | FITTING_IDS | Fitting IDs |

## Example

Assume that:

- `DM_PAL` is a schema belonging to USER1; and
- USER1 has been assigned the AFL__SYS_AFL_AFLPAL_EXECUTE or AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION role.

Multidimensional example with bandwidth offered:

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_KDE_DATA_TBL;
CREATE COLUMN TABLE PAL_KDE_DATA_TBL (
    "ID" INTEGER,
    "X1" DOUBLE,
    "X2" DOUBLE
);
```

```
INSERT INTO PAL_KDE_DATA_TBL VALUES (0, -0.42576979,-1.39613035);
INSERT INTO PAL_KDE_DATA_TBL VALUES (1, 0.88410039, 1.3814935);
INSERT INTO PAL_KDE_DATA_TBL VALUES (2, 0.13412623,-0.03222389);
INSERT INTO PAL_KDE_DATA_TBL VALUES (3, 0.84550359, 2.86792078);
INSERT INTO PAL_KDE_DATA_TBL VALUES (4, 0.28844078, 1.51333705);
INSERT INTO PAL_KDE_DATA_TBL VALUES (5, -0.66678474, 1.24498042);
INSERT INTO PAL_KDE_DATA_TBL VALUES (6, -2.10296835,-1.42832694);
INSERT INTO PAL_KDE_DATA_TBL VALUES (7, 0.76990237,-0.47300711);
INSERT INTO PAL_KDE_DATA_TBL VALUES (8, 0.21029135, 0.32843074);
INSERT INTO PAL_KDE_DATA_TBL VALUES (9, 0.48232251,-0.43796174);
DROP TABLE PAL_KDE_CLASSDATA_TBL;
CREATE COLUMN TABLE PAL_KDE_CLASSDATA_TBL(
    "ID" INTEGER,
    "X1" DOUBLE,
    "X2" DOUBLE
);
INSERT INTO PAL_KDE_CLASSDATA_TBL VALUES (0, -2.10296835,-1.42832694);
INSERT INTO PAL_KDE_CLASSDATA_TBL VALUES (1, -2.10296835, 0.71979692);
INSERT INTO PAL_KDE_CLASSDATA_TBL VALUES (2, -2.10296835, 2.86792078);
INSERT INTO PAL_KDE_CLASSDATA_TBL VALUES (3, -0.60943398,-1.42832694);
INSERT INTO PAL_KDE_CLASSDATA_TBL VALUES (4, -0.60943398, 0.71979692);
INSERT INTO PAL_KDE_CLASSDATA_TBL VALUES (5, -0.60943398, 2.86792078);
INSERT INTO PAL_KDE_CLASSDATA_TBL VALUES (6, 0.88410039,-1.42832694);
INSERT INTO PAL_KDE_CLASSDATA_TBL VALUES (7, 0.88410039, 0.71979692);
INSERT INTO PAL_KDE_CLASSDATA_TBL VALUES (8, 0.88410039, 2.86792078);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 0, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('BUCKET_SIZE', 10, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('METHOD', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('BANDWIDTH', NULL, 0.68129, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('DISTANCE_LEVEL', 2, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('KERNEL', 0, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('STAT_INFO', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('"ABSOLUTE_RESULT_TOLERANCE"', NULL, 0,
NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('RELATIVE_RESULT_TOLERANCE', NULL, 1E-8,
NULL);
CALL _SYS_AFL.PAL_KDE (PAL_KDE_DATA_TBL, PAL_KDE_CLASSDATA_TBL,
"#PAL_PARAMETER_TBL", ?, ?);
```

### Expected result

RESULT:

| | ID | DENSITY_VALUE |
|---|---|---|
| 1 | 0 | -3.324821165744467 |
| 2 | 1 | -5.733966077823183 |
| 3 | 2 | -8.372878214957254 |
| 4 | 3 | -3.1232230820483... |
| 5 | 4 | -2.772520315925144 |
| 6 | 5 | -4.8528171980834... |
| 7 | 6 | -3.4697820187624... |
| 8 | 7 | -2.556680275298552 |
| 9 | 8 | -3.198531045639788 |

STATISTIC:

| | TEST_ID | FITTING_IDS |
|---|---|---|
| 1 | 0 | {"fitting ids":[0,1,2,3,4,5,6,7,8,9]} |
| 2 | 1 | {"fitting ids":[0,1,2,3,4,5,6,7,8,9]} |
| 3 | 2 | {"fitting ids":[0,1,2,3,4,5,6,7,8,9]} |
| 4 | 3 | {"fitting ids":[0,1,2,3,4,5,6,7,8,9]} |
| 5 | 4 | {"fitting ids":[0,1,2,3,4,5,6,7,8,9]} |
| 6 | 5 | {"fitting ids":[0,1,2,3,4,5,6,7,8,9]} |
| 7 | 6 | {"fitting ids":[0,1,2,3,4,5,6,7,8,9]} |
| 8 | 7 | {"fitting ids":[0,1,2,3,4,5,6,7,8,9]} |
| 9 | 8 | {"fitting ids":[0,1,2,3,4,5,6,7,8,9]} |

## KDE model evaluation and parameter selection

A function specific for model evaluation and parameter selection of KDE algorithm.

### Prerequisites

- The input data does not contain any null value.

### _SYS_AFL.PAL_KDE_CV

This is a cross validation function using KDE algorithm. With provided bandwidth candidates, we use leave-one-out strategy to evaluate each possible parameter, and then give the optimal bandwidth according maximum likelihood criteria.

#### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<STATISTIC table>` | OUT |
| 4 | `<OPTIMAL_PARAM table>` | OUT |

#### Input Table

| Table | Column | Data Type | Description |
|---|---|---|---|
| DATA | 1st columns | INTEGER, BIGINT, VARCHAR, or NVARCHAR | Record ID. If this column is absent, the `HAS_ID` parameter must be set to 0. |

| Table | Column | Data Type | Description |
|---|---|---|---|
| | Other columns | INTEGER, BIGINT, VARCHAR, or NVARCHAR | Feature data must be INTEGER or DOUBLE. |

### Parameter Table

Mandatory Parameters

The following parameters are mandatory and must be given a value.

| Name | Data Type | Description |
|---|---|---|
| RESAMPLING_METHOD | NVARCHAR | Specifies the resampling method for model evaluation or parameter selection, only loocv is permitted. |
| EVALUATION_METRIC | NVARCHAR | Specifies the evaluation metric for model evaluation or parameter selection, only NLL is supported. |

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use default value.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| HAS_ID | INTEGER | | Specifies if the first column is ID column. | |
| THREAD_RATIO | DOUBLE | 0.0 | Specify the ratio of total number of threads can be used by this function. The range of this parameter is from 0 to 1, where 0 means only using 1 thread, and 1 means using at most all currently available threads. Values outside this range would be ignored and this function heuristically determines the number of threads to use. | |
| BUCKET_SIZE | INTEGER | 30 | Number of samples in a KD tree or Ball tree leaf node. | Only Valid when Method is 1 or 2. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| KERNEL | INTEGER | 0 | Kernel function type:<br>• 0: Gaussian<br>• 1: Tophat<br>• 2: Epanechnikov<br>• 3: Exponential<br>• 4: Linear<br>• 5: Cosine | |
| METHOD | INTEGER | 0 | Searching method.<br>• 0: Brute force searching<br>• 1: KD-tree searching<br>• 2: Ball-tree searching | |
| BANDWIDTH | DOUBLE | 0 | Bandwidth used during density calculation,<br>0 means providing by optimizer inside. Otherwise bandwidth is provided by end users. | Only valid when data is one dimension. |
| DISTANCE_LEVEL | INTEGER | 2 | Computes the distance between the train data and the test data point.<br>• 1: Manhattan distance<br>• 2: Euclidean distance<br>• 3: Minkowski distance<br>• 4: Chebyshev distance | |
| MINKOWSKI_POWER | DOUBLE | 3.0 | When you use the Minkowski distance, this parameter controls the value of power. | Only valid when DISTANCE_LEVEL is 3. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| ABSOLUTE_RESULT_TOLERANCE | DOUBLE | 0 | The desired absolute tolerance of the result. A larger tolerance generally leads to faster execution. | It can assist to speed up the process if some precise could be tolerated. |
| RELATIVE_RESULT_TOLERANCE | DOUBLE | 1E-8 | The desired relative tolerance of the result. A larger tolerance generally leads to faster execution. | |
| FOLD_NUM | INTEGER | No default value | Specifies the fold number for a cross validation method. | Mandatory and valid only when RESAMPLING_METHOD is set to `cv` or `stratified_cv`. |
| REPEAT_TIMES | INTEGER | 1 | Specifies the number of repeat times for resampling. | |
| PARAM_SEARCH_STRATEGY | NVARCHAR | No default value | Specifies the method to activate parameter selection. Possible values include:<br>• grid<br>• random | |
| SEED | INTEGER | 0 | Specifies the seed for random generation. Use system time when 0 is specified. | |
| PROGRESS_INDICATOR_ID | NVARCHAR | No default value | Sets an ID of progress indicator for model evaluation or parameter selection. No progress indicator is active if no value is provided. | |
| BANDWIDTH _VALUES | NVARCHAR | No default value | Specifies values of parameter BANDWIDTH to be selected. | Only valid when parameter selection is enabled. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| BANDWIDTH _RANGE | NVARCHAR | No default value | Specifies range of parameter BANDWIDTH to be selected. | Only valid when parameter selection is enabled. |

### Output Table

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| STATISTICS | 1st column | VARCHAR or NVARCHAR | STAT_NAME | Statistic name |
| | 2nd column | VARCHAR or NVARCHAR | STAT_VALUE | Statistic value |
| OPTIMAL_PARAM | 1st column | NVARCHAR | PARAM_NAME | Optimal parameters selected |
| | 2nd column | INTEGER | INT_VALUE | |
| | 3rd column | DOUBLE | DOUBLE_VALUE | |
| | 4th column | NVARCHAR | STRING_VALUE | |

## Example

Assume that:

- `DM_PAL` is a schema belonging to USER1; and
- USER1 has been assigned the AFL__SYS_AFL_AFLPAL_EXECUTE or AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION role.

**Optimal bandwidth example:**

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_KDE_CV_DATA_TBL;
CREATE COLUMN TABLE PAL_KDE_CV_DATA_TBL(
    "ID" INTEGER,
    "DATA1" DOUBLE,
    "DATA2" DOUBLE
);
INSERT INTO PAL_KDE_CV_DATA_TBL VALUES (0, -0.42576979,-1.39613035);
INSERT INTO PAL_KDE_CV_DATA_TBL VALUES (1, 0.88410039, 1.3814935);
INSERT INTO PAL_KDE_CV_DATA_TBL VALUES (2, 0.13412623,-0.03222389);
INSERT INTO PAL_KDE_CV_DATA_TBL VALUES (3, 0.84550359, 2.86792078);
INSERT INTO PAL_KDE_CV_DATA_TBL VALUES (4, 0.28844078, 1.51333705);
INSERT INTO PAL_KDE_CV_DATA_TBL VALUES (5, -0.66678474, 1.24498042);
INSERT INTO PAL_KDE_CV_DATA_TBL VALUES (6, -2.10296835,-1.42832694);
INSERT INTO PAL_KDE_CV_DATA_TBL VALUES (7, 0.76990237,-0.47300711);
INSERT INTO PAL_KDE_CV_DATA_TBL VALUES (8, 0.21029135, 0.32843074);
INSERT INTO PAL_KDE_CV_DATA_TBL VALUES (9, 0.48232251,-0.43796174);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL(
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR(1000)
);
```

```
-- set KDE parameter
INSERT INTO #PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 0, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('BUCKET_SIZE', 10, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('METHOD', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('DISTANCE_LEVEL', 2, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('KERNEL', 0, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('ABSOLUTE_RESULT_TOLERANCE', NULL, 0,
NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('RELATIVE_RESULT_TOLERANCE', NULL, 1E-8,
NULL);
-- specify model evaluation settings
INSERT INTO #PAL_PARAMETER_TBL VALUES ('BANDWIDTH_VALUES', NULL, NULL,
'{0.68129, 1.0, 3.0, 5.0}');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('RESAMPLING_METHOD', NULL, NULL, 'cv');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('EVALUATION_METRIC', NULL, NULL, 'NLL');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FOLD_NUM', 10, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('REPEAT_TIMES', 2, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PARAM_SEARCH_STRATEGY', NULL, NULL,
'grid');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SEED', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PROGRESS_INDICATOR_ID', NULL, NULL,
'TEST');
CALL _SYS_AFL.PAL_KDE_CV(PAL_KDE_CV_DATA_TBL, "#PAL_PARAMETER_TBL", ?, ?);
```

**Expected Results**

STATISTIC:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | timeout | FALSE |
| 2 | TEST_1_NLL | 3.09592, 3.59385, 2.98461, 3.33744, 2.8 |
| 3 | TEST_2_NLL | 2.80079, 4.24529, 2.98461, 3.33744, 3.5 |
| 4 | TEST_NLL.... | 3.34835 |
| 5 | TEST_NLL.... | 0.59063 |
| 6 | EVAL_RESU... | {"candidates":[{"TEST_NLL":[[3.14299 |
| 7 | EVAL_RESU... | 8,4.350110760700765,4.248417639624 |

OPTIMAL PARAMETERS:

| | PARAM_NAME | INT_VALUE | DOUBLE_VALUE | STRING_VALUE |
|---|---|---|---|---|
| 1 | BANDWIDTH | ? | 1 | ? |

## 5.9.15 Kolmogorov-Smirnov Test

This function performs one-sample or two-sample Kolmogorov-Smirnov test for goodness of fit.

The Kolmogorov-Smirnov statistic is defined as: `D_n = sup_x{|F(x) - G(x)|}`.

For one-sample KS test, `F(x)` is the underlying distribution of a sample and `G(x)` is a given distribution. For two-sample KS test, `F(x)` and `G(x)` are the underlying distributions of two independent samples. Both tests are valid only for continuous distributions.

The KS value `D_n` will be used to approximate a p-value by comparing its value to a Kolmogorov-Smirnov distribution. If the KS statistic is small or the p-value is high, then we cannot reject the null hypothesis in favor of the alternative.

In PAL, when the input table has only one column, the algorithm performs one-sample KS test; when the input table has two columns, the two-sample KS test is performed.

## Prerequisites

- The input data has one or two columns of INTEGER, DOUBLE, or DECIMAL(p, s) type numbers.
- The input data does not contain null value when one-sample KS test is performed.
- The input data must have more than 2 records.

## _SYS_AFL.PAL_KS_TEST

This function performs KS test on input data.

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<STATISTIC table>` | OUT |

### Input Table

| Table | Column | Data Type | Description | Constraint |
|---|---|---|---|---|
| DATA | 1st column | INTEGER, DOUBLE, or DECIMAL(p, s) | Attribute data | |
| | 2nd column | INTEGER, DOUBLE, or DECIMAL(p, s) | Attribute data | Only valid when performing two-sample KS test. |

Parameter Table

Mandatory Parameters

| Name | Data Type | Default Value | Description | Constraint |
|---|---|---|---|---|
| DISTRIBUTION_NAME | VARCHAR or NVARCHAR | None | Specifies the distribution type:<br>• beta<br>• cauchy<br>• chi_square<br>• exponential<br>• gamma<br>• lognormal<br>• normal<br>• student_t<br>• uniform<br>• weibull | Only valid when the input data has one column. |

Optional parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description | Constraint |
|---|---|---|---|---|
| TEST_TYPE | INTEGER | 0 | The alternative hypothesis type:<br>• 0: Two sides<br>• 1: Less<br>• 2: Greater | Only valid when the input data has two columns. |
| SHAPE1 | DOUBLE | 0.5 | Distribution parameter | Only valid when DISTRIBUTION_NAME is beta. |
| SHAPE2 | DOUBLE | 0.5 | Distribution parameter | Only valid when DISTRIBUTION_NAME is beta. |
| LOCATION | DOUBLE | 0 | Distribution parameter | Only valid when DISTRIBUTION_NAME is cauchy. |
| SCALE | DOUBLE | 1 | Distribution parameter | Only valid when DISTRIBUTION_NAME is cauchy. |
| DEGREES_OF_FREEDOM | DOUBLE | 1 | Distribution parameter | Only valid when DISTRIBUTION_NAME is chi_square. |
| RATE | DOUBLE | 1 | Distribution parameter | Only valid when DISTRIBUTION_NAME is exponential. |
| SHAPE | DOUBLE | 1 | Distribution parameter | Only valid when DISTRIBUTION_NAME is gamma. |

| Name | Data Type | Default Value | Description | Constraint |
|---|---|---|---|---|
| SCALE | DOUBLE | 1 | Distribution parameter | Only valid when DISTRIBUTION_NAME is gamma. |
| LOCATION | DOUBLE | 0 | Distribution parameter | Only valid when DISTRIBUTION_NAME is lognormal. |
| SCALE | DOUBLE | 1 | Distribution parameter | Only valid when DISTRIBUTION_NAME is lognormal. |
| MEAN | DOUBLE | 0 | Distribution parameter | Only valid when DISTRIBUTION_NAME is normal. |
| SD | DOUBLE | 1 | Distribution parameter | Only valid when DISTRIBUTION_NAME is normal. |
| DEGREES_OF_FREEDOM | DOUBLE | 1 | Distribution parameter | Only valid when DISTRIBUTION_NAME is student_t. |
| MIN | DOUBLE | 0 | Distribution parameter | Only valid when DISTRIBUTION_NAME is uniform. |
| MAX | DOUBLE | 1 | Distribution parameter | Only valid when DISTRIBUTION_NAME is uniform. |
| SHAPE | DOUBLE | 1 | Distribution parameter | Only valid when DISTRIBUTION_NAME is weibull. |
| SCALE | DOUBLE | 1 | Distribution parameter | Only valid when DISTRIBUTION_NAME is weibull. |

Output Table

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| STATISTIC | 1st column | NVARCHAR(100) | STAT_NAME | Name of statistics |
| | 2nd column | DOUBLE | STAT_VALUE | Value of statistics |

### Example

Assume that:

- DM_PAL is a schema belonging to USER1;
- USER1 has been assigned the AFL__SYS_AFL_AFLPAL_EXECUTE or AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION role.

#### Example 1: One-sample test

```
SET SCHEMA DM_PAL;
```

```
DROP TABLE PAL_KSTEST_DATA1_TBL;
CREATE COLUMN TABLE PAL_KSTEST_DATA1_TBL ("X1" DOUBLE);
INSERT INTO PAL_KSTEST_DATA1_TBL VALUES (0.58);
INSERT INTO PAL_KSTEST_DATA1_TBL VALUES (0.42);
INSERT INTO PAL_KSTEST_DATA1_TBL VALUES (0.52);
INSERT INTO PAL_KSTEST_DATA1_TBL VALUES (0.33);
INSERT INTO PAL_KSTEST_DATA1_TBL VALUES (0.43);
INSERT INTO PAL_KSTEST_DATA1_TBL VALUES (0.23);
INSERT INTO PAL_KSTEST_DATA1_TBL VALUES (0.58);
INSERT INTO PAL_KSTEST_DATA1_TBL VALUES (0.76);
INSERT INTO PAL_KSTEST_DATA1_TBL VALUES (0.53);
INSERT INTO PAL_KSTEST_DATA1_TBL VALUES (0.64);
DROP TABLE PAL_KSTEST_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_KSTEST_PARAMETER_TBL (
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR (1000)
);
INSERT INTO PAL_KSTEST_PARAMETER_TBL VALUES ('DISTRIBUTION_NAME', NULL, NULL,
'uniform');
INSERT INTO PAL_KSTEST_PARAMETER_TBL VALUES ('MIN', NULL, 0, NULL);
INSERT INTO PAL_KSTEST_PARAMETER_TBL VALUES ('MAX', NULL, 1, NULL);
DROP TABLE PAL_KSTEST_STATISTIC_TBL;
CREATE COLUMN TABLE PAL_KSTEST_STATISTIC_TBL (
    "STAT_NAME" NVARCHAR(100),
    "STAT_VALUE" DOUBLE
);
DO BEGIN
  lt_data = SELECT * FROM PAL_KSTEST_DATA1_TBL;
  lt_param = SELECT * FROM PAL_KSTEST_PARAMETER_TBL;
  CALL _SYS_AFL.PAL_KS_TEST (:lt_data, :lt_param, lt_statistic);
  INSERT INTO PAL_KSTEST_STATISTIC_TBL SELECT * FROM :lt_statistic;
END;
SELECT * FROM PAL_KSTEST_STATISTIC_TBL;
```

### Expected Result

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | KS statistic | 0.26 |
| 2 | p-value | 0.4466249870841973 |

### Example 2: Two-sample test

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_KSTEST_DATA2_TBL;
CREATE COLUMN TABLE PAL_KSTEST_DATA2_TBL ("X1" DOUBLE, "X2" DOUBLE);
INSERT INTO PAL_KSTEST_DATA2_TBL VALUES (0.34235128, 0.38175431);
INSERT INTO PAL_KSTEST_DATA2_TBL VALUES (2.73083142, 1.09566376);
INSERT INTO PAL_KSTEST_DATA2_TBL VALUES (-1.26213692, 0.00211847);
INSERT INTO PAL_KSTEST_DATA2_TBL VALUES (1.41671807, 1.04676821);
INSERT INTO PAL_KSTEST_DATA2_TBL VALUES (-1.68170524, 0.42839085);
INSERT INTO PAL_KSTEST_DATA2_TBL VALUES (-0.0523474, -0.78433454);
INSERT INTO PAL_KSTEST_DATA2_TBL VALUES (-0.18968068, 1.2050457);
INSERT INTO PAL_KSTEST_DATA2_TBL VALUES (-1.09911123, -1.32433733);
INSERT INTO PAL_KSTEST_DATA2_TBL VALUES (-0.55033859, -0.74692157);
INSERT INTO PAL_KSTEST_DATA2_TBL VALUES (-0.675985, 0.43152711);
DROP TABLE PAL_KSTEST_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_KSTEST_PARAMETER_TBL (
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR (1000)
);
INSERT INTO PAL_KSTEST_PARAMETER_TBL VALUES ('TEST_TYPE', 0, NULL, NULL);
```

```
DROP TABLE PAL_KSTEST_STATISTIC_TBL;
CREATE COLUMN TABLE PAL_KSTEST_STATISTIC_TBL (
    "STAT_NAME" NVARCHAR(100),
    "STAT_VALUE" DOUBLE
);
DO BEGIN
  lt_data = SELECT * FROM PAL_KSTEST_DATA2_TBL;
  lt_param = SELECT * FROM PAL_KSTEST_PARAMETER_TBL;
  CALL _SYS_AFL.PAL_KS_TEST (:lt_data, :lt_param, lt_statistic);
  INSERT INTO PAL_KSTEST_STATISTIC_TBL SELECT * FROM :lt_statistic;
END;
SELECT * FROM PAL_KSTEST_STATISTIC_TBL;
```

**Expected Result**

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | KS statistic | 0.4 |
| 2 | p-value | 0.31285267601695577 |

## 5.9.16 Multivariate Analysis

This function calculates several basic multivariate analysis including covariance matrix and Pearson's correlation coefficient matrix. The function treats each column as a data sample.

**Covariance Matrix**

The covariance between two data samples (random variables) x and y is:

$$\mathrm{cov}(x,y)=\frac{1}{n-1}\sum_{i=1}^{n}(x_i-\bar{x})*(y_i-\bar{y})$$

Suppose that each column represents a data sample (random variable), the covariance matrix Σ is defined as the covariance between any two random variables:

$$\Sigma_{ij}=\mathrm{cov}(X_i,X_j)$$

where $X=[X_1,X_2,...,X_n]$.

**Pearson's Correlation Coefficient Matrix**

The Pearson's correlation coefficient between two data samples (random variables) X and Y is the covariance of X and Y divided by the product of standard deviation of X and the standard deviation Y:

$$\mathrm{cor}(x,y)=\frac{\mathrm{cov}(X,Y)}{s_X*s_Y}$$

Similar to the covariance matrix, the Pearson's correlation coefficient matrix Σ is:

$$\Sigma_{ij}=\mathrm{cor}(X_i,X_j)$$

where $X=[X_1,X_2,...,X_n]$.

## Prerequisites

- The input data columns are of INTEGER or DOUBLE data type.
- The input data does not contain null value. The algorithm will issue errors when encountering null values.

## _SYS_AFL.PAL_MULTIVARIATE_ANALYSIS

This function reads input data and calculates the basic multivariate analysis values for each column, including covariance matrix and Pearson's correlation coefficient matrix.

### Overview of all Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<RESULT table>` | OUT |

### Input Table

| Table | Column | Reference for Output Table | Data Type | Description |
|---|---|---|---|---|
| DATA | Dataset columns | DATASET | INTEGER, DECIMAL(p, s), or DOUBLE | Attribute data |

### Parameter Table

Mandatory Parameters

None.

Optional Parameter

The following parameter is optional. If it is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| RESULT_TYPE | INTEGER | 0 | Definition of result matrix.<br>• 0: Covariance matrix<br>• 1: Pearson's correlation coefficient matrix |

### Output Table

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| RESULT | 1st column | NVARCHAR | ID | Row name of output matrix to keep the row order correctly |
| | 2nd column | DOUBLE | DATASET (in input table) column name | Value of covariance or Pearson's correlation coefficient matrix |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_MULTIVARSTAT_DATA_TBL;
CREATE COLUMN TABLE PAL_MULTIVARSTAT_DATA_TBL (
    "X" INTEGER,
    "Y" DOUBLE
);
INSERT INTO PAL_MULTIVARSTAT_DATA_TBL VALUES (1,2.4);
INSERT INTO PAL_MULTIVARSTAT_DATA_TBL VALUES (5,3.5);
INSERT INTO PAL_MULTIVARSTAT_DATA_TBL VALUES (3,8.9);
INSERT INTO PAL_MULTIVARSTAT_DATA_TBL VALUES (10,-1.4);
INSERT INTO PAL_MULTIVARSTAT_DATA_TBL VALUES (-4,-3.5);
INSERT INTO PAL_MULTIVARSTAT_DATA_TBL VALUES (11,32.8);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('RESULT_TYPE',0,null,null); //default
value is 0, it can be {0,1}
CALL "_SYS_AFL"."PAL_MULTIVARIATE_ANALYSIS"(PAL_MULTIVARSTAT_DATA_TBL,
#PAL_PARAMETER_TBL, ?);
```

### Expected Result

| | ID | X | Y |
|---|---|---|---|
| 1 | X | 31.866666666666667 | 44.47333333333333 |
| 2 | Y | 44.47333333333333 | 176.67766666666662 |

> **Note**
>
> The wrapper procedure generation approach to call PAL functions is still supported. For more information, refer to the older versions (prior to SAP HANA Platform 2.0 SPS 02) of this document.

## 5.9.17 One-Sample Median Test

This algorithm performs a one-sample non-parametric test to check whether the median of the data is different from a user specified one.

It implements the sign test which does not have any assumption on the underlying distribution of data. If the data is drawn from a symmetric distribution, you can also use the Wilcoxon signed rank test in PAL.

Given a vector x and a parameter $m_0$, this function tests the following null hypothesis:

$H_0$: the median of the population from which x is drawn is $m_0$.

The alternative hypothesis depends on the type of tests. Specifically,

- If the TEST_TYPE parameter is set to 0, we have
  $H_A$: the median of the population does not equal $m_0$.
- If the TEST_TYPE parameter is set to 1, we have
  $H_A$: the median of the population is less than $m_0$.
- If the TEST_TYPE parameter is set to 2, we have
  $H_A$: the median of the population is greater than $m_0$.

P value of the test is calculated approximately. It is recommended that you use this function when the sample size is no less than 20. This function also reports the estimated median and the corresponding confidence interval.

### Prerequisites

- The input data only has one column with the type of INTEGER or DOUBLE.
- The input data must have more than 2 records.
- The input data does not contain null value.

### _SYS_AFL.PAL_SIGN_TEST

This function tests the median of a dataset.

**Overview of all Tables**

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `< INPUT table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<RESULT table>` | OUT |

**Input Table**

| Table | Column | Data Type | Description |
|---|---|---|---|
| INPUT | 1st column | INTEGER, DOUBLE, or DECIMAL(p, s) | Attribute data |

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| THREAD_RATIO | DOUBLE | 0 | Specifies the ratio of total number of threads that can be used by this function. The value range is from 0 to 1, where 0 means only using 1 thread, and 1 means using at most all the currently available threads. Values outside the range will be ignored and this function heuristically determines the number of threads to use. |
| M0 | DOUBLE | 0 | The median of data. It only matters in the one sample test. |
| CONFIDENCE_INTERVAL | DOUBLE | 0.95 | Confidence interval for the estimated median. |
| TEST_TYPE | INTEGER | 0 | The alternative hypothesis type:<br>• 0: Two sides<br>• 1: Less<br>• 2: Greater |

### Output Table

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| RESULT | 1st column | NVARCHAR(100) | STAT_NAME | Name of statistics |
| | 2nd column | DOUBLE | STAT_VALUE | Value of statistics |

### Example

Assume that:

- DM_PAL is a schema belonging to USER1;
- USER1 has been assigned the AFL__SYS_AFL_AFLPAL_EXECUTE or AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION role.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_SIGNTEST_DATA_TBL;
CREATE COLUMN TABLE PAL_SIGNTEST_DATA_TBL ("X" INT);
INSERT INTO PAL_SIGNTEST_DATA_TBL VALUES (85);
INSERT INTO PAL_SIGNTEST_DATA_TBL VALUES (65);
INSERT INTO PAL_SIGNTEST_DATA_TBL VALUES (20);
INSERT INTO PAL_SIGNTEST_DATA_TBL VALUES (56);
INSERT INTO PAL_SIGNTEST_DATA_TBL VALUES (30);
INSERT INTO PAL_SIGNTEST_DATA_TBL VALUES (46);
INSERT INTO PAL_SIGNTEST_DATA_TBL VALUES (83);
INSERT INTO PAL_SIGNTEST_DATA_TBL VALUES (33);
INSERT INTO PAL_SIGNTEST_DATA_TBL VALUES (89);
INSERT INTO PAL_SIGNTEST_DATA_TBL VALUES (72);
INSERT INTO PAL_SIGNTEST_DATA_TBL VALUES (51);
INSERT INTO PAL_SIGNTEST_DATA_TBL VALUES (76);
INSERT INTO PAL_SIGNTEST_DATA_TBL VALUES (68);
INSERT INTO PAL_SIGNTEST_DATA_TBL VALUES (82);
INSERT INTO PAL_SIGNTEST_DATA_TBL VALUES (27);
INSERT INTO PAL_SIGNTEST_DATA_TBL VALUES (59);
INSERT INTO PAL_SIGNTEST_DATA_TBL VALUES (69);
INSERT INTO PAL_SIGNTEST_DATA_TBL VALUES (40);
INSERT INTO PAL_SIGNTEST_DATA_TBL VALUES (64);
INSERT INTO PAL_SIGNTEST_DATA_TBL VALUES (8);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE
    #PAL_PARAMETER_TBL
    ("PARAM_NAME" VARCHAR(256), "INT_VALUE" INTEGER, "DOUBLE_VALUE" DOUBLE,
"STRING_VALUE" VARCHAR(1000));
INSERT INTO #PAL_PARAMETER_TBL VALUES ('M0',NULL,40,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('TEST_TYPE', 0,NULL,NULL); //default
value is 0, it can be {0,1,2}
CALL _SYS_AFL.PAL_SIGN_TEST(PAL_SIGNTEST_DATA_TBL, #PAL_PARAMETER_TBL, ?);
```

### Expected Result

RESULT:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | total number | 20 |
| 2 | number smaller than m0 | 5 |
| 3 | number larger than m0 | 14 |
| 4 | estimated median | 61.5 |
| 5 | CI for estimated media... | 27 |
| 6 | CI for estimated media... | 83 |
| 7 | sign test p-value | 0.06645742001693122 |

> **Note**
>
> The wrapper procedure generation approach to call PAL functions is still supported. For more information, refer to the older versions (prior to SAP HANA Platform 2.0 SPS 02) of this document.

## Related Information

Wilcox Signed Rank Test [page 1300]

# 5.9.18 T-Test

This algorithm is used to test whether the mean of one sample or the mean difference of two samples is significantly different from a user specified value.

The most frequently used t-tests are as follows:

1. One Sample T-Test: It is a statistical procedure used to determine whether a sample of observations could have been generated by a process with a specific mean.
   Let $x_1, x_2, \ldots, x_n$ be a sample, and $m_0$ is the hypothesized value of the mean.
   Let sample mean of x be:
   $$\bar{x}=\frac{\sum_{i=1}^{n} x_i}{n}.$$
   Therefore, the sample standard deviation is
   $$\hat{\sigma}=\sqrt{\frac{\sum_{i=1}^{n}(x_i-\bar{x})^2}{n-1}}.$$
   Then the test statistic is
   $$t=\frac{\bar{x}-m_0}{\hat{\sigma}/\sqrt{n}}.$$
   The t value will be used to calculate a p-value by comparing with the value of t-distribution. The degrees of freedom is set to `(n-1)`.
2. Paired Sample T-Test: It is a statistic procedure used to determine whether the mean difference between two sets of paired observations is equal to a given value.
   Let $x_1, x_2, \ldots, x_n$ and $y_1, y_2, \ldots, y_n$ be two dependent sample datasets, and $m_0$ is the hypothesized value of the mean difference.
   Let differences between sample x and sample y be $d_1, d_2, \ldots, d_n$, $d_i=x_i-y_i$, the differences of mean between sample x and sample y is
   $$\bar{d}=\frac{\sum_{i=1}^{n} d_i}{n}.$$
   Therefore, the sample standard deviation is
   $$\hat{\sigma}=\sqrt{\frac{\sum_{i=1}^{n}(d_i-\bar{d})^2}{n-1}}.$$
   Then the test statistic is
   $$t=\frac{\bar{d}-m_0}{\hat{\sigma}/\sqrt{n}}.$$
   The t value will be used to calculate a p-value by comparing with the value of t-distribution. The degrees of freedom is set to `(n-1)`.
3. Independent Sample T-Test: It is a statistic procedure used to determine whether the mean difference between two unrelated groups is equal to a given value.
   Let $x_1, x_2, \ldots, x_n$ and $y_1, y_2, \ldots, y_m$ be two independent sample datasets, and $m_0$ is the hypothesized value of the mean difference.

Let the sample mean of x and y be

$$\bar{x}=\frac{\sum_{i=1}^{n} x_i}{n} \text{ and } \bar{y}=\frac{\sum_{i=1}^{m} y_i}{m}.$$

Therefore, the sample standard deviation of x and y are

$$\hat{\sigma}_x=\sqrt{\frac{\sum_{i=1}^{n}(x_i-\bar{x})^2}{n-1}} \text{ and } \hat{\sigma}_y=\sqrt{\frac{\sum_{i=1}^{m}(y_i-\bar{y})^2}{m-1}}.$$

If we assume that two distributions have the same variance, then the test statistic is

$$t=\frac{\bar{x}-\bar{y}-m_0}{\sigma_p\sqrt{\frac{1}{n}+\frac{1}{m}}},$$

where $\sigma_p=\sqrt{\frac{(n-1)\hat{\sigma}_x^2+(m-1)\hat{\sigma}_y^2}{n+m-2}}$.

The t value will be used to calculate a p-value by comparing with the value of t-distribution. The degrees of freedom is set to `(n+m-2)`.

If we assume the variances of two distributions are not equal, then the test statistic is

$$t=\frac{\bar{x}-\bar{y}-m_0}{\bar{\sigma}},$$

where $\bar{\sigma}=\sqrt{\frac{\hat{\sigma}_x^2}{n}+\frac{\hat{\sigma}_y^2}{m}}$.

The t value will be used to calculate a p-value by comparing with the value of t-distribution. The degrees of freedom is calculated as

$$d.f.=\frac{\left(\frac{\hat{\sigma}_x^2}{n}+\frac{\hat{\sigma}_y^2}{m}\right)^2}{\frac{\left(\frac{\hat{\sigma}_x^2}{n}\right)^2}{n-1}+\frac{\left(\frac{\hat{\sigma}_y^2}{m}\right)^2}{m-1}}.$$

In PAL, when the input table has only one column, the algorithm performs one sample t-test; when the input table has two columns and the value of the PAIRED parameter is set to 1, the algorithm performs paired sample t-test; in other cases, the algorithm performs independent sample t-test.

## Prerequisites

The input data does not contain null value when one-sample t-test or paired sample t-test is performed. The algorithm issues errors when encountering null values.

## _SYS_AFL.PAL_T_TEST

This function performs t-test on input data.

### Overview of all Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |

| Position | Table Name | Parameter Type |
|---|---|---|
| 2 | `<PARAMETER table>` | IN |
| 3 | `<STATISTIC table>` | OUT |

### Input table

| Table | Column | Data Type | Description | Constraint |
|---|---|---|---|---|
| DATA | 1st column | INTEGER, DOUBLE, or DECIMAL(p, s) | Attribute data | |
| | 2nd column | INTEGER, DOUBLE, or DECIMAL(p, s) | Attribute data | Only valid when performing independent t-test or paired t-test. |

### Parameter Table

Mandatory Parameter

None.

Optional Parameter

The following the parameters are optional. If a parameter is not specified. PAL will use its default value.

| Name | Data Type | Default Value | Description | Constraint |
|---|---|---|---|---|
| TEST_TYPE | INTEGER | 0 | The alternative hypothesis type.<br>• 0: Two sides<br>• 1: Less<br>• 2: Greater | |
| MU | DOUBLE | 0.0 | Hypothesized value $m_0$ for the test. | |
| PAIRED | INTEGER | 0 | Controls whether to perform paired t-test.<br>• 0: No<br>• 1: Yes | Only valid when the input data has two columns. |
| VAR_EQUAL | INTEGER | 0 | Controls whether to assume that the two samples have equal variance.<br>• 0: No<br>• 1: Yes | Only valid when the input data has two columns and the PAIRED parameter is set to 0. |

| Name | Data Type | Default Value | Description | Constraint |
|---|---|---|---|---|
| CONF_LEVEL | DOUBLE | 0.95 | Specifies confidence level for alternative hypothesis confidence interval. | |

### Output table

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| STATISTIC | 1st column | NVARCHAR(100) | STAT_NAME | Name of statistics |
| | 2nd column | DOUBLE | STAT_VALUE | Value of statistics |

### Examples

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

### Example 1: One-sample T-test

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_TTEST_DATA1_TBL;
CREATE COLUMN TABLE PAL_TTEST_DATA1_TBL ("X1" DOUBLE);
INSERT INTO PAL_TTEST_DATA1_TBL VALUES (1);
INSERT INTO PAL_TTEST_DATA1_TBL VALUES (2);
INSERT INTO PAL_TTEST_DATA1_TBL VALUES (4);
INSERT INTO PAL_TTEST_DATA1_TBL VALUES (7);
INSERT INTO PAL_TTEST_DATA1_TBL VALUES (3);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL(
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('TEST_TYPE', 0, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MU', NULL, 0, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('CONF_LEVEL', NULL, 0.95, NULL);
CALL _SYS_AFL.PAL_T_TEST (PAL_TTEST_DATA1_TBL, #PAL_PARAMETER_TBL, ?);
```

### Expected Result

STATISTIC:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | t-value | 3.3023719320146983 |
| 2 | degree of freedom | 4 |
| 3 | p-value | 0.02986696837235... |
| 4 | _PAL_MEAN_X1_ | 3.4 |
| 5 | confidence level | 0.95 |
| 6 | lowerLimit | 0.541474883021122 |
| 7 | upperLimit | 6.258525116978878 |

### Example 2: Independent Sample T-test

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_TTEST_DATA2_TBL;
CREATE COLUMN TABLE PAL_TTEST_DATA2_TBL ("X1" DOUBLE, "X2" DOUBLE);
INSERT INTO PAL_TTEST_DATA2_TBL VALUES (1, 10);
INSERT INTO PAL_TTEST_DATA2_TBL VALUES (2, 12);
INSERT INTO PAL_TTEST_DATA2_TBL VALUES (4, 11);
INSERT INTO PAL_TTEST_DATA2_TBL VALUES (7, 15);
INSERT INTO PAL_TTEST_DATA2_TBL VALUES (NULL, 10);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('TEST_TYPE', 0, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MU', NULL, 0, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('VAR_EQUAL', 0, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('CONF_LEVEL', NULL, 0.95, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PAIRED', 0, NULL, NULL); -- independent
sample t-test
CALL _SYS_AFL.PAL_T_TEST (PAL_TTEST_DATA2_TBL, #PAL_PARAMETER_TBL, ?);
```

### Expected Result

STATISTIC:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | t-value | -5.0137741307804005 |
| 2 | degree of freedom | 5.649756717903349 |
| 3 | p-value | 0.002874909617985... |
| 4 | _PAL_MEAN_X1_ | 3.5 |
| 5 | _PAL_MEAN_X2_ | 11.6 |
| 6 | confidence level | 0.95 |
| 7 | lowerLimit | -12.113277797918558 |
| 8 | upperLimit | -4.08672220208144 |

### Example 3: Paired Sample T-test

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_TTEST_DATA3_TBL;
CREATE COLUMN TABLE PAL_TTEST_DATA3_TBL ("X1" DOUBLE, "X2" DOUBLE);
INSERT INTO PAL_TTEST_DATA3_TBL VALUES (1, 10);
INSERT INTO PAL_TTEST_DATA3_TBL VALUES (2, 12);
```

```
INSERT INTO PAL_TTEST_DATA3_TBL VALUES (4, 11);
INSERT INTO PAL_TTEST_DATA3_TBL VALUES (7, 15);
INSERT INTO PAL_TTEST_DATA3_TBL VALUES (3, 10);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('TEST_TYPE', 0, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('CONF_LEVEL', NULL, 0.95, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PAIRED', 1, NULL, NULL); -- paired
sample t-test
CALL _SYS_AFL.PAL_T_TEST (PAL_TTEST_DATA3_TBL, #PAL_PARAMETER_TBL, ?);
```

**Expected Result**

STATISTIC:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | t-value | -14.062883981685724 |
| 2 | degree of freedom | 4 |
| 3 | p-value | 0.000148372565074... |
| 4 | _PAL_MEAN_DIFF... | -8.2 |
| 5 | confidence level | 0.95 |
| 6 | lowerLimit | -9.818931742794032 |
| 7 | upperLimit | -6.581068257205964 |

> **Note**
>
> The wrapper procedure generation approach to call PAL functions is still supported. For more information, refer to the older versions (prior to SAP HANA Platform 2.0 SPS 02) of this document.

## 5.9.19 Univariate Analysis

Univariate analysis provides an overview of the dataset, which reveals the most important information of each variable. It is able to handle both continuous and categorical variables even with null value in the dataset.

For any continuous variable, if one denotes the data in one column as $x_i(i=1,\ldots,n)$, univariate analysis returns the following statistical quantities of it. Note that these statistical quantities are calculated under the condition that the data is merely a sample of the population.

- **Valid observations**: the count of all non-null values
- **Min**: the minimum value
- **Lower quartile**: the 25% percentile
- **Median**: the middle value of the data column
- **Upper quartile**: the 75% percentile
- **Max**: the maximum value
- **Mean**: the arithmetic mean

  $\overline{x}=\sum_{i=1}^{n} x_i$

- **Confidence interval for mean**:
  Under the assumption that this column of data follows a normal distribution, it is able to calculate the confidence interval corresponding to a significance level α, i.e. `(1-α)×100%` confidence level, for the mean as

  $$\bar{x} \pm t(1-\frac{\alpha}{2}, n-1) \cdot \frac{\sigma}{\sqrt{n}},$$

  where σ is the estimated standard deviation. The function $t(1-\frac{\alpha}{2}, n-1)$ refers to the Student's `t`-distribution with `(n-1)` degree of freedom.
- **Trimmed mean**:
  It is the average of the data after removing a certain amount at both head and tail. For example, a 10% trimmed mean calculates the average of data after taking out 10% of the largest elements and the same amount of the smallest elements.
- **Variance**:

  $$Var=\frac{1}{n-1}\sum_{i=1}^{n}(x_i-\bar{x})^2$$

- **Standard deviation**:

  $$\sigma=\sqrt{Var}$$

- **Skewness**:

  $$Skewness=\sqrt{n} \cdot \frac{\sum_{i=1}^{n}(x_i-\bar{x})^3}{\left[\sum_{i=1}^{n}(x_i-\bar{x})^2\right]^{3/2}}$$

- **Kurtosis**:

  $$Kurtosis= n \cdot \frac{\sum_{i=1}^{n}(x_i-\bar{x})^4}{\left[\sum_{i=1}^{n}(x_i-\bar{x})^2\right]^2}-3$$

For a categorical variable, this package returns the occurrence and the percentage of each category. Note that null is also treated as a category for the categorical variable.

## Prerequisites

None.

## _SYS_AFL.PAL_UNIVARIATE_ANALYSIS

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<CONTINUOUS RESULT table>` | OUT |

| Position | Table Name | Parameter Type |
|---|---|---|
| 4 | `<CATEGORICAL RESULT table>` | OUT |

### Input Table

| Table | Column | Data Type | Description |
|---|---|---|---|
| DATA | All columns | INTEGER, DOUBLE, DECIMAL(p, s), VARCHAR, or NVARCHAR | Attributes data. There is no ID column by default. If ID column is present, it must be the first column, and the `HAS_ID` parameter must be set to 1. |

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| CATEGORICAL_VARIABLE | VARCHAR | No default value | Indicates whether or not a column data is actually corresponding to a category variable even the data type of this column is INTEGER. By default, 'VARCHAR' or 'NVARCHAR' is category variable, and 'INTEGER' or 'DOUBLE' is continuous variable. The data type of the column specified by this parameter must be INTEGER. |
| HAS_ID | INTEGER | 1 | Specifies whether the input data table has an ID column. If it exists, the ID column must be the first column.<br>• 0: No ID column<br>• 1: Has an ID column |

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| SIGNIFICANCE_LEVEL | DOUBLE | 0.05 | Specifies the significance level when the algorithm calculates the confidence interval of the sample mean. If it is set to 0, the program outputs the mean.<br><br>Value range: Larger than 0 and smaller than 1. |
| TRIMMED_PERCENTAGE | DOUBLE | 0.05 | Specifies the ratio of data at both head and tail that will be dropped in the process of calculating the trimmed mean.<br><br>Value range: No less than 0 and smaller than 0.5. |

### Output Tables

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| CONTINUOUS_RESULT | 1st column | NVARCHAR(256) | VARIABLE_NAME | Variable names. |
| | 2nd column | NVARCHAR(100) | STAT_NAME | Names of statistical quantities, including number of valid observation, min, lower quartile, median, upper quartile, max, mean, confidence interval for the mean (lower and upper bound), trimmed mean, variance, standard deviation, skewness, and kurtosis (14 quantities in total). |
| | 3rd column | DOUBLE | STAT_VALUE | Values for the corresponding statistical quantities. |
| CATEGORICAL_RESULT | 1st column | NVARCHAR(256) | VARIABLE_NAME | Variable names |

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| | 2nd column | NVARCHAR(256) | CATEGORY | Category names of the corresponding variables. Null is also treated as a category. |
| | 3rd column | NVARCHAR(100) | STAT_NAME | Names of statistical quantities: number of observations, percentage of the current category for a variable (including null). |
| | 4th column | DOUBLE | STAT_VALUE | Values for the corresponding statistical quantities. |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA "DM_PAL";
DROP TABLE PAL_UNIVARIATE_ANALYSIS_DATA_TBL;
CREATE COLUMN TABLE
    PAL_UNIVARIATE_ANALYSIS_DATA_TBL
     ("X1" DOUBLE, "X2" DOUBLE, "X3" INTEGER, "X4" VARCHAR(100));
INSERT INTO PAL_UNIVARIATE_ANALYSIS_DATA_TBL VALUES (1.2,   NULL,  1, 'A');
INSERT INTO PAL_UNIVARIATE_ANALYSIS_DATA_TBL VALUES (2.5,   NULL,  2, 'C');
INSERT INTO PAL_UNIVARIATE_ANALYSIS_DATA_TBL VALUES (5.2,   NULL,  3, 'A');
INSERT INTO PAL_UNIVARIATE_ANALYSIS_DATA_TBL VALUES (-10.2, NULL,  2, 'A');
INSERT INTO PAL_UNIVARIATE_ANALYSIS_DATA_TBL VALUES (8.5,   NULL,  2, 'C');
INSERT INTO PAL_UNIVARIATE_ANALYSIS_DATA_TBL VALUES (100,   NULL,  3, 'B');
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL(
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SIGNIFICANCE_LEVEL', NULL, 0.05, NULL);
--default value is 0.05
INSERT INTO #PAL_PARAMETER_TBL VALUES ('TRIMMED_PERCENTAGE', NULL, 0.2, NULL);
--default value is 0.05
INSERT INTO #PAL_PARAMETER_TBL VALUES ('CATEGORY_COL', 3, NULL, NULL); --no
default
CALL _SYS_AFL.PAL_UNIVARIATE_ANALYSIS(PAL_UNIVARIATE_ANALYSIS_DATA_TBL,
#PAL_PARAMETER_TBL, ?, ?);
```

### Expected Result

CONTINUOUS_RESULT:

| | VARIABLE_NAME | STAT_NAME | STAT_VALUE |
|---|---|---|---|
| 1 | X2 | valid observations | 0 |
| 2 | X1 | valid observations | 6 |
| 3 | X1 | min | -10.2 |
| 4 | X1 | lower quartile | 1.2 |
| 5 | X1 | median | 3.85 |
| 6 | X1 | upper quartile | 8.5 |
| 7 | X1 | max | 100 |
| 8 | X1 | mean | 17.866666666666664 |
| 9 | X1 | CI for mean, lo... | -24.87954924997482 |
| 10 | X1 | CI for mean, up... | 60.61288258330815 |
| 11 | X1 | trimmed mean | 4.35 |
| 12 | X1 | variance | 1,659.1426666666666 |
| 13 | X1 | standard deviati... | 40.732575006580014 |
| 14 | X1 | skewness | 1.688494949952235 |
| 15 | X1 | kurtosis | 1.0361477358708582 |

CATEGORICAL_RESULT:

| | VARIABLE_NAME | CATEGORY | STAT_NAME | STAT_VALUE |
|---|---|---|---|---|
| 1 | X3 | __PAL_NULL__ | count | 0 |
| 2 | X3 | __PAL_NULL__ | percentage(%) | 0 |
| 3 | X3 | 1 | count | 1 |
| 4 | X3 | 1 | percentage(%) | 16.666666666666664 |
| 5 | X3 | 2 | count | 3 |
| 6 | X3 | 2 | percentage(%) | 50 |
| 7 | X3 | 3 | count | 2 |
| 8 | X3 | 3 | percentage(%) | 33.33333333333333 |
| 9 | X4 | __PAL_NULL__ | count | 0 |
| 10 | X4 | __PAL_NULL__ | percentage(%) | 0 |
| 11 | X4 | A | count | 3 |
| 12 | X4 | A | percentage(%) | 50 |
| 13 | X4 | B | count | 1 |
| 14 | X4 | B | percentage(%) | 16.666666666666664 |
| 15 | X4 | C | count | 2 |
| 16 | X4 | C | percentage(%) | 33.33333333333333 |

**Note**

The wrapper procedure generation approach to call PAL functions is still supported. For more information, refer to the older versions (prior to SAP HANA Platform 2.0 SPS 02) of this document.

## 5.9.20 Wilcox Signed Rank Test

This algorithm performs a one-sample or paired two-sample non-parametric test to check whether the median of the data is different from a specific value.

It assumes that the underlying distribution of the data set is symmetric. If the population is not symmetrically distributed, it is recommended that you use the one-sample median test function in PAL instead.

As stated above, this function is able to work in two modes:

1. One-sample: With one column of data, it tests the null hypothesis that if the population where the data is drawn from is symmetric about a user defined location parameter $\mu_0$. The alternative hypothesis could be:
   - Two sided: The true location is different from $\mu_0$.
   - Left: The true location is less than $\mu_0$.
   - Greater: The true location is greater than $\mu_0$.
2. Paired two-sample: With two equal sized columns of data, it tests the null hypothesis that the underlying distribution of the difference of these two is symmetric about 0.
   - Two sided: The true difference of locations is different from 0.
   - Left: The true difference of locations is less than 0.
   - Greater: The true difference of locations is greater than 0.

This function reports both the rank sum statistic and the corresponding p value. P value is calculated under the normal approximation to better address the possible ties in the dataset. It also means that it is desirable to have a dataset with at least 20 samples.

### Prerequisites

- The input data has one table with one or more columns of INTEGER or DOUBLE type numbers.
- The input data must have at least 2 records.
- The input data does not contain null value.

### _SYS_AFL.PAL_WILCOX_TEST

This function tests the symmetrical point in one-sample or two-sample data.

**Overview of all Tables**

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<INPUT table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<RESULT table>` | OUT |

**Input Table**

| Table | Column | Data Type | Description |
|---|---|---|---|
| INPUT | 1st column | INTEGER, DOUBLE, or DECIMAL(p, s) | Attribute data |
| | 2nd column | INTEGER, DOUBLE, or DECIMAL(p, s) | Attribute data (optional) |

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| MU | DOUBLE | 0 | The location $\mu_0$ for the one sample test. It does not affect the two-sample test. |
| TEST_TYPE | INTEGER | 0 | The alternative hypothesis type:<br>• 0: Two sides<br>• 1: Less<br>• 2: Greater |
| CORRECTION | INTEGER | 1 | Controls whether or not to include the continuity correction for the p value calculation.<br>• 0: No<br>• 1: Yes |

### Output Table

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| RESULT | 1st column | NVARCHAR(100) | STAT_NAME | Name of statistics |
| | 2nd column | DOUBLE | STAT_VALUE | Value of statistics |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;

- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

### Example 1: One-sample test

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_WILCOXTEST_DATA_TBL;
CREATE COLUMN TABLE PAL_WILCOXTEST_DATA_TBL ("X" INT);
INSERT INTO PAL_WILCOXTEST_DATA_TBL VALUES (85);
INSERT INTO PAL_WILCOXTEST_DATA_TBL VALUES (65);
INSERT INTO PAL_WILCOXTEST_DATA_TBL VALUES (20);
INSERT INTO PAL_WILCOXTEST_DATA_TBL VALUES (56);
INSERT INTO PAL_WILCOXTEST_DATA_TBL VALUES (30);
INSERT INTO PAL_WILCOXTEST_DATA_TBL VALUES (46);
INSERT INTO PAL_WILCOXTEST_DATA_TBL VALUES (83);
INSERT INTO PAL_WILCOXTEST_DATA_TBL VALUES (33);
INSERT INTO PAL_WILCOXTEST_DATA_TBL VALUES (89);
INSERT INTO PAL_WILCOXTEST_DATA_TBL VALUES (72);
INSERT INTO PAL_WILCOXTEST_DATA_TBL VALUES (51);
INSERT INTO PAL_WILCOXTEST_DATA_TBL VALUES (76);
INSERT INTO PAL_WILCOXTEST_DATA_TBL VALUES (68);
INSERT INTO PAL_WILCOXTEST_DATA_TBL VALUES (82);
INSERT INTO PAL_WILCOXTEST_DATA_TBL VALUES (27);
INSERT INTO PAL_WILCOXTEST_DATA_TBL VALUES (59);
INSERT INTO PAL_WILCOXTEST_DATA_TBL VALUES (69);
INSERT INTO PAL_WILCOXTEST_DATA_TBL VALUES (40);
INSERT INTO PAL_WILCOXTEST_DATA_TBL VALUES (64);
INSERT INTO PAL_WILCOXTEST_DATA_TBL VALUES (8);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE
    #PAL_PARAMETER_TBL
    ("PARAM_NAME" VARCHAR(256), "INT_VALUE" INTEGER, "DOUBLE_VALUE" DOUBLE,
"STRING_VALUE" VARCHAR(1000));
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MU',NULL,40,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('TEST_TYPE', 0,NULL,NULL); //default
value is 0, it can be {0,1,2}
INSERT INTO #PAL_PARAMETER_TBL VALUES ('CORRECTION', 1,NULL,NULL);
CALL _SYS_AFL. PAL_WILCOX_TEST(PAL_WILCOXTEST_DATA_TBL, #PAL_PARAMETER_TBL, ?);
```

### Expected Result

RESULT:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | statistic | 158.5 |
| 2 | p-value | 0.011228240845317039 |

### Example 2: Two-sample test

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_WILCOXTEST_DATA_TBL;
CREATE COLUMN TABLE PAL_WILCOXTEST_DATA_TBL ("X1" INT, "X2" INT);
INSERT INTO PAL_WILCOXTEST_DATA_TBL VALUES (85, 60);
INSERT INTO PAL_WILCOXTEST_DATA_TBL VALUES (65, 68);
INSERT INTO PAL_WILCOXTEST_DATA_TBL VALUES (20, 20);
INSERT INTO PAL_WILCOXTEST_DATA_TBL VALUES (56, 89);
INSERT INTO PAL_WILCOXTEST_DATA_TBL VALUES (30, 7 );
INSERT INTO PAL_WILCOXTEST_DATA_TBL VALUES (46, 91);
INSERT INTO PAL_WILCOXTEST_DATA_TBL VALUES (83, 92);
INSERT INTO PAL_WILCOXTEST_DATA_TBL VALUES (33, 64);
INSERT INTO PAL_WILCOXTEST_DATA_TBL VALUES (89, 19);
INSERT INTO PAL_WILCOXTEST_DATA_TBL VALUES (72, 56);
INSERT INTO PAL_WILCOXTEST_DATA_TBL VALUES (51, 15);
INSERT INTO PAL_WILCOXTEST_DATA_TBL VALUES (76, 4 );
```

```
INSERT INTO PAL_WILCOXTEST_DATA_TBL VALUES (68, 17);
INSERT INTO PAL_WILCOXTEST_DATA_TBL VALUES (82, 23);
INSERT INTO PAL_WILCOXTEST_DATA_TBL VALUES (27, 74);
INSERT INTO PAL_WILCOXTEST_DATA_TBL VALUES (59, 27);
INSERT INTO PAL_WILCOXTEST_DATA_TBL VALUES (69, 25);
INSERT INTO PAL_WILCOXTEST_DATA_TBL VALUES (40, 76);
INSERT INTO PAL_WILCOXTEST_DATA_TBL VALUES (64, 26);
INSERT INTO PAL_WILCOXTEST_DATA_TBL VALUES (8,  79);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE
    #PAL_PARAMETER_TBL
    ("PARAM_NAME" VARCHAR(256), "INT_VALUE" INTEGER, "DOUBLE_VALUE" DOUBLE,
"STRING_VALUE" VARCHAR(1000));
INSERT INTO #PAL_PARAMETER_TBL VALUES ('TEST_TYPE', 0,NULL,NULL); //default
value is 0, it can be {0,1,2}
INSERT INTO #PAL_PARAMETER_TBL VALUES ('CORRECTION', 1,NULL,NULL);
CALL _SYS_AFL. PAL_WILCOX_TEST(PAL_WILCOXTEST_DATA_TBL, #PAL_PARAMETER_TBL, ?);
```

**Expected Result**

RESULT:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | statistic | 118.5 |
| 2 | p-value | 0.35462068579545436 |

> **Note**
>
> The wrapper procedure generation approach to call PAL functions is still supported. For more information, refer to the older versions (prior to SAP HANA Platform 2.0 SPS 02) of this document.

### Related Information

One-Sample Median Test [page 1286]

## 5.10 Social Network Analysis Algorithms

This section describes the algorithms provided by the PAL that are mainly used for social network analysis.

### 5.10.1 Link Prediction

Predicting missing links is a common task in social network analysis. The Link Prediction algorithm in PAL provides four methods to compute the distance of any two nodes using existing links in a social network, and make prediction on the missing links based on these distances.

Let *x* and *y* be two nodes in a social network, and $\Gamma(x)$ be the set containing the neighbor nodes of *x*, the four methods to compute the distance of *x* and *y* are briefly described as follows.

**Common Neighbors**

The quantity is computed as the number of common neighbors of *x* and *y*:

$$| \Gamma(x) \cap \Gamma(y) |$$

Then, it is normalized by the total number of nodes.

**Jaccard's Coefficient**

The quantity is just a slight modification of the common neighbors:

$$\frac{| \Gamma(x) \cap \Gamma(y) |}{| \Gamma(x) \cup \Gamma(y) |}$$

**Adamic/Adar**

The quantity is computed as the sum of inverse log degree over all the common neighbors:

$$\sum_{z \in \Gamma(x) \cap \Gamma(y)} \frac{1}{\log | \Gamma(z) |}$$

**Katz$_\beta$**

The quantity is computed as a weighted sum of the number of paths of length *l* connecting *x* and *y*:

$$\sum_{l=1}^{\infty} \beta^{l} | \mathrm{path}_{x,y}^{l} |$$

Where $\beta \in [0,1]$ is the user-specified parameter, and $| \mathrm{path}_{x,y}^{l} |$ is the number of paths with length *l* which starts from node *x* and ends at node *y*.

## Prerequisites

The input data does not contain any null value.

## _SYS_AFL.PAL_LINK_PREDICT

**Overview of All Tables**

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |

| Position | Table Name | Parameter Type |
|---|---|---|
| 2 | `<PARAMETER table>` | IN |
| 3 | `<RESULT table>` | OUT |

### Input Table

| Table | Column | Reference for Output Table | Data Type | Description |
|---|---|---|---|---|
| DATA | 1st column | NODE1 | INTEGER, VARCHAR, or NVARCHAR | Node 1 in existing edge (Node1 – Node2) |
| | 2nd column | NODE2 | INTEGER, VARCHAR, or NVARCHAR | Node 2 in existing edge (Node1 – Node2) |

### Parameter Table

Mandatory Parameter

The following parameter is mandatory and must be given a value.

| Name | Data Type | Description |
|---|---|---|
| METHOD | INTEGER | Prediction method:<br>• 1: Common Neighbors<br>• 2: Jaccard's Coefficient<br>• 3: Adamic/Adar<br>• 4: Katz |

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| BETA | DOUBLE | 0.005 | Parameter for the Katz method.<br>BETA should be between 0 and 1. A smaller BETA is preferred. | Only valid when METHOD is 4. |
| MIN_SCORE | DOUBLE | 0 | The links whose scores are lower than this threshold will be filtered out from the result table. | |

### Output Table

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| RESULT | 1st column | NODE1 (in input table) data type | NODE1 (in input table) column name | Node 1 in missing edge (Node1 – Node2) |
| | 2nd column | NODE2 (in input table) data type | NODE2 (in input table) column name | Node 2 in missing edge (Node1 – Node2) |
| | 3rd column | DOUBLE | SCORE | Prediction score |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_LINK_PREDICTION_DATA_TBL;
CREATE COLUMN TABLE PAL_LINK_PREDICTION_DATA_TBL(
    "NODE1" INTEGER,
    "NODE2" INTEGER
);
INSERT INTO PAL_LINK_PREDICTION_DATA_TBL VALUES ('1', '2');
INSERT INTO PAL_LINK_PREDICTION_DATA_TBL VALUES ('1', '4');
INSERT INTO PAL_LINK_PREDICTION_DATA_TBL VALUES ('2', '3');
INSERT INTO PAL_LINK_PREDICTION_DATA_TBL VALUES ('3', '4');
INSERT INTO PAL_LINK_PREDICTION_DATA_TBL VALUES ('5', '1');
INSERT INTO PAL_LINK_PREDICTION_DATA_TBL VALUES ('6', '2');
INSERT INTO PAL_LINK_PREDICTION_DATA_TBL VALUES ('7', '4');
INSERT INTO PAL_LINK_PREDICTION_DATA_TBL VALUES ('7', '5');
INSERT INTO PAL_LINK_PREDICTION_DATA_TBL VALUES ('6', '7');
INSERT INTO PAL_LINK_PREDICTION_DATA_TBL VALUES ('5', '4');
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL(
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR(1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('METHOD', 1, NULL, NULL);
CALL _SYS_AFL.PAL_LINK_PREDICT(PAL_LINK_PREDICTION_DATA_TBL,
"#PAL_PARAMETER_TBL", ?);
```

### Expected Result

| | NODE1 | NODE2 | SCORE |
|---|---|---|---|
| 1 | 1 | 3 | 0.2857142857142857 |
| 2 | 1 | 6 | 0.14285714285714285 |
| 3 | 1 | 7 | 0.2857142857142857 |
| 4 | 2 | 4 | 0.2857142857142857 |
| 5 | 2 | 5 | 0.14285714285714285 |
| 6 | 2 | 7 | 0.14285714285714285 |
| 7 | 4 | 6 | 0.14285714285714285 |
| 8 | 3 | 5 | 0.14285714285714285 |
| 9 | 3 | 6 | 0.14285714285714285 |
| 10 | 3 | 7 | 0.14285714285714285 |
| 11 | 5 | 6 | 0.14285714285714285 |

> **Note**
>
> The wrapper procedure generation approach to call PAL functions is still supported. For more information, refer to the older versions (prior to SAP HANA Platform 2.0 SPS 02) of this document.

## 5.10.2 PageRank

PageRank is an algorithm used by a search engine to measure the importance of website pages.

A website page is considered more important if it receives more links from other websites. PageRank represents the likelihood that a visitor will visit a particular page by randomly clicking of other webpages. Higher rank in PageRank means greater probability of the site being reached.

PageRank of a page is calculated by:

$$PR(A) = (1-d)/N + d\ (PR(T_1)/C(T_1) + \ldots + PR(T_n)/C(T_n))$$

Where

$PR(A)$ is the PageRank of page A,

$PR(T_i)$ is the PageRank of pages $T_i$ which links to page A,

$C(T_i)$ is the number of outbound links on page $T_i$,

$d$ is a damping factor.

### Prerequisites

The input data does not contain null value.

## _SYS_AFL.PAL_ PAGERANK

This function reads link information and calculates rank for each node.

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<RESULT table>` | OUT |

### Input Table

| Table | Column | Data Type | Description |
|---|---|---|---|
| DATA | 1st column | INTEGER, VARCHAR, or NVARCHAR | The FROM node of a link. |
| | 2nd column | INTEGER, VARCHAR, or NVARCHAR | The TO node of a link. |

### Parameter Table

Mandatory Parameter

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| DAMPING | Double | 0.85 | The damping factor **d**. Valid range is [0, 1]. |
| MAX_ITERATION | Integer | 0 | The maximum number of iterations of power method. The value 0 means no maximum number of iterations is set and the calculation stops when the result converges. |
| THRESHOLD | Double | 1e-6 | Specifies the stop condition. When the mean improvement value of ranks is less than this value, the program stops calculation. |

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| THREAD_RATIO | Double | 1.0 | Specifies the ratio of total number of threads this function uses. The value ranges from 0 to 1. A value of 0 uses one thread. A value of 1 uses all currently available threads. The function ignores values outside this range and heuristically determines the number of threads to use. |

> **Note**
>
> When the `DAMPING` parameter equals 1, PageRank algorithm never converges for some input data. In this case, setting `MAX_ITERATION` to 0 may cause the procedure to become stuck. PAL raises an SQL Warning for this parameter combination after the procedure stops.

### Output Table

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| RESULT | 1st column | VARCHAR or NVARCHAR | NODE | Node name. |
| | 2nd column | DOUBLE | RANK | The PageRank of the corresponding node. |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_PAGERANK_DATA_TBL;
CREATE COLUMN TABLE PAL_PAGERANK_DATA_TBL(
    "FROM_NODE" VARCHAR (100),
    "TO_NODE" VARCHAR (100)
);
INSERT INTO PAL_PAGERANK_DATA_TBL VALUES ('Node1', 'Node2');
INSERT INTO PAL_PAGERANK_DATA_TBL VALUES ('Node1', 'Node3');
INSERT INTO PAL_PAGERANK_DATA_TBL VALUES ('Node1', 'Node4');
INSERT INTO PAL_PAGERANK_DATA_TBL VALUES ('Node2', 'Node3');
INSERT INTO PAL_PAGERANK_DATA_TBL VALUES ('Node2', 'Node4');
INSERT INTO PAL_PAGERANK_DATA_TBL VALUES ('Node3', 'Node1');
INSERT INTO PAL_PAGERANK_DATA_TBL VALUES ('Node4', 'Node1');
INSERT INTO PAL_PAGERANK_DATA_TBL VALUES ('Node4', 'Node3');
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL(
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
```

```
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR(1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('DAMPING', NULL, 0.85, NULL);
CALL _SYS_AFL.PAL_PAGERANK(PAL_PAGERANK_DATA_TBL, "#PAL_PARAMETER_TBL", ?);
```

**Expected Result**

| | NODE | RANK |
|---|---|---|
| 1 | Node1 | 0.3681509531104541 |
| 2 | Node2 | 0.14180962694364851 |
| 3 | Node3 | 0.28796129337481535 |
| 4 | Node4 | 0.20207812657108173 |

## 5.11 Recommender System Algorithms

This section describes the algorithms provided by the PAL that are mainly used in recommender systems.

Recommender system tries to analyze patterns of user interest in items (products) to provide personalized recommendations that suit a user's taste. It seeks to predict the "ratings" or "preferences" that users would give to items and make recommendations according to those predictions.

### 5.11.1 Alternating Least Squares

Alternating least squares (ALS) is a powerful matrix factorization algorithm for building both explicit and implicit feedback based recommender systems.

Matrix factorization approach of building recommender systems provides high accuracy by characterizing both users and items with vectors of latent factors inferred from user-item rating patterns.

The predicted feedback $\hat{r}_{u,v}$ of a user u on an item v of the ALS model is given by

$$\hat{r}_{u,v}=f_u^T f_v,$$

where $f_u$,$f_v$ are the latent factors of u and v respectively.

- If the model is based on explicit feedback, the latent factors are trained by solving the optimization problem:

$$\mathrm{argmin}_{f_u,f_v}\frac{1}{|D|}\left(\sum_{r_{u,v}\in D}\left(r_{u,v}-f_u^T f_v\right)^2+\lambda\left(\sum_u n_u\|f_u\|^2+\sum_v n_v\|f_v\|^2\right)\right),$$

  where $r_{u,v}$ is observed explicit feedback, D is the set of all $r_{u,v}$ values, λ is the regularization parameter, and $n_u$ and $n_v$ are the number of feedback of u and u respectively.
- If the model is based on implicit feedback, the latent factors are trained by solving the optimization problem:

$$\mathrm{argmin}_{f_u,f_v}\frac{1}{N_u N_v}\left(\sum_{u,v}c_{u,v}\left(p_{u,v}-f_u^T f_v\right)^2+\lambda\left(\sum_u\|f_u\|^2+\sum_v\|f_v\|^2\right)\right),$$

where $N_u$ and $N_v$ are the total number of users and items respectively, $p_{u,v}$ indicates the preference of user u to item v, it is derived by binarizing the implicit feedback $r_{u,v}$: $p_{u,v}=\begin{cases}1, & r_{u,v}>0\\ 0, & r_{u,v}\leq 0\end{cases}$. And $c_{u,v}$ is the confidence level of the preference $p_{u,v}$: $c_{u,v}=1+\alpha r_{u,v}$, where $\alpha$ is a constant controlling the rate of increase.

Both optimization problems are solved by the ALS technique which can be parallelized. During each step of ALS, most of the computation is dedicated to solve a lot of linear systems. PAL provides two methods to solve these linear systems:

- Cholesky method – a direct method which gives the exact solution of a linear system.
- conjugate gradient (CG) method – an iterative method which gives an approximate solution.

The time each step takes and the rate of convergence of ALS using the two methods are similar when factor number (length of $f_u$,$f_v$) is small (~10). However, when factor number is large, ALS using the CG method can be much faster while keeping rate of convergence close to ALS using the Cholesky method.

- Null handle of PAL_ALS – data with both null user ID and item ID and data with null rating will be discarded.
- Null handle of PAL_ALS_PREDICT – data with null or invalid (not included in the trained model) user or item ID will not be processed.

## Prerequisites

None

## _SYS_AFL.PAL_ALS

This procedure trains a recommender model using the alternating least squares method.

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<MODEL_METADATA table>` | OUT |
| 4 | `<MODEL_MAP table>` | OUT |
| 5 | `<MODEL_FACTORS table>` | OUT |
| 6 | `<ITERATION_INFORMATION table>` | OUT |
| 7 | `<STATISTICS table>` | OUT |
| 8 | `<OPTIMAL_PARAMETER table>` | OUT |

### Input Tables

| Table | Column | Data Type | Description |
|---|---|---|---|
| DATA | 1st column | INTEGER, VARCHAR, or NVARCHAR | User name or user ID |
| | 2nd column | INTEGER, VARCHAR, or NVARCHAR | Item name or item ID |
| | 3rd columns | INTEGER, DOUBLE, or DECIMAL(p, s) | User feedback of the item |

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| HAS_ID | INTEGER | 0 | Specifies whether the input data table has an ID column. If it exists, the ID column must be the first column.<br>• 0: indicates there is no ID column in the data table. The algorithm will use all the columns to do the training and return the results.<br>• 1: indicates that the first column of the data table is ID column. The algorithm will ignore the first column to do the training and return the results. | |
| FACTOR_NUMBER | INTEGER | 8 | Length of factor vectors in the model. | |
| SEED | INTEGER | 0 | Specifies the seed for random number generator.<br>• 0: Uses the current time as the seed.<br>• Others: Uses the specified value as the seed. | |
| REGULARIZATION | DOUBLE | 1e-2 | L2 regularization of the factors. | |
| MAX_ITERATION | INTEGER | 20 | Maximum number of iterations. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| EXIT_THRESHOLD | DOUBLE | 0 | The algorithm exits if the value of cost function is not decreased more than the value of EXIT_THRESHOLD since the last check. If EXIT_THRESHOLD is set to 0, there is no check and the algorithm only exits on reaching the maximum number of iterations. Evaluations of cost function require additional calculations. You can set this parameter to 0 to avoid it. | |
| EXIT_INTERVAL | INTEGER | 5 | The algorithm calculates cost function and checks the exit criteria every `<EXIT_INTERVAL>` iterations. Evaluations of cost function require additional calculations. | Only valid when EXIT_THRESHOLD is not 0. |
| IMPLICIT_TRAIN | INTEGER | 0 | • 0: explicit ALS<br>• 1: implicit ALS | |
| LINEAR_SYSTEM_SOLVER | INTEGER | 0 | • 0: cholesky solver<br>• 1: cg solver (recommended when FACTOR_NUMBER is large) | |
| CG_MAX_ITERATION | INTEGER | 3 | Maximum number of iteration of cg solver. | Only valid when LINEAR_SYSTEM_SOLVER is 1. |
| ALPHA | DOUBLE | 1.0 | Used when computing the confidence level in implicit ALS. | Only valid when IMPLICIT_TRAIN is 1. |
| THREAD_RATIO | DOUBLE | 0 | Specifies the upper limit of thread usage in proportion of the current available threads. The valid range of the value is [0,1], 0 indicates that single thread is used and 1 means all the current available threads are used if necessary. Other values are ignored and the procedure determines the number of threads to use heuristically. | |

### Model Evaluation and Parameter Selection

The following parameters are only valid when model evaluation or parameter selection is enabled.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| RESAMPLING_METHOD | NVARCHAR | No default value | Specifies the resampling method for model evaluation or parameter selection.<br>• cv<br>• cv_sha<br>• cv_hyperband<br>• bootstrap<br>• bootstrap_sha<br>• bootstrap_hyperband<br>If no value is specified for this parameter, neither model evaluation nor parameter selection is activated. | Options like *_sha and *_hyperband can only be used in parameter selection, can not be used in model evaluation. |
| EVALUATION_METRIC | NVARCHAR | No default value | Specifies the evaluation metric for model evaluation or parameter selection.<br>• RMSE | |
| FOLD_NUM | INTEGER | 1 | Specifies the fold number for the cross validation method. | Mandatory and valid only when RESAMPLING_METHOD is set to cv*. |
| REPEAT_TIMES | INTEGER | 1 | Specifies the number of repeat times for resampling. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| PARAM_SEARCH_STRATEGY | NVARCHAR | No default value | Specifies the method to activate parameter selection.<br>• grid<br>• random | When *_sha is specified in RESAMPLING_METHOD, PARAM_SEARCH_STRATEGY is mandatory. When *_hyperband is specified in RESAMPLING_METHOD, PARAM_SEARCH_STRATEGY is set to random whatever other value is provided. Otherwise if this parameter is not set, only model evaluation is activated. |
| RANDOM_SEARCH_TIMES | INTEGER | No default value | Specifies the number of times to randomly select candidate parameters for selection. | Mandatory and valid when PARAM_SEARCH_STRATEGY is set to random. |
| REDUCTION_RATE | DOUBLE | 3.0 | Specifies reduction rate in SHA or hyperband method. | Only valid when RESAMPLING_METHOD is specified as *_sha or *_hyperband. |
| MIN_RESOURCE_RATE | DOUBLE | 0.0 | Specifies the minimum resource rate that should be used in SHA or hyperband iteration. | Only valid when RESAMPLING_METHOD is specified as *_sha or *_hyperband. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| AGGRESSIVE_ELIMINATION | INTEGER | 0 | Specifies whether to apply aggressive elimination while using SHA method. Aggressive elimination happens when the data size and parameters size to be searched does not match and there are still bunch of parameters to be searched while data size reaches its upper limits. If aggressive elimination is applied, lower bound of limit of data size will be used multiple times first to reduce number of parameters.<br>• 0: does not apply aggressive elimination<br>• 1: applies aggressive elimination | Only valid when RESAMPLING_METHOD is specified as *_sha. |
| SEED | INTEGER | 0 | Specifies the seed for random generation. Uses system time when 0 is specified. | |
| TIMEOUT | INTEGER | 0 | Specifies maximum running time for model evaluation or parameter selection, in seconds. No timeout when 0 is specified. | |
| PROGRESS_INDICATOR_ID | NVARCHAR | No default value | Sets an ID of progress indicator for model evaluation or parameter selection. No progress indicator is active if no value is provided. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| FACTOR_NUMBER_VALUES | NVARCHAR | No default value | Specifies values of FACTOR_NUMBER to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| FACTOR_NUMBER_RANGE | NVARCHAR | No default value | Specifies the range of FACTOR_NUMBER to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| REGULARIZATION_VALUES | NVARCHAR | No default value | Specifies values of REGULARIZATION to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| REGULARIZATION_RANGE | NVARCHAR | No default value | Specifies the range of REGULARIZATION to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| ALPHA_VALUES | NVARCHAR | No default value | Specifies values of ALPHA to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified and IMPLICIT_TRAIN is 1. |
| ALPHA_RANGE | NVARCHAR | No default value | Specifies the range of ALPHA to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified and IMPLICIT_TRAIN is 1. |

### Output Tables

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| MODEL_METADATA | 1st column | INTEGER | ROW_INDEX | ID |
| | 2nd column | NVARCHAR(5000) | METADATA_CONTENT | Model metadata content |
| MODEL_MAP | 1st column | INTEGER | ID | ID |
| | 2nd column | NVARCHAR(1000) | MAP | Map |
| MODEL_FACTORS | 1st column | INTEGER | FACTOR_ID | Factor ID |
| | 2nd column | DOUBLE | FACTOR | Factors |
| ITERATION_INFORMATION | 1st column | NVARCHAR(100) | ITERATION | Iteration |
| | 2nd column | NVARCHAR(100) | COST | Cost function value of corresponding iteration |

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| | 3rd column | NVARCHAR(100) | RMSE | Root mean square error of corresponding iteration |
| STATISTICS | 1st column | NVARCHAR(256) | STAT_NAME | Statistic names |
| | 2nd column | NVARCHAR(1000) | STAT_VALUE | Statistic values |
| OPTIMAL_PARAMETER | 1st column | NVARCHAR(256) | PARAM_NAME | Optimal parameters selected |
| | 2nd column | INTEGER | INT_VALUE | |
| | 3rd column | DOUBLE | DOUBLE_VALUE | |
| | 4th column | NVARCHAR(1000) | STRING_VALUE | |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

**Example 1: Train an ALS model without parameter selection**

```
SET SCHEMA "DM_PAL";
DROP TABLE PAL_ALS_DATA_TBL;
CREATE COLUMN TABLE PAL_ALS_DATA_TBL ("USER" NVARCHAR(100), "MOVIE"
NVARCHAR(100), "RATING" DOUBLE);
INSERT INTO PAL_ALS_DATA_TBL VALUES ('A', 'Movie1', 4.8);
INSERT INTO PAL_ALS_DATA_TBL VALUES ('A', 'Movie2', 4.0);
INSERT INTO PAL_ALS_DATA_TBL VALUES ('A', 'Movie4', 4.0);
INSERT INTO PAL_ALS_DATA_TBL VALUES ('A', 'Movie5', 4.0);
INSERT INTO PAL_ALS_DATA_TBL VALUES ('A', 'Movie6', 4.8);
INSERT INTO PAL_ALS_DATA_TBL VALUES ('A', 'Movie8', 3.8);
INSERT INTO PAL_ALS_DATA_TBL VALUES ('A', 'Bad_Movie', 2.5);
INSERT INTO PAL_ALS_DATA_TBL VALUES ('B', 'Movie2', 4.8);
INSERT INTO PAL_ALS_DATA_TBL VALUES ('B', 'Movie3', 4.8);
INSERT INTO PAL_ALS_DATA_TBL VALUES ('B', 'Movie4', 5.0);
INSERT INTO PAL_ALS_DATA_TBL VALUES ('B', 'Movie5', 5.0);
INSERT INTO PAL_ALS_DATA_TBL VALUES ('B', 'Movie7', 3.5);
INSERT INTO PAL_ALS_DATA_TBL VALUES ('B', 'Movie8', 4.8);
INSERT INTO PAL_ALS_DATA_TBL VALUES ('B', 'Bad_Movie', 2.8);
INSERT INTO PAL_ALS_DATA_TBL VALUES ('C', 'Movie1', 4.1);
INSERT INTO PAL_ALS_DATA_TBL VALUES ('C', 'Movie2', 4.2);
INSERT INTO PAL_ALS_DATA_TBL VALUES ('C', 'Movie4', 4.2);
INSERT INTO PAL_ALS_DATA_TBL VALUES ('C', 'Movie5', 4.0);
INSERT INTO PAL_ALS_DATA_TBL VALUES ('C', 'Movie6', 4.2);
INSERT INTO PAL_ALS_DATA_TBL VALUES ('C', 'Movie7', 3.2);
INSERT INTO PAL_ALS_DATA_TBL VALUES ('C', 'Movie8', 3.0);
INSERT INTO PAL_ALS_DATA_TBL VALUES ('C', 'Bad_Movie', 2.5);
INSERT INTO PAL_ALS_DATA_TBL VALUES ('D', 'Movie1', 4.5);
INSERT INTO PAL_ALS_DATA_TBL VALUES ('D', 'Movie3', 3.5);
INSERT INTO PAL_ALS_DATA_TBL VALUES ('D', 'Movie4', 4.5);
INSERT INTO PAL_ALS_DATA_TBL VALUES ('D', 'Movie6', 3.9);
INSERT INTO PAL_ALS_DATA_TBL VALUES ('D', 'Movie7', 3.5);
INSERT INTO PAL_ALS_DATA_TBL VALUES ('D', 'Movie8', 3.5);
```

```
INSERT INTO PAL_ALS_DATA_TBL VALUES ('D', 'Bad_Movie', 2.5);
INSERT INTO PAL_ALS_DATA_TBL VALUES ('E', 'Movie1', 4.5);
INSERT INTO PAL_ALS_DATA_TBL VALUES ('E', 'Movie2', 4.0);
INSERT INTO PAL_ALS_DATA_TBL VALUES ('E', 'Movie3', 3.5);
INSERT INTO PAL_ALS_DATA_TBL VALUES ('E', 'Movie4', 4.5);
INSERT INTO PAL_ALS_DATA_TBL VALUES ('E', 'Movie5', 4.5);
INSERT INTO PAL_ALS_DATA_TBL VALUES ('E', 'Movie6', 4.2);
INSERT INTO PAL_ALS_DATA_TBL VALUES ('E', 'Movie7', 3.5);
INSERT INTO PAL_ALS_DATA_TBL VALUES ('E', 'Movie8', 3.5);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL ("PARAM_NAME"
VARCHAR(256), "INT_VALUE" INTEGER, "DOUBLE_VALUE" DOUBLE, "STRING_VALUE"
VARCHAR(1000));
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FACTOR_NUMBER', 2, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('REGULARIZATION', NULL, 1e-2, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MAX_ITERATION', 20, NULL, NULL);
--INSERT INTO #PAL_PARAMETER_TBL VALUES ('EXIT_THRESHOLD', NULL, 1e-6, NULL);
--INSERT INTO #PAL_PARAMETER_TBL VALUES ('EXIT_INTERVAL', 5, NULL, NULL);
--INSERT INTO #PAL_PARAMETER_TBL VALUES ('LINEAR_SYSTEM_SOLVER', 0, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 0, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SEED', 1, NULL, NULL);
DROP TABLE PAL_ALS_MODEL_METADATA_TBL;
DROP TABLE PAL_ALS_MODEL_MAP_TBL;
DROP TABLE PAL_ALS_MODEL_FACTORS_TBL;
CREATE COLUMN TABLE PAL_ALS_MODEL_METADATA_TBL ("ROW_INDEX" INTEGER,
"METADATA_CONTENT" NVARCHAR(5000));
CREATE COLUMN TABLE PAL_ALS_MODEL_MAP_TBL ("ID" INTEGER, "MAP" NVARCHAR(1000));
CREATE COLUMN TABLE PAL_ALS_MODEL_FACTORS_TBL ("FACTOR_ID" INTEGER, "FACTOR"
DOUBLE);
CALL _SYS_AFL.PAL_ALS (PAL_ALS_DATA_TBL,
#PAL_PARAMETER_TBL, PAL_ALS_MODEL_METADATA_TBL, PAL_ALS_MODEL_MAP_TBL,
PAL_ALS_MODEL_FACTORS_TBL, ?, ?, ?) WITH OVERVIEW;
```

### Expected Result

MODEL_METADATA:

| | ROW_INDEX | METADATA_CONTENT |
|---|---|---|
| 1 | 0 | {"alpha":1.0,"exit_interval":5,"exit_threshold":0.0,"factor_number":2,"impli... |

MODEL_MAP:

| | ID | MAP |
|---|---|---|
| 1 | 0 | A |
| 2 | 1 | B |
| 3 | 2 | C |
| 4 | 3 | D |
| 5 | 4 | E |
| 6 | 5 | Movie1 |
| 7 | 6 | Movie2 |
| 8 | 7 | Movie4 |
| 9 | 8 | Movie5 |
| 10 | 9 | Movie6 |
| 11 | 10 | Movie8 |
| 12 | 11 | Bad_Movie |
| 13 | 12 | Movie3 |
| 14 | 13 | Movie7 |

MODEL_FACTORS:

| | FACTOR_ID | FACTOR |
|---|---|---|
| 1 | 0 | 1.1087754003694403 |
| 2 | 1 | -0.5823922812456241 |
| 3 | 2 | 1.355926210353692 |
| 4 | 3 | -0.7609686192815823 |
| 5 | 4 | 1.0841263622536566 |
| 6 | 5 | 0.281748826856790... |
| 7 | 6 | 1.1452444871042171 |
| 8 | 7 | 0.4186307823876875 |
| 9 | 8 | 1.1512570010257241 |
| 10 | 9 | 0.315341526589648... |
| 11 | 10 | 4.0024514911767595 |
| 12 | 11 | -0.473777482049698 |
| 13 | 12 | 3.5989874455744375 |
| 14 | 13 | 0.082773274043853... |
| 15 | 14 | 3.787112609542216 |
| 16 | 15 | 0.2704272211537582 |
| 17 | 16 | 3.7230743913683675 |
| 18 | 17 | 0.148151073593124... |
| 19 | 18 | 3.8533108062932144 |
| 20 | 19 | -0.8047811483444225 |
| 21 | 20 | 3.130351614215395 |
| 22 | 21 | -0.6001900635983951 |
| 23 | 22 | 2.18534444560339 |
| 24 | 23 | 0.094860758769336... |
| 25 | 24 | 3.2058636307867183 |
| 26 | 25 | -0.5394381116163688 |
| 27 | 26 | 2.845945209378896 |
| 28 | 27 | 0.489825779410849... |

ITERATION_INFORMATION:

| | ITERATION | COST | RMSE |
|---|---|---|---|
| 1 | 20 | 0.14724755464106934 | 0.1086315164152475 |

STATISTICS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| | | |
| | | |

OPTIMAL_PARAMETER:

| | PARAM_NAME | INT_VALUE | DOUBLE_VALUE | STRING_VALUE |
|---|---|---|---|---|
| | | | | |
| | | | | |

**Example 2: Train an implicit ALS model with parameter selection**

```
SET SCHEMA "DM_PAL";
DROP TABLE PAL_ALS_DATA_TBL;
CREATE COLUMN TABLE PAL_ALS_DATA_TBL ("USER" NVARCHAR(100), "MOVIE"
NVARCHAR(100), "RATING" DOUBLE);
INSERT INTO PAL_ALS_DATA_TBL VALUES ('A', 'Movie1', 4.8);
INSERT INTO PAL_ALS_DATA_TBL VALUES ('A', 'Movie2', 4.0);
INSERT INTO PAL_ALS_DATA_TBL VALUES ('A', 'Movie4', 4.0);
INSERT INTO PAL_ALS_DATA_TBL VALUES ('A', 'Movie5', 4.0);
INSERT INTO PAL_ALS_DATA_TBL VALUES ('A', 'Movie6', 4.8);
INSERT INTO PAL_ALS_DATA_TBL VALUES ('A', 'Movie8', 3.8);
INSERT INTO PAL_ALS_DATA_TBL VALUES ('A', 'Bad_Movie', 2.5);
INSERT INTO PAL_ALS_DATA_TBL VALUES ('B', 'Movie2', 4.8);
INSERT INTO PAL_ALS_DATA_TBL VALUES ('B', 'Movie3', 4.8);
INSERT INTO PAL_ALS_DATA_TBL VALUES ('B', 'Movie4', 5.0);
INSERT INTO PAL_ALS_DATA_TBL VALUES ('B', 'Movie5', 5.0);
INSERT INTO PAL_ALS_DATA_TBL VALUES ('B', 'Movie7', 3.5);
INSERT INTO PAL_ALS_DATA_TBL VALUES ('B', 'Movie8', 4.8);
INSERT INTO PAL_ALS_DATA_TBL VALUES ('B', 'Bad_Movie', 2.8);
INSERT INTO PAL_ALS_DATA_TBL VALUES ('C', 'Movie1', 4.1);
INSERT INTO PAL_ALS_DATA_TBL VALUES ('C', 'Movie2', 4.2);
INSERT INTO PAL_ALS_DATA_TBL VALUES ('C', 'Movie4', 4.2);
INSERT INTO PAL_ALS_DATA_TBL VALUES ('C', 'Movie5', 4.0);
INSERT INTO PAL_ALS_DATA_TBL VALUES ('C', 'Movie6', 4.2);
INSERT INTO PAL_ALS_DATA_TBL VALUES ('C', 'Movie7', 3.2);
INSERT INTO PAL_ALS_DATA_TBL VALUES ('C', 'Movie8', 3.0);
INSERT INTO PAL_ALS_DATA_TBL VALUES ('C', 'Bad_Movie', 2.5);
INSERT INTO PAL_ALS_DATA_TBL VALUES ('D', 'Movie1', 4.5);
INSERT INTO PAL_ALS_DATA_TBL VALUES ('D', 'Movie3', 3.5);
INSERT INTO PAL_ALS_DATA_TBL VALUES ('D', 'Movie4', 4.5);
INSERT INTO PAL_ALS_DATA_TBL VALUES ('D', 'Movie6', 3.9);
INSERT INTO PAL_ALS_DATA_TBL VALUES ('D', 'Movie7', 3.5);
INSERT INTO PAL_ALS_DATA_TBL VALUES ('D', 'Movie8', 3.5);
INSERT INTO PAL_ALS_DATA_TBL VALUES ('D', 'Bad_Movie', 2.5);
INSERT INTO PAL_ALS_DATA_TBL VALUES ('E', 'Movie1', 4.5);
INSERT INTO PAL_ALS_DATA_TBL VALUES ('E', 'Movie2', 4.0);
INSERT INTO PAL_ALS_DATA_TBL VALUES ('E', 'Movie3', 3.5);
INSERT INTO PAL_ALS_DATA_TBL VALUES ('E', 'Movie4', 4.5);
INSERT INTO PAL_ALS_DATA_TBL VALUES ('E', 'Movie5', 4.5);
INSERT INTO PAL_ALS_DATA_TBL VALUES ('E', 'Movie6', 4.2);
INSERT INTO PAL_ALS_DATA_TBL VALUES ('E', 'Movie7', 3.5);
INSERT INTO PAL_ALS_DATA_TBL VALUES ('E', 'Movie8', 3.5);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL ("PARAM_NAME"
VARCHAR(256), "INT_VALUE" INTEGER, "DOUBLE_VALUE" DOUBLE, "STRING_VALUE"
VARCHAR(1000));
INSERT INTO #PAL_PARAMETER_TBL VALUES ('IMPLICIT_TRAIN', 1, NULL, NULL);
```

```
INSERT INTO #PAL_PARAMETER_TBL VALUES ('RESAMPLING_METHOD', NULL, NULL, 'cv');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PARAM_SEARCH_STRATEGY', NULL, NULL,
'grid');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('EVALUATION_METRIC', NULL, NULL, 'RMSE');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FOLD_NUM', 5, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('REPEAT_TIMES', 1, NULL, NULL);
--INSERT INTO #PAL_PARAMETER_TBL VALUES ('TIMEOUT', 0, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PROGRESS_INDICATOR_ID', NULL, NULL,
'PAL_ALS');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FACTOR_NUMBER_RANGE', NULL, NULL, '[1,
1, 3]');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('REGULARIZATION_VALUES', NULL, NULL,
'{1e-1, 1e-2, 1e-3}');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('ALPHA_VALUES', NULL, NULL, '{1.0, 2.0}');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MAX_ITERATION', 20, NULL, NULL);
--INSERT INTO #PAL_PARAMETER_TBL VALUES ('EXIT_THRESHOLD', NULL, 1e-6, NULL);
--INSERT INTO #PAL_PARAMETER_TBL VALUES ('EXIT_INTERVAL', 5, NULL, NULL);
--INSERT INTO #PAL_PARAMETER_TBL VALUES ('LINEAR_SYSTEM_SOLVER', 0, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 0.5, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SEED', 1, NULL, NULL);
DROP TABLE PAL_ALS_MODEL_METADATA_TBL;
DROP TABLE PAL_ALS_MODEL_MAP_TBL;
DROP TABLE PAL_ALS_MODEL_FACTORS_TBL;
CREATE COLUMN TABLE PAL_ALS_MODEL_METADATA_TBL ("ROW_INDEX" INTEGER,
"METADATA_CONTENT" NVARCHAR(5000));
CREATE COLUMN TABLE PAL_ALS_MODEL_MAP_TBL ("ID" INTEGER, "MAP" NVARCHAR(1000));
CREATE COLUMN TABLE PAL_ALS_MODEL_FACTORS_TBL ("FACTOR_ID" INTEGER, "FACTOR"
DOUBLE);
CALL _SYS_AFL.PAL_ALS (PAL_ALS_DATA_TBL,
#PAL_PARAMETER_TBL, PAL_ALS_MODEL_METADATA_TBL, PAL_ALS_MODEL_MAP_TBL,
PAL_ALS_MODEL_FACTORS_TBL, ?, ?, ?) WITH OVERVIEW;
```

### Expected Result

MODEL_METADATA:

| | ROW_INDEX | METADATA_CONTENT |
|---|---|---|
| 1 | 0 | {"alpha":2.0,"exit_interval":5,"exit_threshold":0.0,"factor_number":1,"impli... |

MODEL_MAP:

| | ID | MAP |
|---|---|---|
| 1 | 0 | A |
| 2 | 1 | B |
| 3 | 2 | C |
| 4 | 3 | D |
| 5 | 4 | E |
| 6 | 5 | Movie1 |
| 7 | 6 | Movie2 |
| 8 | 7 | Movie4 |
| 9 | 8 | Movie5 |
| 10 | 9 | Movie6 |
| 11 | 10 | Movie8 |
| 12 | 11 | Bad_Movie |
| 13 | 12 | Movie3 |
| 14 | 13 | Movie7 |

MODEL_FACTORS:

| | FACTOR_ID | FACTOR |
|---|---|---|
| 1 | 0 | 0.2384230384908289 |
| 2 | 1 | 0.24015421614375218 |
| 3 | 2 | 0.2426617052284789 |
| 4 | 3 | 0.23888039208056958 |
| 5 | 4 | 0.24354848312689506 |
| 6 | 5 | 4.0494208947242125 |
| 7 | 6 | 4.04006734612655 |
| 8 | 7 | 4.152028767653444 |
| 9 | 8 | 4.0421564818135804 |
| 10 | 9 | 4.0447888749591705 |
| 11 | 10 | 4.153960017314627 |
| 12 | 11 | 3.995866744141494 |
| 13 | 12 | 3.860536614247436 |
| 14 | 13 | 4.017111070499886 |

ITERATION_INFORMATION:

| | ITERATION | COST | RMSE |
|---|---|---|---|
| 1 | 20 | 0.1736424840683281 | 0.412760358163604 |

STATISTICS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | timeout | FALSE |
| 2 | TEST_RMSE | 0.677942, 0.31277, 0.737421, 0.682409, 0.708985 |
| 3 | TEST_RMS... | 0.623906 |
| 4 | TEST_RMS... | 0.030818 |
| 5 | EVAL_RES... | {"candidates":[{"TEST_RMSE":[[0.75697065678... |
| 6 | EVAL_RES... | SE":[[0.7136335950621910,0.37566812942271... |
| 7 | EVAL_RES... | 871399,1.829725333113362,1.8960904395407... |
| 8 | EVAL_RES... | ,2.1585115427440]],"TEST_RMSE.MEAN":1.924... |
| 9 | EVAL_RES... | 063,"TEST_RMSE.VAR":0.09701032029199523,"... |

OPTIMAL_PARAMETER:

| | PARAM_NAME | INT_VALUE | DOUBLE_VALUE | STRING_VALUE |
|---|---|---|---|---|
| 1 | ALPHA | ? | 2 | ? |
| 2 | REGULARIZATION | ? | 0.001 | ? |
| 3 | FACTOR_NUMBER | 1 | ? | ? |

## _SYS_AFL.PAL_ALS_PREDICT

This procedure predicts users feedback against items based on the recommender model trained by PAL_ALS.

**Overview of All Tables**

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 4 | `<MODEL_METADATA table>` | IN |
| 5 | `<MODEL_MAP table>` | IN |
| 6 | `<MODEL_FACTORS table>` | IN |

| Position | Table Name | Parameter Type |
|---|---|---|
| 7 | `<PARAMETER table>` | IN |
| 8 | `<RESULT table>` | OUT |

### Input Tables

| Table | Column | Reference | Data Type | Description |
|---|---|---|---|---|
| DATA | 1st column | ID | INTEGER | ID |
| | 2nd column | User | INTEGER, VARCHAR, or NVARCHAR | User name or user ID |
| | 3rd column | ITEM | INTEGER, VARCHAR, or NVARCHAR | Item name or item ID |
| MODEL_METADATA | 1st column | | INTEGER | ID |
| | 2nd column | | NVARCHAR | Model metadata content |
| MODEL_MAP | 1st column | | INTEGER | ID |
| | 2nd column | | NVARCHAR | Map |
| MODEL_FACTORS | 1st column | | INTEGER | Factor ID |
| | 2nd column | | DOUBLE | Factors |

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

The following parameter is optional. If it is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| THREAD_RATIO | DOUBLE | 0 | Specifies the upper limit of thread usage in proportion of current available threads. The valid range of the value is [0,1], 0 indicates that a single thread is used and 1 means all the current available threads are used if necessary. Other values are ignored and the procedure determines the number of threads to use heuristically. |

### Output Table

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| RESULT | 1st column | ID data type | ID column name | ID |
| | 2nd column | USER data type | USER column name | User name or user ID |
| | 3rd column | ITEM data type | ITEM column name | Item name or item ID |
| | 4th column | DOUBLE | PREDICTION | Predicted feedback of user to item |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

**Example: Use the ALS model trained in Example 1 of _SYS_AFL.PAL_ALS section**

```
SET SCHEMA "DM_PAL";
DROP TABLE PAL_ALS_PREDICT_DATA_TBL;
CREATE COLUMN TABLE PAL_ALS_PREDICT_DATA_TBL ("ID" INTEGER, "USER"
NVARCHAR(100), "MOVIE" NVARCHAR(100));
INSERT INTO PAL_ALS_PREDICT_DATA_TBL VALUES (1, 'A', 'Movie3');
INSERT INTO PAL_ALS_PREDICT_DATA_TBL VALUES (2, 'A', 'Movie7');
INSERT INTO PAL_ALS_PREDICT_DATA_TBL VALUES (3, 'B', 'Movie1');
INSERT INTO PAL_ALS_PREDICT_DATA_TBL VALUES (4, 'B', 'Movie6');
INSERT INTO PAL_ALS_PREDICT_DATA_TBL VALUES (5, 'C', 'Movie3');
INSERT INTO PAL_ALS_PREDICT_DATA_TBL VALUES (6, 'D', 'Movie2');
INSERT INTO PAL_ALS_PREDICT_DATA_TBL VALUES (7, 'D', 'Movie5');
INSERT INTO PAL_ALS_PREDICT_DATA_TBL VALUES (8, 'E', 'Bad_Movie');
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL ("PARAM_NAME"
VARCHAR(256), "INT_VALUE" INTEGER, "DOUBLE_VALUE" DOUBLE, "STRING_VALUE"
VARCHAR(1000));
INSERT INTO #PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 1, NULL);
```

```
CALL _SYS_AFL.PAL_ALS_PREDICT (PAL_ALS_PREDICT_DATA_TBL,
PAL_ALS_MODEL_METADATA_TBL, PAL_ALS_MODEL_MAP_TBL, PAL_ALS_MODEL_FACTORS_TBL,
#PAL_PARAMETER_TBL, ?);
```

**Expected Result**

RESULT:

| | ID | USER | MOVIE | PREDICTION |
|---|---|---|---|---|
| 1 | 1 | A | Movie3 | 3.86874732317046 |
| 2 | 2 | A | Movie7 | 2.870243285874576 |
| 3 | 3 | B | Movie1 | 5.78755867891785 |
| 4 | 4 | B | Movie6 | 5.837218318171589 |
| 5 | 5 | C | Movie3 | 3.32357522081635 |
| 6 | 6 | D | Movie2 | 4.156371971675182 |
| 7 | 7 | D | Movie5 | 4.325851021643377 |
| 8 | 8 | E | Bad_Movie | 2.545806629137357 |

## 5.11.2 Factorized Polynomial Regression Models

Recommender system in PAL supports the Factorized Polynomial Regression Models or Factorization Machines approach.

Factorization approach, for example matrix factorization, provides high accuracy in several important prediction problems, including recommender systems. In its basic form, matrix factorization characterizes both users and items by vectors of latent factors inferred from user-item rating patterns and high correspondence between user and item factors will lead to a recommendation. Factorization machines approach for recommender system is more general than common factorization approaches as it can characterize latent factors not only for users and items themselves, but also for side features related to them by making use of additional information, and therefore, makes the predictions more accurate. PAL supports three kinds of additional side information besides the user-item ratings:

- Global side features which are related to each specific user-item rating/transaction. For example, location or time of a movie was rated by or lent to a user;
- User side features which are related to each user, such as gender, age, education, etc.;
- Item side features which are related to each item, such as genre of a movie.

Categorical features are transformed into a vector of 0s and 1s. In the model of factorization machines, each user, item and side feature is represented by a weight and a vector of latent factors. These weights and factors are trained based on user-item rating patterns and all other side information, and afterwards be used in the prediction of unknown ratings. The predicted rating $\hat{r}_{u,v}$ of a user u on an item v is given by

$$\hat{r}_{u,v} = w_0 + \sum_{i \in \{u,v,S\}} x_i w_i + \frac{1}{2} \sum_{i,j \in \{u,v,S\}, i \neq j} x_i x_j f_i^T f_j ,$$

where $w_0$ represents the global bias, S is the set of all side features, $x_i, x_j$ are the values of corresponding side features with $x_u \equiv x_v \equiv 1$, and $w_i$ and $f_i, f_j$ are, respectively, the weight and the factor vectors of user, item or

side features. Weights of users, items or side features in the model represent biases associated with them. For example, a user who tends to give higher ratings than average will get a higher weight, and an item lower rated by most users tends to get a lower weight. Meanwhile, factors of users, items and side features give representation of effects of the interactions between them by taking inner product of corresponding factor vectors. As can be seen from the formula, biases of side features and their interactions with user, item and within each other are also accounted to the prediction.

Parameters, i.e. weights and factors, of the model are trained by solving the optimization problem

$$\mathrm{argmin}_{w_i,\ f_i} \frac{1}{|D|}\left( \sum_{u,\ v,\ r_{u,v}\in D} l(\hat{r}_{u,\ v},\ r_{u,v}) + \lambda_w \sum_{i\in\{u,\ v,\ S\}} w_i^2 + \lambda_f \sum_{i\in\{u,\ v,\ S\}} \|f_i\|_2^2 \right),$$

where $D$ is the set of all the observed user-item ratings, $l(x,y)=(x-y)^2$ is the loss function, $r_{u,v}$ is the observed rating, and $\lambda_w$,$\lambda_f$ are the regularization amount of weights and factors, respectively.

In PAL, the above optimization problem is solved by stochastic gradient descent (SGD) method. SGD iterates over cases $(u,v,r_{u,v})\in D$ and performs updates on the model parameters for each case:

$$\partial_\theta \Phi(\theta) = \frac{\partial}{\partial\theta} l(\hat{r}_{u,\ v},\ r_{u,v}) + 2\lambda_\theta \theta,$$

$$\theta \leftarrow \theta - \eta \partial_\theta \Phi(\theta),\ \ \theta\in\Theta(u,v,r_{u,v}),$$

where $\partial_\theta\ \Phi(\theta)$ represents the derivative of $\theta$, $\theta(u,v,r_{u,v})$ is the set of all parameters related to the current case $(u,v,r_{u,v})$, i.e. weights and factors of user $u$, item $v$, and all the side features related. $\eta$ is the learning rate or step size and $\lambda_\theta$ is the corresponding regularization amount of $\theta$.

Besides SGD, several variations of it are also supported.

- Momentum: Momentum method adds a fraction $\gamma$ of the last step's update to the current step's update, which intends to accelerate SGD in the relevant direction and dampen oscillations:

  $$\rho_t \leftarrow \gamma\rho_{t-1} + \eta\partial_\theta\Phi(\theta),$$
  $$\theta \leftarrow \theta - \rho_t,$$

  where $t$ is the current step. The default value of the momentum $\gamma$ in PAL is 0.9.
- Nesterov accelerated gradient (NAG): NAG is a slightly different version of the momentum method. Instead of using the current $\theta$, NAG uses the approximate future position of $\theta$ to calculate the derivative to give the momentum term some kind of prescience:

  $$\rho_t \leftarrow \gamma\rho_{t-1} + \eta\partial_\theta\Phi(\theta - \gamma\rho_{t-1}),$$
  $$\theta \leftarrow \theta - \rho_t.$$

  The default value of the momentum $\gamma$ in PAL is also 0.9.
- Adagrad: Adagrad is a method that adapts the learning rate $\eta$ to each parameter, performing larger updates for infrequent and smaller updates for frequent parameters, and hence is well suited for dealing with sparse data. It modifies the learning rate $\eta$ at each step $t$ for each parameter $\theta$ based on the sum of squares of its past derivatives:

  $$G_{\theta,t} \leftarrow G_{\theta,t-1} + \left(\partial_\theta\Phi(\theta)\right)^2,$$
  $$\theta \leftarrow \theta - \frac{\eta}{\sqrt{G_{\theta,t}}}\partial_\theta\Phi(\theta),$$

  where $G_{\theta,0}$ is set to 1 to avoid large value of $G_{\theta,t}^{-\frac{1}{2}}$. The main benefit of Adagrad is that it is insensitive to the learning rate $\eta$, and therefore, saves the effort for tuning it.

For parallelizing SGD, PAL implements Hogwild! algorithm. Under the assumption that componentwise addition operation is atomic, Hogwild! gives a lock-free update scheme for SGD, which allows processors access to shared memory with the possibility of overwriting each other's work. It achieves a fairly speedup with parallelization when the associated optimization problem is sparse.

- Null handle of PAL_FRM: data with both null user ID and item ID, and data with null rating will be discarded; null value of numerical feature will be replaced by mean; null value of categorical feature will not be used (transformed to an all zero vector).
- Null handle of PAL_FRM_PREDICT: data with null or invalid (not included in the trained model) user ID or item ID will be proceeded using only the information of the given item or user; data with both null user ID and item ID will not be proceeded; null value of numerical feature will be replaced by mean; null value of categorical feature will not be used (transformed to an all zero vector).

## Prerequisites

None

## _SYS_AFL.PAL_FRM

This function trains a recommender model based on factorized polynomial regression models/factorization machines.

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<USER_INFORMATION table>` | IN |
| 3 | `<ITEM_INFORMATION table>` | IN |
| 4 | `<PARAMETER table>` | IN |
| 5 | `<MODEL_METADATA table>` | OUT |
| 6 | `<MODEL table>` | OUT |
| 7 | `<MODEL_FACTORS table>` | OUT |
| 8 | `<ITERATION_INFORMATION table>` | OUT |
| 9 | `<STATISTICS table>` | OUT |
| 10 | `<OPTIMAL_PARAMETER table>` | OUT |

### Input Tables

| Table | Column | Data Type | Description |
|---|---|---|---|
| DATA | 1st column | INTEGER, VARCHAR, or NVARCHAR | User name or user ID |
| | 2nd column | INTEGER, VARCHAR, or NVARCHAR | Item name or item ID |
| | Side feature columns | INTEGER, DOUBLE, DECIMAL(p, s), VARCHAR, or NVARCHAR | Global side features. Both continuous and categorical features are supported. By default, features of INTEGER and DOUBLE types are treated as continuous, and features of STRING type are treated as categorical.<br><br>Categorical feature with multiple tags is supported. Each tag should be separated by ",". For example, to represent multiple genres of a movie, you can use "Action,Adventure,Drama,Thriller". Space is not neglected. For example, "Action", " Action" and "Action " are treated as different tags.<br><br>Leave these columns empty if no side information is provided. |
| | Last column | INTEGER or DOUBLE | User's feedback (rating) of the item |
| USER_INFORMATION | 1st column | INTEGER, VARCHAR, or NVARCHAR | User name or user ID |
| | Side feature columns | INTEGER, DOUBLE, DECIMAL(p, s), VARCHAR, or NVARCHAR | User side features.<br><br>Leave these columns empty if no side information is provided. |
| ITEM_INFORMATION | 1st column | INTEGER, VARCHAR, or NVARCHAR | Item name or item ID |

| Table | Column | Data Type | Description |
|---|---|---|---|
| | Side feature columns | INTEGER, DOUBLE, DECIMAL(p, s), VARCHAR, or NVARCHAR | Item side features.<br>Leave these columns empty if no side information is provided. |

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| HAS_ID | INTEGER | 0 | • HAS_ID = 0 indicates there is no ID column in the data table. The algorithm will use all the columns to do the training and return the results.<br>• HAS_ID = 1 indicates that the FIRST column of the data table is ID column. The algorithm will ignore the FIRST column to do the training and return the results. | |
| CATEGORICAL_VARIABLE | VARCHAR | No default value | Indicates whether or not a column data is actually corresponding to a category variable even the data type of this column is INTEGER. By default, 'VARCHAR' or 'NVARCHAR' is category variable, and 'INTEGER' or 'DOUBLE' is continuous variable. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| USER_CATEGORICAL_VARIABLE | VARCHAR | No default value | Name of user side feature columns that should be treated as categorical. | |
| ITEM_CATEGORICAL_VARIABLE | VARCHAR | No default value | Name of item side feature columns that should be treated as categorical. | |
| EXCLUDED_FEATURE | VARCHAR | No default value | Specifies the feature(s) to be excluded from training. | |
| USER_EXCLUDED_FEATURE | VARCHAR | No default value | Specifies the user side feature(s) to be excluded from training. | |
| ITEM_EXCLUDED_FEATURE | VARCHAR | No default value | Specifies the item side feature(s) to be excluded from training. | |
| METHOD | INTEGER | 0 | Specifies the method<br>• 0: SGD<br>• 1: MOMENTUM<br>• 2: NAG<br>• 3: ADAGRAD | |
| FACTOR_NUMBER | INTEGER | 8 | Length of factor vectors $f_i$ in the model. | |
| INITIALIZATION | DOUBLE | 1e-2 | Variance of the normal distribution used to initialize the model parameters. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| SEED | INTEGER | 0 | Specifies the seed for random number generator.<br>• 0: Uses the current time as seed<br>• Others: Uses the specified value as seed<br>Note that due to the inherently randomicity of parallel SGD, models of different trainings might be different even with the same SEED. | |
| LEARNING_RATE | DOUBLE | 0 | Specifies the learning rate $\eta$.<br>If you set it to the default value 0, the function chooses the step size automatically, based on a small part of the dataset. | |
| LINEAR_REGULARIZATION | DOUBLE | 1e-10 | L2 regularization of the weights $\lambda_w$. | |
| REGULARIZATION | DOUBLE | 1e-8 | L2 regularization of the factors $\lambda_f$. | |
| MAX_ITERATION | INTEGER | 50 | Maximum number of iterations. | |
| SGD_EXIT_THRESHOLD | DOUBLE | 1e-5 | Exit threshold $\varepsilon$.<br>The algorithm exits when the cost function has not decreased more than $\varepsilon$ in SGD_EXIT_INTERVAL steps. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| SGD_EXIT_INTERVAL | INTEGER | 5 | The algorithm exits when the cost function has not decreased more than SGD_EXIT_THRESHOLD in SGD_EXIT_INTERVAL steps. | |
| THREAD_RATIO | DOUBLE | 0 | Specify the upper limit of thread usage in proportion of current available threads. The valid range of the value is [0,1], 0 indicates that single thread shall be used and 1 means all the current available threads will be used if necessary. Other value input will be ignored and the procedure will determine the number of threads to use heuristically. It is recommended to use this parameter instead of THREAD_NUMBER. | |
| MOMENTUM | DOUBLE | 0.90 | Momentum $\gamma$ in method MOMENTUM or NAG. | Only valid when method MOMENTUM or NAG is specified. |

### Model Evaluation and Parameter Selection

The following parameters are only valid when model evaluation or parameter selection is enabled.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| RESAMPLING_METHOD | NVARCHAR | No default value | Specifies the resampling method for model evaluation or parameter selection.<br>• cv<br>• cv_sha<br>• cv_hyperband<br>• bootstrap<br>• bootstrap_sha<br>• bootstrap_hyperband<br>If no value is specified for this parameter, neither model evaluation nor parameter selection is activated. | Options like *_sha and *_hyperband can only be used in parameter selection, can not be used in model evaluation. |
| EVALUATION_METRIC | NVARCHAR | No default value | Specifies the evaluation metric for model evaluation or parameter selection.<br>• RMSE | |
| FOLD_NUM | INTEGER | 1 | Specifies the fold number for the cross validation method. | Mandatory and valid only when RESAMPLING_METHOD is set to cv*. |
| REPEAT_TIMES | INTEGER | 1 | Specifies the number of repeat times for resampling. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| PARAM_SEARCH_STRATEGY | NVARCHAR | No default value | Specifies the method to activate parameter selection.<br>• grid<br>• random | When *_sha is specified in RESAMPLING_METHOD, PARAM_SEARCH_STRATEGY is mandatory. When *_hyperband is specified in RESAMPLING_METHOD, PARAM_SEARCH_STRATEGY is set to random whatever other value is provided. Otherwise if this parameter is not set, only model evaluation is activated. |
| RANDOM_SEARCH_TIMES | INTEGER | No default value | Specifies the number of times to randomly select candidate parameters for selection. | Mandatory and valid when PARAM_SEARCH_STRATEGY is set to random. |
| REDUCTION_RATE | DOUBLE | 3.0 | Specifies reduction rate in SHA or hyperband method. | Only valid when RESAMPLING_METHOD is specified as *_sha or *_hyperband. |
| MIN_RESOURCE_RATE | DOUBLE | 0.0 | Specifies the minimum resource rate that should be used in SHA or hyperband iteration. | Only valid when RESAMPLING_METHOD is specified as *_sha or *_hyperband. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| AGGRESSIVE_ELIMINATION | INTEGER | 0 | Specifies whether to apply aggressive elimination while using SHA method. Aggressive elimination happens when the data size and parameters size to be searched does not match and there are still bunch of parameters to be searched while data size reaches its upper limits. If aggressive elimination is applied, lower bound of limit of data size will be used multiple times first to reduce number of parameters.<br>• 0: does not apply aggressive elimination<br>• 1: applies aggressive elimination | Only valid when RESAMPLING_METHOD is specified as *_sha. |
| SEED | INTEGER | 0 | Specifies the seed for random generation. Uses system time when 0 is specified. | |
| TIMEOUT | INTEGER | 0 | Specifies maximum running time for model evaluation or parameter selection, in seconds. No timeout when 0 is specified. | |
| PROGRESS_INDICATOR_ID | NVARCHAR | No default value | Sets an ID of progress indicator for model evaluation or parameter selection. No progress indicator is active if no value is provided. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| FACTOR_NUMBER_VALUES | NVARCHAR | No default value | Specifies values of FACTOR_NUMBER to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| FACTOR_NUMBER_RANGE | NVARCHAR | No default value | Specifies the range of FACTOR_NUMBER to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| LINEAR_REGULARIZATION_VALUES | NVARCHAR | No default value | Specifies values of LINEAR_REGULARIZATION to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| LINEAR_REGULARIZATION_RANGE | NVARCHAR | No default value | Specifies the range of LINEAR_REGULARIZATION to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| REGULARIZATION_VALUES | NVARCHAR | No default value | Specifies values of REGULARIZATION to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| REGULARIZATION_RANGE | NVARCHAR | No default value | Specifies the range of REGULARIZATION to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| MOMENTUM_VALUES | NVARCHAR | No default value | Specifies values of MOMENTUM to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified and METHOD is MOMENTUM or NAG. |
| MOMENTUM_RANGE | NVARCHAR | No default value | Specifies the range of MOMENTUM to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified and METHOD is MOMENTUM or NAG. |

### Output Tables

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| MODEL_METADATA | 1st column | INTEGER | ROW_INDEX | ID |
| | 2nd column | NVARCHAR(5000) | METADATA_CONTENT | Model metadata content |
| MODEL | 1st column | INTEGER | ID | ID |
| | 2nd column | NVARCHAR(1000) | MAP | Map |

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| | 3rd column | DOUBLE | WEIGHT | Weight |
| MODEL_FACTORS | 1st column | INTEGER | FACTOR_ID | Factor ID |
| | 2nd column | DOUBLE | FACTOR | Factors |
| ITERATION_INFORMATION | 1st column | NVARCHAR(100) | ITERATION | Step |
| | 2nd column | NVARCHAR(100) | COST | Cost function value of corresponding step |
| | 3rd column | NVARCHAR(100) | RMSE | Root mean square error of corresponding step |
| | 4th column | NVARCHAR(100) | STEP_SIZE | Step size of corresponding step. Note that step size of method ADAGRAD will be constant. |
| STATISTICS | 1st column | NVARCHAR(256) | STAT_NAME | Statistic names. |
| | 2nd column | NVARCHAR(1000) | STAT_VALUE | Statistic values. |
| OPTIMAL_PARAMETER | 1st column | NVARCHAR(256) | PARAM_NAME | Optimal parameters selected. |
| | 2nd column | INTEGER | INT_VALUE | |
| | 3rd column | DOUBLE | DOUBLE_VALUE | |
| | 4th column | NVARCHAR(1000) | STRING_VALUE | |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

#### Example 1: Train an FRM model without parameter selection

```
SET SCHEMA "DM_PAL";
DROP TABLE PAL_FRM_DATA_TBL;
CREATE COLUMN TABLE PAL_FRM_DATA_TBL ("ID" INTEGER, "USER" NVARCHAR(100),
"MOVIE" NVARCHAR(100), "TIMESTAMP" INTEGER, "RATING" DOUBLE); -- DATA table has
an ID column, we will set HAS_ID = 1 to ignore it
INSERT INTO PAL_FRM_DATA_TBL VALUES (1, 'A', 'Movie1', 3, 4.8);
INSERT INTO PAL_FRM_DATA_TBL VALUES (2, 'A', 'Movie2', 3, 4.0);
INSERT INTO PAL_FRM_DATA_TBL VALUES (3, 'A', 'Movie4', 1, 4.0);
INSERT INTO PAL_FRM_DATA_TBL VALUES (4, 'A', 'Movie5', 2, 4.0);
INSERT INTO PAL_FRM_DATA_TBL VALUES (5, 'A', 'Movie6', 3, 4.8);
INSERT INTO PAL_FRM_DATA_TBL VALUES (6, 'A', 'Movie8', 2, 3.8);
```

```
INSERT INTO PAL_FRM_DATA_TBL VALUES (7, 'A', 'Bad_Movie', 1, 2.5);
INSERT INTO PAL_FRM_DATA_TBL VALUES (8, 'B', 'Movie2', 3, 4.8);
INSERT INTO PAL_FRM_DATA_TBL VALUES (9, 'B', 'Movie3', 2, 4.8);
INSERT INTO PAL_FRM_DATA_TBL VALUES (10, 'B', 'Movie4', 2, 5.0);
INSERT INTO PAL_FRM_DATA_TBL VALUES (11, 'B', 'Movie5', 4, 5.0);
INSERT INTO PAL_FRM_DATA_TBL VALUES (12, 'B', 'Movie7', 1, 3.5);
INSERT INTO PAL_FRM_DATA_TBL VALUES (13, 'B', 'Movie8', 2, 4.8);
INSERT INTO PAL_FRM_DATA_TBL VALUES (14, 'B', 'Bad_Movie', 3, 2.8);
INSERT INTO PAL_FRM_DATA_TBL VALUES (15, 'C', 'Movie1', 2, 4.1);
INSERT INTO PAL_FRM_DATA_TBL VALUES (16, 'C', 'Movie2', 4, 4.2);
INSERT INTO PAL_FRM_DATA_TBL VALUES (17, 'C', 'Movie4', 3, 4.2);
INSERT INTO PAL_FRM_DATA_TBL VALUES (18, 'C', 'Movie5', 1, 4.0);
INSERT INTO PAL_FRM_DATA_TBL VALUES (19, 'C', 'Movie6', 4, 4.2);
INSERT INTO PAL_FRM_DATA_TBL VALUES (20, 'C', 'Movie7', 3, 3.2);
INSERT INTO PAL_FRM_DATA_TBL VALUES (21, 'C', 'Movie8', 1, 3.0);
INSERT INTO PAL_FRM_DATA_TBL VALUES (22, 'C', 'Bad_Movie', 2, 2.5);
INSERT INTO PAL_FRM_DATA_TBL VALUES (23, 'D', 'Movie1', 3, 4.5);
INSERT INTO PAL_FRM_DATA_TBL VALUES (24, 'D', 'Movie3', 2, 3.5);
INSERT INTO PAL_FRM_DATA_TBL VALUES (25, 'D', 'Movie4', 2, 4.5);
INSERT INTO PAL_FRM_DATA_TBL VALUES (26, 'D', 'Movie6', 2, 3.9);
INSERT INTO PAL_FRM_DATA_TBL VALUES (27, 'D', 'Movie7', 4, 3.5);
INSERT INTO PAL_FRM_DATA_TBL VALUES (28, 'D', 'Movie8', 3, 3.5);
INSERT INTO PAL_FRM_DATA_TBL VALUES (29, 'D', 'Bad_Movie', 3, 2.5);
INSERT INTO PAL_FRM_DATA_TBL VALUES (30, 'E', 'Movie1', 2, 4.5);
INSERT INTO PAL_FRM_DATA_TBL VALUES (31, 'E', 'Movie2', 2, 4.0);
INSERT INTO PAL_FRM_DATA_TBL VALUES (32, 'E', 'Movie3', 2, 3.5);
INSERT INTO PAL_FRM_DATA_TBL VALUES (33, 'E', 'Movie4', 4, 4.5);
INSERT INTO PAL_FRM_DATA_TBL VALUES (34, 'E', 'Movie5', 3, 4.5);
INSERT INTO PAL_FRM_DATA_TBL VALUES (35, 'E', 'Movie6', 2, 4.2);
INSERT INTO PAL_FRM_DATA_TBL VALUES (36, 'E', 'Movie7', 4, 3.5);
INSERT INTO PAL_FRM_DATA_TBL VALUES (37, 'E', 'Movie8', 3, 3.5);
DROP TABLE PAL_FRM_USER_INFORMATION_TBL;
CREATE COLUMN TABLE PAL_FRM_USER_INFORMATION_TBL ("USER" NVARCHAR(100),
"USER_SIDE_FEATURE" DOUBLE);
-- There is no side information for user provided. --
DROP TABLE PAL_FRM_ITEM_INFORMATION_TBL;
CREATE COLUMN TABLE PAL_FRM_ITEM_INFORMATION_TBL ("MOVIE" VARCHAR(100), "GENRES"
NVARCHAR(100));
INSERT INTO PAL_FRM_ITEM_INFORMATION_TBL VALUES ('Movie1', 'Sci-Fi');
INSERT INTO PAL_FRM_ITEM_INFORMATION_TBL VALUES ('Movie2', 'Drama,Romance');
INSERT INTO PAL_FRM_ITEM_INFORMATION_TBL VALUES ('Movie3', 'Drama,Sci-Fi');
INSERT INTO PAL_FRM_ITEM_INFORMATION_TBL VALUES ('Movie4', 'Crime,Drama');
INSERT INTO PAL_FRM_ITEM_INFORMATION_TBL VALUES ('Movie5', 'Crime,Drama');
INSERT INTO PAL_FRM_ITEM_INFORMATION_TBL VALUES ('Movie6', 'Sci-Fi');
INSERT INTO PAL_FRM_ITEM_INFORMATION_TBL VALUES ('Movie7', 'Crime,Drama');
INSERT INTO PAL_FRM_ITEM_INFORMATION_TBL VALUES ('Movie8', 'Sci-Fi,Thriller');
INSERT INTO PAL_FRM_ITEM_INFORMATION_TBL VALUES ('Bad_Movie',
'Romance,Thriller');
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL ("PARAM_NAME"
VARCHAR(256), "INT_VALUE" INTEGER, "DOUBLE_VALUE" DOUBLE, "STRING_VALUE"
VARCHAR(1000));
INSERT INTO #PAL_PARAMETER_TBL VALUES ('HAS_ID', 1, NULL, NULL); -- ignore the
first column in DATA table
INSERT INTO #PAL_PARAMETER_TBL VALUES ('CATEGORICAL_VARIABLE', NULL, NULL,
'TIMESTAMP'); -- column TIMESTAMP is categorical
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FACTOR_NUMBER', 2, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('METHOD', 3, NULL, NULL);-- use ADAGRAD
method
INSERT INTO #PAL_PARAMETER_TBL VALUES ('LEARNING_RATE', NULL, 0, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MAX_ITERATION', 100, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 0.5, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SEED', 1, NULL, NULL);
DROP TABLE PAL_FRM_MODEL_METADATA_TBL;
DROP TABLE PAL_FRM_MODEL_TBL;
DROP TABLE PAL_FRM_MODEL_FACTORS_TBL;
CREATE COLUMN TABLE PAL_FRM_MODEL_METADATA_TBL ("ROW_INDEX" INTEGER,
"METADATA_CONTENT" NVARCHAR(5000));
```

```
CREATE COLUMN TABLE PAL_FRM_MODEL_TBL ("ID" INTEGER, "MAP" NVARCHAR(1000),
"WEIGHT" DOUBLE);
CREATE COLUMN TABLE PAL_FRM_MODEL_FACTORS_TBL ("FACTOR_ID" INTEGER, "FACTOR"
DOUBLE);
CALL _SYS_AFL.PAL_FRM (PAL_FRM_DATA_TBL, PAL_FRM_USER_INFORMATION_TBL,
PAL_FRM_ITEM_INFORMATION_TBL, #PAL_PARAMETER_TBL, PAL_FRM_MODEL_METADATA_TBL,
PAL_FRM_MODEL_TBL, PAL_FRM_MODEL_FACTORS_TBL, ?, ?, ?) WITH OVERVIEW;
```

**Expected Result**

Results of different runs might be slightly different due to the inherently randomicity of parallel SGD.

MODEL_METADAT:

| | ROW_INDEX | METADATA_CONTENT |
|---|---|---|
| 1 | 0 | {"bias":3.948648648648649,"category_col":[2],"eta":0,"factor_num":2,"int... |

MODEL:

| | ID | MAP | WEIGHT |
|---|---|---|---|
| 1 | 0 | A | -0.13763430247869374 |
| 2 | 1 | B | 0.5574204355766743 |
| 3 | 2 | C | -0.37020331731350997 |
| 4 | 3 | D | -0.26005312669855624 |
| 5 | 4 | E | -0.12159787428810553 |
| 6 | 5 | Bad_Movie | -0.8049120303192876 |
| 7 | 6 | Movie1 | 0.461275903315181 |
| 8 | 7 | Movie2 | 0.45301244099532306 |
| 9 | 8 | Movie3 | -0.2803383479014807 |
| 10 | 9 | Movie4 | 0.44400089760593126 |
| 11 | 10 | Movie5 | 0.31074765603969773 |
| 12 | 11 | Movie6 | 0.13668774832600106 |
| 13 | 12 | Movie7 | -0.5053672259103076 |
| 14 | 13 | Movie8 | 0.09508129004602986 |
| 15 | 14 | 1 | -0.30186988931571185 |
| 16 | 15 | 2 | -0.08504168106972641 |
| 17 | 16 | 3 | -0.016862103716358... |
| 18 | 17 | 4 | 0.24522261594084796 |
| 19 | 18 | Crime | 0.07818841560312932 |
| 20 | 19 | Drama | 0.07405157190327478 |
| 21 | 20 | Romance | -0.40551697306176876 |
| 22 | 21 | Sci-Fi | 0.3367943612360086 |
| 23 | 22 | Thriller | -0.5755688756754188 |

MODEL_FACTORS:

| | FACTOR_ID | FACTOR |
|---|---|---|
| 1 | 0 | 0.05477885874060784 |
| 2 | 1 | 0.07220889424142755 |
| 3 | 2 | -0.2299513381869753 |
| 4 | 3 | 0.5702452241326292 |
| 5 | 4 | 0.024124206931757... |
| 6 | 5 | -0.3933139093281848 |
| 7 | 6 | -0.1690539644662948 |
| 8 | 7 | -0.170904120483974... |
| 9 | 8 | 0.32562786127750154 |
| 10 | 9 | -0.246521115270211... |
| 11 | 10 | 0.4020394878905077 |
| 12 | 11 | -0.5131546101754966 |
| 13 | 12 | 0.12842274971453918 |
| 14 | 13 | -0.038203515813799... |
| 15 | 14 | -0.4008427609800718 |
| 16 | 15 | 0.07731347003403424 |
| 17 | 16 | -0.044038364486803... |
| 18 | 17 | 0.5227930086776066 |
| 19 | 18 | -0.260677214541642... |
| 20 | 19 | -0.3614701265755306 |
| 21 | 20 | 0.38866582926386467 |
| 22 | 21 | -0.074480336521329... |
| 23 | 22 | 0.47326970176586314 |
| 24 | 23 | 0.2314284644399839 |
| 25 | 24 | 0.06701927003860657 |
| 26 | 25 | -0.080579844641489... |
| 27 | 26 | -0.5707015020742643 |

ITERATION_INFORMATION:

| | ITERATION | COST | RMSE | STEP_SIZE |
|---|---|---|---|---|
| 1 | 1 | 0.33136013815153637 | 0.5756388956013739 | 0.390625 |
| 2 | 2 | 0.1440239060214606 | 0.3795048168095933 | 0.390625 |
| 3 | 3 | 0.08028135673263748 | 0.28333964881851... | 0.390625 |
| 4 | 4 | 0.05571876352809555 | 0.23604822206978... | 0.390625 |
| 5 | 5 | 0.06193700661164663 | 0.24887146531956... | 0.390625 |
| 6 | 6 | 0.0254860276012188 | 0.15964343721459... | 0.390625 |
| 7 | 7 | 0.019403190048837... | 0.13929533184864... | 0.390625 |
| 8 | 8 | 0.020288004412496... | 0.14243596378294... | 0.390625 |
| 9 | 9 | 0.015303964473146... | 0.12370919041247... | 0.390625 |
| 10 | 10 | 0.010700095418654... | 0.10344126206557... | 0.390625 |
| 11 | 11 | 0.015842777325362... | 0.12586809246178... | 0.390625 |
| 12 | 12 | 0.009805589959257... | 0.09902317534105... | 0.390625 |
| 13 | 13 | 0.008522286850806... | 0.09231622942389... | 0.390625 |
| 14 | 14 | 0.006163387567655... | 0.07850724028241... | 0.390625 |
| 15 | 15 | 0.004301666280382... | 0.065587083507804 | 0.390625 |
| 16 | 16 | 0.004515638573547... | 0.06719849579200... | 0.390625 |
| 17 | 17 | 0.002996572550397... | 0.05474095111948... | 0.390625 |
| 18 | 18 | 0.002391145082595... | 0.0488993278947703 | 0.390625 |
| 19 | 19 | 0.002613397438771... | 0.0511213907286093 | 0.390625 |
| 20 | 20 | 0.002413948150955... | 0.04913193759675... | 0.390625 |
| 21 | 21 | 0.001674416646266... | 0.04091962567926... | 0.390625 |
| 22 | 22 | 0.00138399852500095 | 0.03720211865789... | 0.390625 |
| 23 | 23 | 0.001036089764462... | 0.03218833432282... | 0.390625 |
| 24 | 24 | 0.001201134447348... | 0.03465737370952... | 0.390625 |
| 25 | 25 | 0.001319690986677... | 0.03632753890802... | 0.390625 |
| 26 | 26 | 0.000876837941446... | 0.02961143373337... | 0.390625 |
| 27 | 27 | 0.001273523437511 | 0.03568644746315 | 0.390625 |

STATISTICS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| | | |

OPTIMAL_PARAMETER:

| | PARAM_NAME | INT_VALUE | DOUBLE_VALUE | STRING_VALUE |
|---|---|---|---|---|
| | | | | |
| | | | | |

**Example 2: Train an FRM model with parameter selection**

```
SET SCHEMA "DM_PAL";
DROP TABLE PAL_FRM_DATA_TBL;
CREATE COLUMN TABLE PAL_FRM_DATA_TBL ("ID" INTEGER, "USER" NVARCHAR(100),
"MOVIE" NVARCHAR(100), "TIMESTAMP" INTEGER, "RATING" DOUBLE); -- DATA table has
an ID column, we will set HAS_ID = 1 to ignore it
INSERT INTO PAL_FRM_DATA_TBL VALUES (1, 'A', 'Movie1', 3, 4.8);
INSERT INTO PAL_FRM_DATA_TBL VALUES (2, 'A', 'Movie2', 3, 4.0);
INSERT INTO PAL_FRM_DATA_TBL VALUES (3, 'A', 'Movie4', 1, 4.0);
INSERT INTO PAL_FRM_DATA_TBL VALUES (4, 'A', 'Movie5', 2, 4.0);
INSERT INTO PAL_FRM_DATA_TBL VALUES (5, 'A', 'Movie6', 3, 4.8);
INSERT INTO PAL_FRM_DATA_TBL VALUES (6, 'A', 'Movie8', 2, 3.8);
INSERT INTO PAL_FRM_DATA_TBL VALUES (7, 'A', 'Bad_Movie', 1, 2.5);
INSERT INTO PAL_FRM_DATA_TBL VALUES (8, 'B', 'Movie2', 3, 4.8);
INSERT INTO PAL_FRM_DATA_TBL VALUES (9, 'B', 'Movie3', 2, 4.8);
INSERT INTO PAL_FRM_DATA_TBL VALUES (10, 'B', 'Movie4', 2, 5.0);
INSERT INTO PAL_FRM_DATA_TBL VALUES (11, 'B', 'Movie5', 4, 5.0);
INSERT INTO PAL_FRM_DATA_TBL VALUES (12, 'B', 'Movie7', 1, 3.5);
INSERT INTO PAL_FRM_DATA_TBL VALUES (13, 'B', 'Movie8', 2, 4.8);
INSERT INTO PAL_FRM_DATA_TBL VALUES (14, 'B', 'Bad_Movie', 3, 2.8);
INSERT INTO PAL_FRM_DATA_TBL VALUES (15, 'C', 'Movie1', 2, 4.1);
INSERT INTO PAL_FRM_DATA_TBL VALUES (16, 'C', 'Movie2', 4, 4.2);
INSERT INTO PAL_FRM_DATA_TBL VALUES (17, 'C', 'Movie4', 3, 4.2);
INSERT INTO PAL_FRM_DATA_TBL VALUES (18, 'C', 'Movie5', 1, 4.0);
INSERT INTO PAL_FRM_DATA_TBL VALUES (19, 'C', 'Movie6', 4, 4.2);
INSERT INTO PAL_FRM_DATA_TBL VALUES (20, 'C', 'Movie7', 3, 3.2);
INSERT INTO PAL_FRM_DATA_TBL VALUES (21, 'C', 'Movie8', 1, 3.0);
INSERT INTO PAL_FRM_DATA_TBL VALUES (22, 'C', 'Bad_Movie', 2, 2.5);
INSERT INTO PAL_FRM_DATA_TBL VALUES (23, 'D', 'Movie1', 3, 4.5);
INSERT INTO PAL_FRM_DATA_TBL VALUES (24, 'D', 'Movie3', 2, 3.5);
INSERT INTO PAL_FRM_DATA_TBL VALUES (25, 'D', 'Movie4', 2, 4.5);
INSERT INTO PAL_FRM_DATA_TBL VALUES (26, 'D', 'Movie6', 2, 3.9);
INSERT INTO PAL_FRM_DATA_TBL VALUES (27, 'D', 'Movie7', 4, 3.5);
INSERT INTO PAL_FRM_DATA_TBL VALUES (28, 'D', 'Movie8', 3, 3.5);
INSERT INTO PAL_FRM_DATA_TBL VALUES (29, 'D', 'Bad_Movie', 3, 2.5);
INSERT INTO PAL_FRM_DATA_TBL VALUES (30, 'E', 'Movie1', 2, 4.5);
INSERT INTO PAL_FRM_DATA_TBL VALUES (31, 'E', 'Movie2', 2, 4.0);
INSERT INTO PAL_FRM_DATA_TBL VALUES (32, 'E', 'Movie3', 2, 3.5);
INSERT INTO PAL_FRM_DATA_TBL VALUES (33, 'E', 'Movie4', 4, 4.5);
INSERT INTO PAL_FRM_DATA_TBL VALUES (34, 'E', 'Movie5', 3, 4.5);
INSERT INTO PAL_FRM_DATA_TBL VALUES (35, 'E', 'Movie6', 2, 4.2);
INSERT INTO PAL_FRM_DATA_TBL VALUES (36, 'E', 'Movie7', 4, 3.5);
INSERT INTO PAL_FRM_DATA_TBL VALUES (37, 'E', 'Movie8', 3, 3.5);
DROP TABLE PAL_FRM_USER_INFORMATION_TBL;
CREATE COLUMN TABLE PAL_FRM_USER_INFORMATION_TBL ("USER" NVARCHAR(100),
"USER_SIDE_FEATURE" DOUBLE);
-- There is no side information for user provided. --
DROP TABLE PAL_FRM_ITEM_INFORMATION_TBL;
CREATE COLUMN TABLE PAL_FRM_ITEM_INFORMATION_TBL ("MOVIE" VARCHAR(100), "GENRES"
NVARCHAR(100));
INSERT INTO PAL_FRM_ITEM_INFORMATION_TBL VALUES ('Movie1', 'Sci-Fi');
INSERT INTO PAL_FRM_ITEM_INFORMATION_TBL VALUES ('Movie2', 'Drama,Romance');
INSERT INTO PAL_FRM_ITEM_INFORMATION_TBL VALUES ('Movie3', 'Drama,Sci-Fi');
INSERT INTO PAL_FRM_ITEM_INFORMATION_TBL VALUES ('Movie4', 'Crime,Drama');
INSERT INTO PAL_FRM_ITEM_INFORMATION_TBL VALUES ('Movie5', 'Crime,Drama');
INSERT INTO PAL_FRM_ITEM_INFORMATION_TBL VALUES ('Movie6', 'Sci-Fi');
INSERT INTO PAL_FRM_ITEM_INFORMATION_TBL VALUES ('Movie7', 'Crime,Drama');
```

```
INSERT INTO PAL_FRM_ITEM_INFORMATION_TBL VALUES ('Movie8', 'Sci-Fi,Thriller');
INSERT INTO PAL_FRM_ITEM_INFORMATION_TBL VALUES ('Bad_Movie',
'Romance,Thriller');
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL ("PARAM_NAME"
VARCHAR(256), "INT_VALUE" INTEGER, "DOUBLE_VALUE" DOUBLE, "STRING_VALUE"
VARCHAR(1000));
INSERT INTO #PAL_PARAMETER_TBL VALUES ('HAS_ID', 1, NULL, NULL); -- ignore the
first column in DATA table
INSERT INTO #PAL_PARAMETER_TBL VALUES ('CATEGORICAL_VARIABLE', NULL, NULL,
'TIMESTAMP'); -- column TIMESTAMP is categorical
INSERT INTO #PAL_PARAMETER_TBL VALUES ('RESAMPLING_METHOD', NULL, NULL, 'cv');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('METHOD', 1, NULL, NULL); -- use MOMENTUM
method
INSERT INTO #PAL_PARAMETER_TBL VALUES ('LEARNING_RATE', NULL, 0, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MAX_ITERATION', 100, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PARAM_SEARCH_STRATEGY', NULL, NULL,
'grid');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('EVALUATION_METRIC', NULL, NULL, 'RMSE');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FOLD_NUM', 5, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('REPEAT_TIMES', 1, NULL, NULL);
--INSERT INTO #PAL_PARAMETER_TBL VALUES ('TIMEOUT', 0, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PROGRESS_INDICATOR_ID', NULL, NULL,
'PAL_FRM');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FACTOR_NUMBER_RANGE', NULL, NULL, '[1,
1, 3]');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('LINEAR_REGULARIZATION_VALUES', NULL,
NULL, '{1e-6, 1e-8, 1e-10}');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('REGULARIZATION_VALUES', NULL, NULL,
'{1e-4, 1e-6, 1e-8}');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MOMENTUM_VALUES', NULL, NULL, '{0.8,
0.9}');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 0.5, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SEED', 1, NULL, NULL);
DROP TABLE PAL_FRM_MODEL_METADATA_TBL;
DROP TABLE PAL_FRM_MODEL_TBL;
DROP TABLE PAL_FRM_MODEL_FACTORS_TBL;
CREATE COLUMN TABLE PAL_FRM_MODEL_METADATA_TBL ("ROW_INDEX" INTEGER,
"METADATA_CONTENT" NVARCHAR(5000));
CREATE COLUMN TABLE PAL_FRM_MODEL_TBL ("ID" INTEGER, "MAP" NVARCHAR(1000),
"WEIGHT" DOUBLE);
CREATE COLUMN TABLE PAL_FRM_MODEL_FACTORS_TBL ("FACTOR_ID" INTEGER, "FACTOR"
DOUBLE);
CALL _SYS_AFL.PAL_FRM (PAL_FRM_DATA_TBL, PAL_FRM_USER_INFORMATION_TBL,
PAL_FRM_ITEM_INFORMATION_TBL, #PAL_PARAMETER_TBL, PAL_FRM_MODEL_METADATA_TBL,
PAL_FRM_MODEL_TBL, PAL_FRM_MODEL_FACTORS_TBL, ?, ?, ?) WITH OVERVIEW;
```

### Expected Result

Results of different runs might be slightly different due to the inherently randomicity of parallel SGD.

**MODEL_METADATA:**

| | ROW_INDEX | METADATA_CONTENT |
|---|---|---|
| 1 | 0 | {"bias":3.9486486486486494,"eta":0.0,"factor_num":1,"interactionCategor... |

MODEL:

| | ID | MAP | WEIGHT |
|---|---|---|---|
| 1 | 0 | A | -0.09997406311644869 |
| 2 | 1 | B | 0.5328741438157202 |
| 3 | 2 | C | -0.32402883946993466 |
| 4 | 3 | D | -0.13030170859640824 |
| 5 | 4 | E | -0.09508920133478113 |
| 6 | 5 | Bad_Movie | -0.7628029597620826 |
| 7 | 6 | Movie1 | 0.34754256113633886 |
| 8 | 7 | Movie2 | 0.39752645758385013 |
| 9 | 8 | Movie3 | -0.3725061090335111 |
| 10 | 9 | Movie4 | 0.48224613827163887 |
| 11 | 10 | Movie5 | 0.4093192142593134 |
| 12 | 11 | Movie6 | 0.06667489591938486 |
| 13 | 12 | Movie7 | -0.5717386457670108 |
| 14 | 13 | Movie8 | -0.019773597067857... |
| 15 | 14 | 1 | -0.47193516165828575 |
| 16 | 15 | 2 | -0.23432097667358823 |
| 17 | 16 | 3 | -0.20880866559356523 |
| 18 | 17 | 4 | -0.019669503006330... |
| 19 | 18 | Crime | 0.08842091619634736 |
| 20 | 19 | Drama | 0.3478961201910527 |
| 21 | 20 | Romance | -0.07026624855215988 |
| 22 | 21 | Sci-Fi | 0.5619516088190025 |
| 23 | 22 | Thriller | -0.4622228213916527 |

MODEL_FACTORS:

| | FACTOR_ID | FACTOR |
|---|---|---|
| 1 | 0 | -0.11685801840843917 |
| 2 | 1 | 0.3906687893193793 |
| 3 | 2 | -0.08466911656332665 |
| 4 | 3 | 0.10750625069575064 |
| 5 | 4 | -0.22287579625725767 |
| 6 | 5 | 0.1289424909245846 |
| 7 | 6 | -0.06029460935276821 |
| 8 | 7 | 0.1349207400277093 |
| 9 | 8 | 0.10025224060029933 |
| 10 | 9 | 0.0242470584122480... |
| 11 | 10 | -0.17028860230532483 |
| 12 | 11 | -0.3977287180844284 |
| 13 | 12 | -0.08867687856941175 |
| 14 | 13 | 0.3587138227217409 |
| 15 | 14 | -0.03338893020688217 |
| 16 | 15 | 0.4916376266282123 |
| 17 | 16 | -0.5122181482144312 |
| 18 | 17 | 0.08916854393225232 |
| 19 | 18 | -0.1154954372684597 |
| 20 | 19 | 0.0099058265441090... |
| 21 | 20 | 0.13675645299767678 |
| 22 | 21 | -0.17190793754276945 |
| 23 | 22 | 0.2312618342432318 |

ITERATION_INFORMATION:

| | ITERATION | COST | RMSE | STEP_SIZE |
|---|---|---|---|---|
| 1 | 1 | 0.391804271419606 | 0.625942573635564 | 0.048828125 |
| 2 | 2 | 0.7066489770764434 | 0.8406240268751143 | 0.02903337683112112 |
| 3 | 3 | 0.34851634862737... | 0.5903526187669818 | 0.02142047547123197 |
| 4 | 4 | 0.3686806117880261 | 0.6071906691039372 | 0.017263349150062... |
| 5 | 5 | 0.2915064665101952 | 0.53991330753201 | 0.01460301544161348 |
| 6 | 6 | 0.17609975861435... | 0.41964229361873... | 0.012736690918565... |
| 7 | 7 | 0.12903623532838... | 0.3592160044731056 | 0.011346097668092... |
| 8 | 8 | 0.08991685289168... | 0.2998612425603972 | 0.01026484881901507 |
| 9 | 9 | 0.05904798092875... | 0.2429977344014241 | 0.009396977037591... |
| 10 | 10 | 0.06315216493004... | 0.25130078490473... | 0.008683004931830... |
| 11 | 11 | 0.05216171362121... | 0.22838921021161 | 0.008083985648250... |
| 12 | 12 | 0.04203748644543... | 0.20503024475935... | 0.007573281730974... |
| 13 | 13 | 0.03892240976837... | 0.19728741126998... | 0.007132019689437... |
| 14 | 14 | 0.03551440980264... | 0.188452434168683 | 0.006746430037205... |
| 15 | 15 | 0.03299185125111... | 0.18163634507422... | 0.006406216377817... |
| 16 | 16 | 0.03281760391236... | 0.18115605519834... | 0.006103515625 |
| 17 | 17 | 0.03379127351500... | 0.1838237929298346 | 0.005832212732275... |
| 18 | 18 | 0.03039884670341... | 0.17435239969509... | 0.002793737988155... |
| 19 | 19 | 0.02816478061053... | 0.16782340055162... | 0.002682716938129... |
| 20 | 20 | 0.02723800124708... | 0.16503913007386... | 0.00258147281113419 |
| 21 | 21 | 0.02694810477427... | 0.16415851405908... | 0.002488717381566... |
| 22 | 22 | 0.02630088718161... | 0.16217521625883... | 0.002403383312619... |
| 23 | 23 | 0.02541761095154... | 0.15942874659026... | 0.002324578276892... |
| 24 | 24 | 0.02546652262172... | 0.1595820693130256 | 0.002251550129598... |
| 25 | 25 | 0.02403171309564... | 0.1550213763556249 | 0.002183660134277... |
| 26 | 26 | 0.02383571174908... | 0.15438790679263... | 0.002120362139754... |
| 27 | 27 | 0.02379913907005 | 0.15426941727675 | 0.002061186213247 |

STATISTICS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | timeout | FALSE |
| 2 | TEST_RMSE | 0.326365, 0.311011, 0.310482, 0.398279, 0.252378 |
| 3 | TEST_RMSE.MEAN | 0.319703 |
| 4 | TEST_RMSE.VAR | 0.00272797 |
| 5 | EVAL_RESULTS_1 | {"candidates":[{"TEST_RMSE":[[0.355191386847... |
| 6 | EVAL_RESULTS_2 | 8604,"TEST_RMSE.VAR":0.002727943513294949,... |
| 7 | EVAL_RESULTS_3 | ST_RMSE":[[0.3264848514771128,0.3110113077... |
| 8 | EVAL_RESULTS_4 | ,"TEST_RMSE.VAR":0.007097745703488181,"par... |
| 9 | EVAL_RESULTS_5 | "TEST_RMSE":[[0.355183511339385 6,0.3439294... |
| 10 | EVAL_RESULTS_6 | MSE.VAR":0.002727968475460109,"parameters"... |
| 11 | EVAL_RESULTS_7 | 47659195,0.3327930570360040,0.37988046110... |
| 12 | EVAL_RESULTS_8 | 8114953799612,"parameters":"MOMENTUM=0.... |
| 13 | EVAL_RESULTS_9 | 234503364,0.3433206080040884,0.3549945241... |
| 14 | EVAL_RESULTS_10 | .VAR":0.005873653623155743,"parameters":"M... |
| 15 | EVAL_RESULTS_11 | 933,0.3327933269943308,0.3798837892437122,... |
| 16 | EVAL_RESULTS_12 | 65,"parameters":"MOMENTUM=0.8;REGULARIZ... |
| 17 | EVAL_RESULTS_13 | 38,0.3447538921732173,0.4104809568591178,0.... |
| 18 | EVAL_RESULTS_14 | meters":"MOMENTUM=0.9;REGULARIZATION=... |
| 19 | EVAL_RESULTS_15 | 6421348885888,0.2919580140392778,0.362109... |
| 20 | EVAL_RESULTS_16 | "MOMENTUM=0.8;REGULARIZATION=9.99999e... |

OPTIMAL_PARAMETER:

| | PARAM_NAME | INT_VALUE | DOUBLE_VALUE | STRING_VALUE |
|---|---|---|---|---|
| 1 | MOMENTUM | ? | 0.9 | ? |
| 2 | REGULARIZATION | ? | 0.00000001 | ? |
| 3 | LINEAR_REGULA... | ? | 0.000001 | ? |
| 4 | FACTOR_NUMBER | 1 | ? | ? |

## _SYS_AFL.PAL_FRM_PREDICT

This function predicts users' ratings against items based on the recommender model trained by PAL_FRM.

**Overview of All Tables**

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<USER_INFORMATION table>` | IN |
| 3 | `<ITEM_INFORMATION table>` | IN |
| 4 | `<MODEL_METADATA table>` | IN |
| 5 | `<MODEL table>` | IN |
| 6 | `<MODEL_FACTORS table>` | IN |
| 7 | `<PARAMETER table>` | IN |
| 8 | `<RESULT table>` | OUT |

### Input Tables

| Table | Column | Reference for Output Table | Data Type | Description |
|---|---|---|---|---|
| DATA | 1st column | ID | INTEGER | ID |
| | 2nd column | | INTEGER, VARCHAR, or NVARCHAR | User name or user ID |
| | 3rd column | | INTEGER, VARCHAR, or NVARCHAR | Item name or item ID |
| | Side feature columns | | INTEGER, DOUBLE, DECIMAL(p, s), VARCHAR, or NVARCHAR | Global side features |
| USER_INFORMATION | 1st column | | INTEGER, VARCHAR, or NVARCHAR | User name or user ID |
| | Side feature columns | | INTEGER, DOUBLE, DECIMAL(p, s), VARCHAR, or NVARCHAR | Side features about user |
| ITEM_INFORMATION | 1st column | | INTEGER, VARCHAR, or NVARCHAR | Item name or item ID |
| | Side feature columns | | INTEGER, DOUBLE, DECIMAL(p, s), VARCHAR, or NVARCHAR | Side features about item |
| MODEL_METADATA | 1st column | | INTEGER | ID |

| Table | Column | Reference for Output Table | Data Type | Description |
|---|---|---|---|---|
| | 2nd column | | NVARCHAR | Model metadata content |
| MODEL | 1st column | | INTEGER | ID |
| | 2nd column | | NVARCHAR | Map |
| | 3rd column | | DOUBLE or DECIMAL(p, s) | Weight |
| MODEL_FACTORS | 1st column | | INTEGER | Factor ID |
| | 2nd column | | DOUBLE or DECIMAL(p, s) | Factors |

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

The following parameter is optional. If it is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| THREAD_RATIO | DOUBLE | 0 | Specifies the upper limit of thread usage in proportion of current available threads. The valid range of the value is [0,1], 0 indicates that single thread shall be used and 1 means all the current available threads will be used if necessary. Other value input will be ignored and the procedure will determine the number of threads to use heuristically. It is recommended to use this parameter instead of 'THREAD_NUMBER'. |

### Output Table

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| RESULT | 1st column | ID (in input table) data type | ID (in input table) column name | ID |
| | 2nd column | NVARCHAR(1000) | USER | User name or user ID |
| | 3rd column | NVARCHAR(1000) | ITEM | Item name or item ID |
| | 4th column | DOUBLE | PREDICTION | Predicted rating |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

**Example: Use the FRM model trained in Example 1 of _SYS_AFL.PAL_FRM section**

```
SET SCHEMA "DM_PAL";
DROP TABLE PAL_FRM_PREDICT_DATA_TBL;
CREATE COLUMN TABLE PAL_FRM_PREDICT_DATA_TBL ("ID" INTEGER, "USER"
NVARCHAR(100), "MOVIE" NVARCHAR(100), "TIMESTAMP" INTEGER);
INSERT INTO PAL_FRM_PREDICT_DATA_TBL VALUES (1, 'A', 'Movie3', null); --
TIMESTAMP for this term is unknown
INSERT INTO PAL_FRM_PREDICT_DATA_TBL VALUES (2, 'A', 'Movie7', 4);
INSERT INTO PAL_FRM_PREDICT_DATA_TBL VALUES (3, 'B', 'Movie1', 2);
INSERT INTO PAL_FRM_PREDICT_DATA_TBL VALUES (4, 'B', 'Movie6', 3);
INSERT INTO PAL_FRM_PREDICT_DATA_TBL VALUES (5, 'C', 'Movie3', 2);
INSERT INTO PAL_FRM_PREDICT_DATA_TBL VALUES (6, 'D', 'Movie2', 1);
INSERT INTO PAL_FRM_PREDICT_DATA_TBL VALUES (7, 'D', 'Movie5', 3);
INSERT INTO PAL_FRM_PREDICT_DATA_TBL VALUES (8, 'E', 'Bad_Movie', 2);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL ("PARAM_NAME"
VARCHAR(256), "INT_VALUE" INTEGER, "DOUBLE_VALUE" DOUBLE, "STRING_VALUE"
VARCHAR(1000));
INSERT INTO #PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 0.5, NULL);
CALL _SYS_AFL.PAL_FRM_PREDICT (PAL_FRM_PREDICT_DATA_TBL,
PAL_FRM_USER_INFORMATION_TBL, PAL_FRM_ITEM_INFORMATION_TBL,
PAL_FRM_MODEL_METADATA_TBL, PAL_FRM_MODEL_TBL, PAL_FRM_MODEL_FACTORS_TBL,
#PAL_PARAMETER_TBL, ?);
```

### Expected Result

RESULT:

| | ID | USER | ITEM | PREDICTION |
|---|---|---|---|---|
| 1 | 1 | A | Movie3 | 3.6259957117511203 |
| 2 | 2 | A | Movie7 | 3.534096468893867 |
| 3 | 3 | B | Movie1 | 5.530200704404194 |
| 4 | 4 | B | Movie6 | 5.416352793100551 |
| 5 | 5 | C | Movie3 | 3.1106442556310183 |
| 6 | 6 | D | Movie2 | 3.5606847985549743 |
| 7 | 7 | D | Movie5 | 4.113229534224349 |
| 8 | 8 | E | Bad_Movie | 2.3474784745536654 |

> **Note**
>
> The wrapper procedure generation approach to call PAL functions is still supported. For more information, refer to the older versions (prior to SAP HANA Platform 2.0 SPS 02) of this document.

## 5.11.3 Field-Aware Factorization Machine

In computational advertising, click-rough rate (CTR) and conversion rate (CVR), are two keys evaluating advertisement traffic. A recommender system is used to solve such a problem given some external features, including user information and items.

It is not uncommon that most features are categorical. With one-hot encoding, categorical feature is transformed to numeric one, rendering a quite sparse and huge feature space. Some features can affect the prediction together, for example, China and Chinese New Year, USA and Thanks giving. Therefore, it can be useful to add a second polynomial degree to the regression model, for example,

$$\hat{y}(x)=w_0+\sum_{i=1}^{n} w_i x_i+\sum_{i=1}^{n}\sum_{j=i+1}^{n} w_{ij} x_i x_j$$

where the model parameters to be estimated are:

$w_0 \in R, w \in R^n, w \in R^{n \times n}$

n is the number of features.

It can be seen that there are in total $\frac{n(n-1)}{2}$ parameters, any two of which are independent. However, normal linear regression algorithm is found difficult to estimate the second-degree parameter $w_{ij}$, as it requires a large number of non-zero $x_i$ and $x_j$. But the feature space is so sparse that there is insufficient data for training, giving rise to inaccuracy of model. To resolve this, FM proposes matrix factorisation:

$w = v^T v$

where $v \in R^{n \times k}$, and k is the number of dimensionality of the factorisation. Hence, the regression model is then

$$\hat{y}(x)=w_0+\sum_{i=1}^{n} w_i x_i+\sum_{i=1}^{n}\sum_{j=i+1}^{n} <v_i, v_j> x_i x_j$$

where < , > indicates the dot product of two vectors, which is $w_{ij} = <v_i, v_j>$. The number of the second degree parameters then reduces from n*n to n*k ($k \ll n$), and that's the core of FM. Moreover, the parameters to $x_i x_j$ and $x_i x_l$ are not independent any longer, therefore the $v_i$ parameter can be estimated from feature combination $x_i x_j$ wherever $x_i$ is non-zero, which can effectively address the sparsity problem to a large extent. The time complexity of FM parameter estimation is O(kn).

One more step beyond, field-aware factorisation machine (FFM) is suggested. FFM introduces a concept of filed to indicate that some similar features belong to a same field. In practice, naturally, features spanned from the same categorical variable are considered of the same filed. The representation for FFM regression is therefore

$$\hat{y}(x)=w_0+\sum_{i=1}^{n} w_i x_i+\sum_{i=1}^{n}\sum_{j=i+1}^{n} <v_{i,f_j}, v_{j,f_i}> x_i x_j$$

where $f_i$ is the field of feature i, $v \in R^{n*k*f}$, f is the number of fields. Hence, the number of the second-degree parameters is nfk, much larger than that of nk in FM, and its estimation complexity is $O(kn^2)$. It is noted that FFM is most suited to categorical features. A numeric feature is either regarded as a single filed, or discretised to categorical. If all features are numeric and treated as a single feature, which means each field consists of only one feature, FFM degenerates to FM.

FFM can be applied to a variety of prediction tasks, for example, binary classification, regression, and ranking. From the perspective of maximum a posterior (MAP), the loss functions are defined as:

binary classification:

$$f=\frac{\lambda}{2}\|\theta\|^2+\sum_{i=1}^{n} \log\left(1+\exp\left(-y_i\phi(\theta, x_i)\right)\right)$$

regression:

$$f=\frac{\lambda}{2}\|\theta\|^2+\frac{1}{2}\sum_{i=1}^{n}\left\|y_i-\phi(\theta, x_i)\right\|^2$$

ranking:

$$f=\frac{\lambda}{2}\|\theta\|^2-\sum_{i=1}^{n}\sum_{j=1}^{c} y_{ij} \log\left[G\left(\alpha_j-\phi(\theta, x_i)\right)-G\left(\alpha_{j-1}-\phi(\theta, x_i)\right)\right]$$

where $\theta=(w_0, w, v)$, $\phi(\theta, x_i)=\hat{y}(x_i)$, $G(z)=\frac{1}{1+\exp(-z)}$, $\lambda$ is the $L^2$ regularisation parameter to w and v, and $\alpha$ is the intercepts for ordinal regression. Evidently, these loss function are from logistic regression, linear regression, and cumulative logit model, respectively.

The minimum of loss function can be solved by stochastic gradient descent (SGD) algorithm, specifically AdaGrad, by requiring only its first derivatives to $\theta$. As the estimated parameter space is large, it might be easily over-fitted, hence some early-stopping strategy is applied, validation-based, for instance.

## Prerequisites

- The number of categories should be exactly 2 for binary classification, should be at least 2 for ranking.
- An ID column should be provided for prediction, and no NULL value is allowed in this column.
- All features in FFM training should be present in FFM predict.

## _SYS_AFL.PAL_FFM

This function trains the FFM model for binary classification, regression, or ranking.

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<META table>` | OUT |
| 4 | `<COEFFICIENT table>` | OUT |
| 5 | `<STATISTICS table>` | OUT |
| 6 | `<CROSS_VALIDATION table>` | OUT |

### Signature

### Input Table

| Table | Column | Data Type | Description | Constraint |
|---|---|---|---|---|
| DATA | Columns | VARCHAR, NVARCHAR, INTEGER, DOUBLE, or DECIMAL(p,s) | Independent variables.<br>Discrete values: INTEGER, VARCHAR, or NVARCHAR<br>Continuous values: INTEGER, DOUBLE, or DECIMAL(p,s) | If the HAS_ID parameter is 1, the first column is ID column and can be of BIGINT type as well. |

| Table | Column | Data Type | Description | Constraint |
|---|---|---|---|---|
| | Last column | INTEGER, DOUBLE, DECIMAL(p,s), VARCHAR, or NVARCHAR | Dependent variable.<br>The varchar type is regarded as categorical; the decimal and double types are as numeric.<br>The integer type is by default treated as numeric, but it can also be intentionally defined as categorical variable, specified by CATEGORICAL_VARIABLE. | If DEPENDENT_VARIABLE is set to other columns, the last column is used as independent variable. |

**Parameter Table**

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | | Description | Dependency |
|---|---|---|---|---|
| TASK | VARCHAR, or NVARCHAR | Classification if response is categorical; regression if response is numericDefault Value | The task of FFM:<br>• Classification: binary classification<br>• Regression: linear regression<br>• Ranking: ranking using ordinal regression | If response is of string type, it can be only classification or ranking;<br>If response is of double type, it can be only classification (less or equal to 0 is class 0, others are class 1) or regression.<br>If response is of integer type, it can be any task, based on the type being categorical or numeric. |
| DEPENDENT_VARIABLE | VARCHAR, or NVARCHAR | Last column name | Specifies the dependent variable. | |

| Name | Data Type | | Description | Dependency |
|---|---|---|---|---|
| CATEGORICAL_VARIABLE | VARCHAR or NVARCHAR | Detected from input data | DefIndicates which variables are treated as categorical. The default behavior is:<br>• STRING: categorical<br>• INTEGER and DOUBLE: continuous | VALID only for INTEGER variables; omitted otherwise. |
| DELIMITER | VARCHAR, or NVARCHAR | , (comma) | The delimiter to separate string features. For example, "China, USA" indicates two feature values "China" and "USA". | Valid only for string feature. |
| ORDERING | VARCHAR, or NVARCHAR | No default value | Specifies the categories orders for ranking. | For string category only, separated by comma.<br>The default behavior is that integer category is ordered by its numeric value, whereas the string category is ordered by its alphabet order. |
| NORMALISE | INTEGER | 1 | Specifies whether to normalise each instance so that its L1 norm is 1. | |
| INCLUDE_CONSTANT | INTEGER | 1 | Specifies whether to include the $w_0$ constant part. | Always 1 for ranking task. |
| INCLUDE_LINEAR | INTEGER | 1 | Specifies whether to include the $w$ constant part. | |
| EARLY_STOP | INTEGER | 1 | Specifies whether to early stop the SGD optimisation. | Valid only if the value of TRAIN_RATIO is less than 1. |

| Name | Data Type | | Description | Dependency |
|---|---|---|---|---|
| SEED | INTEGER | 0 | Specifies the seed for random generator:<br>• 0: current time for seed<br>• Others: specified seed | |
| K_NUM | INTEGER | 4 | The factorisation dimensionality. | |
| MAX_ITERATION | INTEGER | 20 | The number of iterations for SGD. | |
| TRAIN_RATIO | DOUBLE | 0.8 if number of instances not less than 40, 1.0 otherwise | The ratio of training data set, and the remaining data set for validation. For example, 0.8 indicates that 80% for training, and the remaining 20% for validation. | |
| LEARNING_RATE | DOUBLE | 0.2 | The learning rate for SGD iteration. | |
| LINEAR_LAMBDA | DOUBLE | 1e-5 | The L2 regularisation parameter for `w`. | |
| POLY2_LAMBDA | DOUBLE | 1e-5 | The L2 regularisation parameter for `v`. | |
| CONVERGENCE_CRITERION | DOUBLE | 1e-5 | The criterion to determine the convergence of SGD. | |
| CONVERGENCE_INTERVAL | INTEGER | 5 | The interval of two iterations for comparison to determine the convergence, which are compare f(i) and f(i-5). | |
| HANDLE_MISSING | INTEGER | 2 | Specifies how to handle missing feature:<br>• 1: remove missing rows.<br>• 2: replace missing rows with 0. | |

| Name | Data Type | | Description | Dependency |
|---|---|---|---|---|
| HAS_ID | INTEGER | 0 | Specifies whether the first column of DATA table is used as ID column:<br>• 0: No.<br>• 1: Yes; the first column is discarded during training. | |

### Output Tables

| Table | Column | Data Type | Column Name | Description | Constraint |
|---|---|---|---|---|---|
| META | 1st column | INTEGER | ROW_INDEX | Indicates the row | Must be the first column |
| | 2nd column | NVARCHAR(5000) | META_VALUE | Meta data for model | JSON |
| COEFFICIENT | 1st column | INTEGER | COEFF_INDEX | Indicates the row | Must be the first column |
| | 2nd column | NVARCHAR(1000) | FEATURE | Feature name | |
| | 3rd column | NVARCHAR(1000) | FIELD | Field name | |
| | 4th column | INTEGER | K | The factorisation number | |
| | 5th column | DOUBLE | COEFFICIENT | The parameter value | |
| STATS | 1st column | NVARCHAR(1000) | STAT_NAME | Statistics name | |
| | 2nd column | NVARCHAR(1000) | STAT_VALUE | Statistics value | |
| CV | 1st column | NVARCHAR(256) | PARM_NAME | Parameter name | |
| | 2nd column | INTEGER | INT_VALUE | Integer value | |
| | 3rd column | DOUBLE | DOUBLE_VALUE | Double value | |
| | 4th column | NVARCHAR(1000) | STRING_VALUE | String value | |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;

- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

**Example 1: Binary classification**

```
SET SCHEMA "DM_PAL";
DROP TABLE PAL_FFM_DATA_TBL;
CREATE COLUMN TABLE PAL_FFM_DATA_TBL ("USER" NVARCHAR(100), "MOVIE"
NVARCHAR(100), "TIMESTAMP" INTEGER, "CTR" NVARCHAR(100));
INSERT INTO PAL_FFM_DATA_TBL VALUES ('A', 'Movie1', 3, 'Click');
INSERT INTO PAL_FFM_DATA_TBL VALUES ('A', 'Movie2', 3, 'Click');
INSERT INTO PAL_FFM_DATA_TBL VALUES ('A', 'Movie4', 1, 'Not click');
INSERT INTO PAL_FFM_DATA_TBL VALUES ('A', 'Movie5', 2, 'Click');
INSERT INTO PAL_FFM_DATA_TBL VALUES ('A', 'Movie6', 3, 'Click');
INSERT INTO PAL_FFM_DATA_TBL VALUES ('A', 'Movie8', 2, 'Not click');
INSERT INTO PAL_FFM_DATA_TBL VALUES ('A', 'Movie0, Movie3', 1, 'Click');
INSERT INTO PAL_FFM_DATA_TBL VALUES ('B', 'Movie2', 3, 'Click');
INSERT INTO PAL_FFM_DATA_TBL VALUES ('B', 'Movie3', 2, 'Click');
INSERT INTO PAL_FFM_DATA_TBL VALUES ('B', 'Movie4', 2, 'Not click');
INSERT INTO PAL_FFM_DATA_TBL VALUES ('B',  null, 4, 'Not click');
INSERT INTO PAL_FFM_DATA_TBL VALUES ('B', 'Movie7', 1, 'Click');
INSERT INTO PAL_FFM_DATA_TBL VALUES ('B', 'Movie8', 2, 'Not click');
INSERT INTO PAL_FFM_DATA_TBL VALUES ('B', 'Movie0', 3, 'Not click');
INSERT INTO PAL_FFM_DATA_TBL VALUES ('C', 'Movie1', 2, 'Click');
INSERT INTO PAL_FFM_DATA_TBL VALUES ('C', 'Movie2, Movie5, Movie7', 4, 'Not
click');
INSERT INTO PAL_FFM_DATA_TBL VALUES ('C', 'Movie4', 3, 'Not click');
INSERT INTO PAL_FFM_DATA_TBL VALUES ('C', 'Movie5', 1, 'Not click');
INSERT INTO PAL_FFM_DATA_TBL VALUES ('C', 'Movie6', null, 'Click');
INSERT INTO PAL_FFM_DATA_TBL VALUES ('C', 'Movie7', 3, 'Not click');
INSERT INTO PAL_FFM_DATA_TBL VALUES ('C', 'Movie8', 1, 'Click');
INSERT INTO PAL_FFM_DATA_TBL VALUES ('C', 'Movie0', 2, 'Click');
INSERT INTO PAL_FFM_DATA_TBL VALUES ('D', 'Movie1', 3, 'Click');
INSERT INTO PAL_FFM_DATA_TBL VALUES ('D', 'Movie3', 2, 'Click');
INSERT INTO PAL_FFM_DATA_TBL VALUES ('D', 'Movie4, Movie7', 2, 'Click');
INSERT INTO PAL_FFM_DATA_TBL VALUES ('D', 'Movie6', 2, 'Click');
INSERT INTO PAL_FFM_DATA_TBL VALUES ('D', 'Movie7', 4, 'Not click');
INSERT INTO PAL_FFM_DATA_TBL VALUES ('D', 'Movie8', 3, 'Not click');
INSERT INTO PAL_FFM_DATA_TBL VALUES ('D', 'Movie0', 3, 'Not click');
INSERT INTO PAL_FFM_DATA_TBL VALUES ('E', 'Movie1', 2, 'Not click');
INSERT INTO PAL_FFM_DATA_TBL VALUES ('E', 'Movie2', 2, 'Click');
INSERT INTO PAL_FFM_DATA_TBL VALUES ('E', 'Movie3', 2, 'Click');
INSERT INTO PAL_FFM_DATA_TBL VALUES ('E', 'Movie4', 4, 'Click');
INSERT INTO PAL_FFM_DATA_TBL VALUES ('E', 'Movie5', 3, 'Click');
INSERT INTO PAL_FFM_DATA_TBL VALUES ('E', 'Movie6', 2, 'Not click');
INSERT INTO PAL_FFM_DATA_TBL VALUES ('E', 'Movie7', 4, 'Not click');
INSERT INTO PAL_FFM_DATA_TBL VALUES ('E', 'Movie8', 3, 'Not click');
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL ("PARAM_NAME"
VARCHAR(256), "INT_VALUE" INTEGER, "DOUBLE_VALUE" DOUBLE, "STRING_VALUE"
VARCHAR(1000));
INSERT INTO #PAL_PARAMETER_TBL VALUES ('TASK', NULL, NULL, 'classification');
--INSERT INTO #PAL_PARAMETER_TBL VALUES ('HAS_ID', 1, NULL, NULL); -- ignore the
first column in DATA table
INSERT INTO #PAL_PARAMETER_TBL VALUES ('CATEGORICAL_VARIABLE', NULL, NULL,
'TIMESTAMP'); -- column TIMESTAMP is categorical
INSERT INTO #PAL_PARAMETER_TBL VALUES ('DELIMITER', NULL, NULL, ',');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('K_NUM', 4, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('EARLY_STOP', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('LEARNING_RATE', NULL, 0.2, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MAX_ITERATION', 20, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('TRAIN_RATIO', NULL, 0.8, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('LINEAR_LAMBDA', NULL, 1e-5, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('POLY2_LAMBDA', NULL, 1e-6, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SEED', 1, NULL, NULL);
DROP TABLE PAL_FFM_META_TBL;
```

```
CREATE COLUMN TABLE PAL_FFM_META_TBL ("ROW_INDEX" INTEGER, "META_VALUE"
NVARCHAR(5000));
DROP TABLE PAL_FFM_COEFF_TBL;
CREATE COLUMN TABLE PAL_FFM_COEFF_TBL ("ID" INTEGER, "FEATURE" NVARCHAR(1000),
"FIELD" NVARCHAR(1000), "K" INTEGER, "COEFF" DOUBLE);
--CALL _SYS_AFL.PAL_FFM (PAL_FFM_DATA_TBL, #PAL_PARAMETER_TBL, PAL_FFM_META_TBL,
PAL_FFM_COEFF_TBL, ?, ?) WITH OVERVIEW;
CALL _SYS_AFL.PAL_FFM (PAL_FFM_DATA_TBL, #PAL_PARAMETER_TBL, ?, ?, ?, ?);
```

**Expected Results**

META:

| | ROW_INDEX | META_VALUE |
|---|---|---|
| 1 | 0 | {"Cat":{"CTR":["Click","Not click"],"MOVIE":["Movie0","Movie1","Movie2","Movie3","Movi... |

COEFFICIENT:

| | COEFF_INDEX | FEATURE | FIELD | K | COEFFICIENT |
|---|---|---|---|---|---|
| 1 | 0 | c | ? | ? | -0.031662396524501826 |
| 2 | 1 | USER:A | ? | ? | -0.13690224341587381 |
| 3 | 2 | USER:B | ? | ? | -0.04620828797368197 |
| 4 | 3 | USER:C | ? | ? | 0.10801252582347129 |
| 5 | 4 | USER:D | ? | ? | -0.04806942044296522 |
| 6 | 5 | USER:E | ? | ? | 0.12043111910482923 |
| 7 | 6 | MOVIE:... | ? | ? | 0.06325028765675819 |
| 8 | 7 | MOVIE:... | ? | ? | -0.026585898743830716 |
| 9 | 8 | MOVIE:... | ? | ? | -0.1745061480004107 |
| 10 | 9 | MOVIE:... | ? | ? | -0.22425819935968322 |
| 11 | 10 | MOVIE:... | ? | ? | 0.17727310642096775 |
| 12 | 11 | MOVIE:... | ? | ? | -0.050153899982991745 |
| 13 | 12 | MOVIE:... | ? | ? | -0.11082102509637112 |
| 14 | 13 | MOVIE:... | ? | ? | 0.11793601855195585 |
| 15 | 14 | MOVIE:... | ? | ? | 0.14133742969374394 |
| 16 | 15 | TIMES... | ? | ? | -0.04852195507615037 |
| 17 | 16 | TIMES... | ? | ? | -0.2144766491775144 |
| 18 | 17 | TIMES... | ? | ? | -0.056187110107451685 |
| 19 | 18 | TIMES... | ? | ? | 0.3484412029151476 |
| 20 | 19 | USER:A | USER | 0 | 0.4985924041151328 |
| 21 | 20 | USER:A | USER | 1 | 0.46627868068408274 |
| 22 | 21 | USER:A | USER | 2 | 0.064062223886153 |
| 23 | 22 | USER:A | USER | 3 | 0.4995202577326368 |
| 24 | 23 | USER:A | MOVIE | 0 | 0.1277861177345113 |
| 25 | 24 | USER:A | MOVIE | 1 | 0.15174642750681291 |
| 26 | 25 | USER:A | MOVIE | 2 | 0.15097982789916728 |

STATS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | task | classification |
| 2 | feature_num | 18 |
| 3 | field_num | 3 |
| 4 | k_num | 4 |
| 5 | category | Click, Not ... |
| 6 | iter | 3 |
| 7 | tr-loss | 0.64093165... |
| 8 | va-loss | 0.74523547... |

CV

Reserved.

**Example 2: Regression**

```
SET SCHEMA "DM_PAL";
DROP TABLE PAL_FFM_DATA_TBL;
CREATE COLUMN TABLE PAL_FFM_DATA_TBL ("USER" NVARCHAR(100), "MOVIE"
NVARCHAR(100), "TIMESTAMP" INTEGER, "RATE" INTEGER);
INSERT INTO PAL_FFM_DATA_TBL VALUES ('A', 'Movie1', 3, 0);
INSERT INTO PAL_FFM_DATA_TBL VALUES ('A', 'Movie2', 3, 5);
INSERT INTO PAL_FFM_DATA_TBL VALUES ('A', 'Movie4', 1, 0);
INSERT INTO PAL_FFM_DATA_TBL VALUES ('A', 'Movie5', 2, 1);
INSERT INTO PAL_FFM_DATA_TBL VALUES ('A', 'Movie6', 3, 2);
INSERT INTO PAL_FFM_DATA_TBL VALUES ('A', 'Movie8', 2, 0);
INSERT INTO PAL_FFM_DATA_TBL VALUES ('A', 'Movie0, Movie3', 1, 5);
INSERT INTO PAL_FFM_DATA_TBL VALUES ('B', 'Movie2', 3, 4);
INSERT INTO PAL_FFM_DATA_TBL VALUES ('B', 'Movie3', 2, 4);
INSERT INTO PAL_FFM_DATA_TBL VALUES ('B', 'Movie4', 2, 0);
INSERT INTO PAL_FFM_DATA_TBL VALUES ('B',  null, 4, 3);
INSERT INTO PAL_FFM_DATA_TBL VALUES ('B', 'Movie7', 1, 4);
INSERT INTO PAL_FFM_DATA_TBL VALUES ('B', 'Movie8', 2, 0);
INSERT INTO PAL_FFM_DATA_TBL VALUES ('B', 'Movie0', 3, 4);
INSERT INTO PAL_FFM_DATA_TBL VALUES ('C', 'Movie1', 2, 3);
INSERT INTO PAL_FFM_DATA_TBL VALUES ('C', 'Movie2, Movie5, Movie7', 4, 2);
INSERT INTO PAL_FFM_DATA_TBL VALUES ('C', 'Movie4', 3, 1);
INSERT INTO PAL_FFM_DATA_TBL VALUES ('C', 'Movie5', 1, 0);
INSERT INTO PAL_FFM_DATA_TBL VALUES ('C', 'Movie6', null, 5);
INSERT INTO PAL_FFM_DATA_TBL VALUES ('C', 'Movie7', 3, 0);
INSERT INTO PAL_FFM_DATA_TBL VALUES ('C', 'Movie8', 1, 5);
INSERT INTO PAL_FFM_DATA_TBL VALUES ('C', 'Movie0', 2, 3);
INSERT INTO PAL_FFM_DATA_TBL VALUES ('D', 'Movie1', 3, 0);
INSERT INTO PAL_FFM_DATA_TBL VALUES ('D', 'Movie3', 2, 5);
INSERT INTO PAL_FFM_DATA_TBL VALUES ('D', 'Movie4, Movie7', 2, 5);
INSERT INTO PAL_FFM_DATA_TBL VALUES ('D', 'Movie6', 2, 5);
INSERT INTO PAL_FFM_DATA_TBL VALUES ('D', 'Movie7', 4, 0);
INSERT INTO PAL_FFM_DATA_TBL VALUES ('D', 'Movie8', 3, 1);
INSERT INTO PAL_FFM_DATA_TBL VALUES ('D', 'Movie0', 3, 1);
INSERT INTO PAL_FFM_DATA_TBL VALUES ('E', 'Movie1', 2, 1);
INSERT INTO PAL_FFM_DATA_TBL VALUES ('E', 'Movie2', 2, 5);
INSERT INTO PAL_FFM_DATA_TBL VALUES ('E', 'Movie3', 2, 3);
INSERT INTO PAL_FFM_DATA_TBL VALUES ('E', 'Movie4', 4, 2);
INSERT INTO PAL_FFM_DATA_TBL VALUES ('E', 'Movie5', 3, 5);
INSERT INTO PAL_FFM_DATA_TBL VALUES ('E', 'Movie6', 2, 0);
INSERT INTO PAL_FFM_DATA_TBL VALUES ('E', 'Movie7', 4, 2);
INSERT INTO PAL_FFM_DATA_TBL VALUES ('E', 'Movie8', 3, 0);
DROP TABLE #PAL_PARAMETER_TBL;
```

```
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL ("PARAM_NAME"
VARCHAR(256), "INT_VALUE" INTEGER, "DOUBLE_VALUE" DOUBLE, "STRING_VALUE"
VARCHAR(1000));
INSERT INTO #PAL_PARAMETER_TBL VALUES ('TASK', NULL, NULL, 'regression');
--INSERT INTO #PAL_PARAMETER_TBL VALUES ('HAS_ID', 1, NULL, NULL); -- ignore the
first column in DATA table
INSERT INTO #PAL_PARAMETER_TBL VALUES ('CATEGORICAL_VARIABLE', NULL, NULL,
'TIMESTAMP'); -- column TIMESTAMP is categorical
INSERT INTO #PAL_PARAMETER_TBL VALUES ('DELIMITER', NULL, NULL, ',');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FACTOR_NUMBER', 4, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('EARLY_STOP', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('LEARNING_RATE', NULL, 0.2, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MAX_ITERATION', 20, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('TRAIN_RATIO', NULL, 0.8, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('LINEAR_LAMBDA', NULL, 1e-5, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('POLY2_LAMBDA', NULL, 1e-6, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SEED', 1, NULL, NULL);
DROP TABLE PAL_FFM_META_TBL;
CREATE COLUMN TABLE PAL_FFM_META_TBL ("ROW_INDEX" INTEGER, "META_VALUE"
NVARCHAR(5000));
DROP TABLE PAL_FFM_COEFF_TBL;
CREATE COLUMN TABLE PAL_FFM_COEFF_TBL ("ID" INTEGER, "FEATURE" NVARCHAR(1000),
"FIELD" NVARCHAR(1000), "K" INTEGER, "COEFF" DOUBLE);
--CALL _SYS_AFL.PAL_FFM (PAL_FFM_DATA_TBL, #PAL_PARAMETER_TBL, PAL_FFM_META_TBL,
PAL_FFM_COEFF_TBL, ?, ?) WITH OVERVIEW;
CALL _SYS_AFL.PAL_FFM (PAL_FFM_DATA_TBL, #PAL_PARAMETER_TBL, ?, ?, ?, ?);
```

### Expected Results

META:

| | ROW_INDEX | META_VALUE |
|---|---|---|
| 1 | 0 | {"Cat":{"MOVIE":["Movie0","Movie1","Movie2","Movie3","Movie4","Movie5","Movie6","M... |

COEFFICIENT:

| | COEFF_INDEX | FEATURE | FIELD | K | COEFFICIENT |
|---|---|---|---|---|---|
| 1 | 0 | c | ? | ? | 1.0909289307266947 |
| 2 | 1 | USER:A | ? | ? | -0.2668772558892727 |
| 3 | 2 | USER:B | ? | ? | 0.796491476673184 |
| 4 | 3 | USER:C | ? | ? | 0.247097126385107... |
| 5 | 4 | USER:D | ? | ? | 0.5147258971363523 |
| 6 | 5 | USER:E | ? | ? | 0.1632277285694419 |
| 7 | 6 | MOVIE:... | ? | ? | 0.3530123381985857 |
| 8 | 7 | MOVIE:... | ? | ? | -0.5724168681194862 |
| 9 | 8 | MOVIE:... | ? | ? | 0.8482879640063895 |
| 10 | 9 | MOVIE:... | ? | ? | 0.9408183494420563 |
| 11 | 10 | MOVIE:... | ? | ? | -0.22024698131759... |
| 12 | 11 | MOVIE:... | ? | ? | 0.3289435252234563 |
| 13 | 12 | MOVIE:... | ? | ? | 0.201761877830507... |
| 14 | 13 | MOVIE:... | ? | ? | 0.3192243443808067 |
| 15 | 14 | MOVIE:... | ? | ? | 0.110757451874747... |
| 16 | 15 | TIMES... | ? | ? | 0.5538475990796558 |
| 17 | 16 | TIMES... | ? | ? | 0.7411501467583085 |
| 18 | 17 | TIMES... | ? | ? | 0.244724922365169... |
| 19 | 18 | TIMES... | ? | ? | -0.03809626381984... |
| 20 | 19 | USER:A | USER | 0 | 0.4985924041151328 |
| 21 | 20 | USER:A | USER | 1 | 0.466278680684082... |
| 22 | 21 | USER:A | USER | 2 | 0.064062223886153 |
| 23 | 22 | USER:A | USER | 3 | 0.4995202577326368 |
| 24 | 23 | USER:A | MOVIE | 0 | -0.06864857473986... |
| 25 | 24 | USER:A | MOVIE | 1 | 0.8028232966056866 |
| 26 | 25 | USER:A | MOVIE | 2 | 0.9965240380690346 |

STATS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | task | regression |
| 2 | feature_num | 18 |
| 3 | field_num | 3 |
| 4 | k_num | 4 |
| 5 | iter | 15 |
| 6 | tr-loss | 0.45033677... |
| 7 | va-loss | 1.68968130... |

CV

Reserved.

**Example 3: Ranking**

```
SET SCHEMA "DM_PAL";
```

```
DROP TABLE PAL_FFM_DATA_TBL;
CREATE COLUMN TABLE PAL_FFM_DATA_TBL ("USER" NVARCHAR(100), "MOVIE"
NVARCHAR(100), "TIMESTAMP" INTEGER, "RANK" NVARCHAR(100));
INSERT INTO PAL_FFM_DATA_TBL VALUES ('A', 'Movie1', 3, 'medium');
INSERT INTO PAL_FFM_DATA_TBL VALUES ('A', 'Movie2', 3, 'too high');
INSERT INTO PAL_FFM_DATA_TBL VALUES ('A', 'Movie4', 1, 'medium');
INSERT INTO PAL_FFM_DATA_TBL VALUES ('A', 'Movie5', 2, 'too low');
INSERT INTO PAL_FFM_DATA_TBL VALUES ('A', 'Movie6', 3, 'low');
INSERT INTO PAL_FFM_DATA_TBL VALUES ('A', 'Movie8', 2, 'low');
INSERT INTO PAL_FFM_DATA_TBL VALUES ('A', 'Movie0, Movie3', 1, 'too high');
INSERT INTO PAL_FFM_DATA_TBL VALUES ('B', 'Movie2', 3, 'high');
INSERT INTO PAL_FFM_DATA_TBL VALUES ('B', 'Movie3', 2, 'high');
INSERT INTO PAL_FFM_DATA_TBL VALUES ('B', 'Movie4', 2, 'medium');
INSERT INTO PAL_FFM_DATA_TBL VALUES ('B',  null, 4, 'medium');
INSERT INTO PAL_FFM_DATA_TBL VALUES ('B', 'Movie7', 1, 'high');
INSERT INTO PAL_FFM_DATA_TBL VALUES ('B', 'Movie8', 2, 'high');
INSERT INTO PAL_FFM_DATA_TBL VALUES ('B', 'Movie0', 3, 'high');
INSERT INTO PAL_FFM_DATA_TBL VALUES ('C', 'Movie1', 2, 'medium');
INSERT INTO PAL_FFM_DATA_TBL VALUES ('C', 'Movie2, Movie5, Movie7', 4, 'low');
INSERT INTO PAL_FFM_DATA_TBL VALUES ('C', 'Movie4', 3, 'too low');
INSERT INTO PAL_FFM_DATA_TBL VALUES ('C', 'Movie5', 1, 'high');
INSERT INTO PAL_FFM_DATA_TBL VALUES ('C', 'Movie6', null, 'too high');
INSERT INTO PAL_FFM_DATA_TBL VALUES ('C', 'Movie7', 3, 'high');
INSERT INTO PAL_FFM_DATA_TBL VALUES ('C', 'Movie8', 1, 'too high');
INSERT INTO PAL_FFM_DATA_TBL VALUES ('C', 'Movie0', 2, 'medium');
INSERT INTO PAL_FFM_DATA_TBL VALUES ('D', 'Movie1', 3, 'too high');
INSERT INTO PAL_FFM_DATA_TBL VALUES ('D', 'Movie3', 2, 'too high');
INSERT INTO PAL_FFM_DATA_TBL VALUES ('D', 'Movie4, Movie7', 2, 'too high');
INSERT INTO PAL_FFM_DATA_TBL VALUES ('D', 'Movie6', 2, 'too high');
INSERT INTO PAL_FFM_DATA_TBL VALUES ('D', 'Movie7', 4, 'too high');
INSERT INTO PAL_FFM_DATA_TBL VALUES ('D', 'Movie8', 3, 'too low');
INSERT INTO PAL_FFM_DATA_TBL VALUES ('D', 'Movie0', 3, 'too low');
INSERT INTO PAL_FFM_DATA_TBL VALUES ('E', 'Movie1', 2, 'too low');
INSERT INTO PAL_FFM_DATA_TBL VALUES ('E', 'Movie2', 2, 'too high');
INSERT INTO PAL_FFM_DATA_TBL VALUES ('E', 'Movie3', 2, 'medium');
INSERT INTO PAL_FFM_DATA_TBL VALUES ('E', 'Movie4', 4, 'low');
INSERT INTO PAL_FFM_DATA_TBL VALUES ('E', 'Movie5', 3, 'too high');
INSERT INTO PAL_FFM_DATA_TBL VALUES ('E', 'Movie6', 2, 'low');
INSERT INTO PAL_FFM_DATA_TBL VALUES ('E', 'Movie7', 4, 'low');
INSERT INTO PAL_FFM_DATA_TBL VALUES ('E', 'Movie8', 3, 'too low');
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL ("PARAM_NAME"
VARCHAR(256), "INT_VALUE" INTEGER, "DOUBLE_VALUE" DOUBLE, "STRING_VALUE"
VARCHAR(1000));
INSERT INTO #PAL_PARAMETER_TBL VALUES ('TASK', NULL, NULL, 'ranking');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('ORDERING', NULL, NULL, 'too low, low,
medium, high, too high');
--INSERT INTO #PAL_PARAMETER_TBL VALUES ('HAS_ID', 1, NULL, NULL); -- ignore the
first column in DATA table
INSERT INTO #PAL_PARAMETER_TBL VALUES ('CATEGORICAL_VARIABLE', NULL, NULL,
'TIMESTAMP'); -- column TIMESTAMP is categorical
INSERT INTO #PAL_PARAMETER_TBL VALUES ('DELIMITER', NULL, NULL, ',');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FACTOR_NUMBER', 4, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('EARLY_STOP', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('LEARNING_RATE', NULL, 0.2, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MAX_ITERATION', 20, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('TRAIN_RATIO', NULL, 0.8, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('LINEAR_LAMBDA', NULL, 1e-5, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('POLY2_LAMBDA', NULL, 1e-6, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SEED', 1, NULL, NULL);
DROP TABLE PAL_FFM_META_TBL;
CREATE COLUMN TABLE PAL_FFM_META_TBL ("ROW_INDEX" INTEGER, "META_VALUE"
NVARCHAR(5000));
DROP TABLE PAL_FFM_COEFF_TBL;
CREATE COLUMN TABLE PAL_FFM_COEFF_TBL ("ID" INTEGER, "FEATURE" NVARCHAR(1000),
"FIELD" NVARCHAR(1000), "K" INTEGER, "COEFF" DOUBLE);
--CALL _SYS_AFL.PAL_FFM (PAL_FFM_DATA_TBL, #PAL_PARAMETER_TBL, PAL_FFM_META_TBL,
PAL_FFM_COEFF_TBL, ?, ?) WITH OVERVIEW;
```

```
CALL _SYS_AFL.PAL_FFM (PAL_FFM_DATA_TBL, #PAL_PARAMETER_TBL, ?, ?, ?, ?);
```

### Expected Results

META:

| | ROW_INDEX | META_VALUE |
|---|---|---|
| 1 | 0 | {"Cat":{"MOVIE":["Movie0","Movie1","Movie2","Movie3","Movie4","Movie5","Movie6","M... |

COEFFICIENT:

| | COEFF_INDEX | FEATURE | FIELD | K | COEFFICIENT |
|---|---|---|---|---|---|
| 1 | 0 | c:too low\|low | ? | ? | -0.308075655755055 |
| 2 | 1 | c:low\|medium | ? | ? | 0.6219321884946667 |
| 3 | 2 | c:medium\|hi... | ? | ? | 1.4743852259795671 |
| 4 | 3 | c:high\|too hi... | ? | ? | 2.5084130626957855 |
| 5 | 4 | USER:A | ? | ? | -0.0384287941170... |
| 6 | 5 | USER:B | ? | ? | 0.9595114230705054 |
| 7 | 6 | USER:C | ? | ? | 0.5689620689805208 |
| 8 | 7 | USER:D | ? | ? | 0.8942732338943824 |
| 9 | 8 | USER:E | ? | ? | -0.3509931866706... |
| 10 | 9 | MOVIE:Movi... | ? | ? | -0.1519243140591... |
| 11 | 10 | MOVIE:Movi... | ? | ? | 0.10547924939210... |
| 12 | 11 | MOVIE:Movi... | ? | ? | 0.5963056526782091 |
| 13 | 12 | MOVIE:Movi... | ? | ? | 0.6315745174863396 |
| 14 | 13 | MOVIE:Movi... | ? | ? | 0.03167879110440... |
| 15 | 14 | MOVIE:Movi... | ? | ? | 0.206913166900934 |
| 16 | 15 | MOVIE:Movi... | ? | ? | 0.3587589717781842 |
| 17 | 16 | MOVIE:Movi... | ? | ? | 0.9377921815860627 |
| 18 | 17 | MOVIE:Movi... | ? | ? | -0.3885725821223... |
| 19 | 18 | TIMESTAMP:1 | ? | ? | 1.1840924227476362 |
| 20 | 19 | TIMESTAMP:2 | ? | ? | 0.5055707918304868 |
| 21 | 20 | TIMESTAMP:3 | ? | ? | 0.2799011768869794 |
| 22 | 21 | TIMESTAMP:4 | ? | ? | 0.2940222662623343 |
| 23 | 22 | USER:A | USER | 0 | 0.4985924041151328 |
| 24 | 23 | USER:A | USER | 1 | 0.46627868068408... |
| 25 | 24 | USER:A | USER | 2 | 0.064062223886153 |
| 26 | 25 | USER:A | USER | 3 | 0.4995202577326368 |

STATS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | task | ranking |
| 2 | feature_num | 18 |
| 3 | field_num | 3 |
| 4 | k_num | 4 |
| 5 | category | too low, lo... |
| 6 | iter | 14 |
| 7 | tr-loss | 1.34320135... |
| 8 | va-loss | 1.55097921... |

## _SYS_AFL.PAL_FFM_PREDICT

This algorithm carries out prediction based on FFM model.

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 3 | `<META table>` | IN |
| 4 | `<COEFFICIENT table>` | IN |
| 5 | `<PARAMETER table>` | OUT |
| 6 | `<RESULT table>` | OUT |

### Signature

| Table | Column | Reference | Data Type | Description | Constraint |
|---|---|---|---|---|---|
| DATA | 1st column | ID | INTEGER, BIGINT, VARCHAR, or NVARCHAR | Data ID | Must be the first column |
| | Other columns | | INTEGER, VARCHAR, or NVARCHAR | Independent variables | |
| META | 1st column | | INTEGER | Indicates the row | Must be the first column |

| Table | Column | Reference | Data Type | Description | Constraint |
|---|---|---|---|---|---|
| | 2nd column | | VARCHAR, or NVARCHAR | Meta data for model | JSON |
| COEFFICIENT | 1st column | | INTEGER | Indicates the row | Must be the first column |
| | 2nd column | | VARCHAR, or NVARCHAR | Feature name | |
| | 3rd column | | VARCHAR, or NVARCHAR | Field name | |
| | 4th column | | INTEGER | The factorisation number | |
| | 5th column | | DOUBLE | The parameter value | |

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| THREAD_RATIO | DOUBLE | -1 | The ratio of available threads.<br>• 0: single thread<br>• 0~1: percentage<br>• Others: heuristically determined | |
| HANDLE_MISSING | INTEGER | 2 | Specifies how to handle missing features:<br>• 1: remove missing rows<br>• 2: replace missing rows with 0 | |

### Output Tables

| Table | Column | Data Type | Column Name | Description | Constraint |
|---|---|---|---|---|---|
| RESULT | 1st column | ID column type | ID column name | ID | |
| | 2nd column | NVARCHAR(100) | SCORE | Predict value | |
| | 3rd column | DOUBLE | CONFIDENCE | The confidence of a category | Null for regression |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

**Example 1: Binary classification, regression, and ranking predict**

```
SET SCHEMA "DM_PAL";
DROP TABLE PAL_FFM_DATA_TBL;
CREATE COLUMN TABLE PAL_FFM_DATA_TBL ("ID" INTEGER, "USER" NVARCHAR(100),
"MOVIE" NVARCHAR(100), "TIMESTAMP" INTEGER);
INSERT INTO PAL_FFM_DATA_TBL VALUES (1, 'A', 'Movie0', 3);
INSERT INTO PAL_FFM_DATA_TBL VALUES (2, 'A', 'Movie4', 1);
INSERT INTO PAL_FFM_DATA_TBL VALUES (3, 'B', 'Movie3', 2);
INSERT INTO PAL_FFM_DATA_TBL VALUES (4, 'B',  null, 5);
INSERT INTO PAL_FFM_DATA_TBL VALUES (5, 'C', 'Movie2', 2);
INSERT INTO PAL_FFM_DATA_TBL VALUES (6, 'F', 'Movie4', 3);
INSERT INTO PAL_FFM_DATA_TBL VALUES (7, 'D', 'Movie2', 2);
INSERT INTO PAL_FFM_DATA_TBL VALUES (8, 'D', 'Movie4', 1);;
INSERT INTO PAL_FFM_DATA_TBL VALUES (9, 'E', 'Movie7', 4);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL ("PARAM_NAME"
VARCHAR(256), "INT_VALUE" INTEGER, "DOUBLE_VALUE" DOUBLE, "STRING_VALUE"
VARCHAR(1000));
INSERT INTO #PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 1.0, NULL);
CALL _SYS_AFL.PAL_FFM_PREDICT (PAL_FFM_DATA_TBL, PAL_FFM_META_TBL,
PAL_FFM_COEFF_TBL, #PAL_PARAMETER_TBL, ?);
```

### Expected Results

RESULT (Binary classification):

| | ID | SCORE | CONFIDENCE |
|---|---|---|---|
| 1 | 1 | Not click | 0.5435374564615567 |
| 2 | 2 | Not click | 0.5454704648136381 |
| 3 | 3 | Click | 0.542736831984304 |
| 4 | 4 | Click | 0.5194578396768814 |
| 5 | 5 | Click | 0.511001002207135 |
| 6 | 6 | Not click | 0.5346096106632054 |
| 7 | 7 | Click | 0.537739273014967 |
| 8 | 8 | Not click | 0.5367811385552157 |
| 9 | 9 | Not click | 0.6354121531806209 |

RESULT (Regression):

| | ID | SCORE | CONFIDENCE |
|---|---|---|---|
| 1 | 1 | 2.978197866860172 | ? |
| 2 | 2 | 0.43883354766746385 | ? |
| 3 | 3 | 3.765106298778723 | ? |
| 4 | 4 | 1.8874204073998788 | ? |
| 5 | 5 | 3.588371752514674 | ? |
| 6 | 6 | 1.3448502862740495 | ? |
| 7 | 7 | 5.268571202934171 | ? |
| 8 | 8 | 0.8713338730015039 | ? |
| 9 | 9 | 2.347070689885986 | ? |

RESULT (Ranking):

| | ID | SCORE | CONFIDENCE |
|---|---|---|---|
| 1 | 1 | high | 0.29420579062044705 |
| 2 | 2 | medium | 0.20989340423722114 |
| 3 | 3 | too low | 0.31660938607937616 |
| 4 | 4 | high | 0.21967058419745936 |
| 5 | 5 | too high | 0.22254497788271843 |
| 6 | 6 | high | 0.38562113821634253 |
| 7 | 7 | too low | 0.40769505794313154 |
| 8 | 8 | too low | 0.2952001437544478 |
| 9 | 9 | high | 0.28263275677684924 |

## Related Information

Generalised Linear Models [page 456]
Factorized Polynomial Regression Models [page 1328]

## 5.12 Miscellaneous

This section describes the ABC Analysis and Weighted Score Table algorithms that are provided by the Predictive Analysis Library.

### 5.12.1 ABC Analysis

This algorithm is used to classify objects (such as customers, employees, or products) based on a particular measure (such as revenue or profit).

ABC analysis suggests that inventories of an organization are not of equal value, thus can be grouped into three categories (A, B, and C) by their estimated importance. “A” items are very important for an organization. “B” items are of medium importance, that is, less important than “A” items and more important than “C” items. “C” items are of the least importance.

An example of ABC classification is as follows:

- “A” items – 20% of the items (customers) accounts for 70% of the revenue.
- “B” items – 30% of the items (customers) accounts for 20% of the revenue.
- “C” items – 50% of the items (customers) accounts for 10% of the revenue.

#### Prerequisites

- Input data cannot contain null value.
- The item names in the input table must be of VARCHAR or NVARCHAR data type and be unique.

#### _SYS_AFL.PAL_ABC

This function performs the ABC analysis algorithm.

**Overview of All Tables**

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table type>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<RESULT table>` | OUT |

**Input Table**

| Table | Column | Reference for Output Table | Data Type | Description |
|---|---|---|---|---|
| DATA | 1st column | ID | INTEGER, VARCHAR, or NVARCHAR | Record ID |
| | 2nd column | | DOUBLE or DECIMAL(p, s) | Value |

### Parameter Table

Mandatory Parameters

The following parameters are mandatory and must be given values.

| Name | Data Type | Description |
|---|---|---|
| PERCENT_A | DOUBLE | Interval for A class |
| PERCENT_B | DOUBLE | Interval for B class |
| PERCENT_C | DOUBLE | Interval for C class |

Optional Parameter

The following parameter is optional. If it is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| THREAD_RATIO | DOUBLE | 1.0 | Specifies the ratio of total number of threads that can be used by this function. The value range is from 0 to 1, where 0 means only using 1 thread, and 1 means using at most all the currently available threads. Values outside the range will be ignored and this function heuristically determines the number of threads to use. |

### Output Table

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| RESULT | 1st column | VARCHAR or NVARCHAR | ABC_CLASS | ABC class |
| | 2nd column | ID (in input table) data type | ID (in input table) column name | Record ID |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_ABC_DATA_TBL;
CREATE COLUMN TABLE PAL_ABC_DATA_TBL(
    "ITEM" VARCHAR(100),
    "VALUE" DOUBLE
);
INSERT INTO PAL_ABC_DATA_TBL VALUES ('item1', 15.4);
INSERT INTO PAL_ABC_DATA_TBL VALUES ('item2', 200.4);
INSERT INTO PAL_ABC_DATA_TBL VALUES ('item3', 280.4);
INSERT INTO PAL_ABC_DATA_TBL VALUES ('item4', 100.9);
INSERT INTO PAL_ABC_DATA_TBL VALUES ('item5', 40.4);
INSERT INTO PAL_ABC_DATA_TBL VALUES ('item6', 25.6);
INSERT INTO PAL_ABC_DATA_TBL VALUES ('item7', 18.4);
INSERT INTO PAL_ABC_DATA_TBL VALUES ('item8', 10.5);
INSERT INTO PAL_ABC_DATA_TBL VALUES ('item9', 96.15);
INSERT INTO PAL_ABC_DATA_TBL VALUES ('item10', 9.4);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL(
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR(1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 0.3, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PERCENT_A', NULL, 0.7, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PERCENT_B', NULL, 0.2, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PERCENT_C', NULL, 0.1, NULL);
CALL _SYS_AFL.PAL_ABC(PAL_ABC_DATA_TBL, "#PAL_PARAMETER_TBL", ?);
```

### Expected Result

PAL_ABC_RESULT_TBL:

| | ABC_CLASS | ITEM |
|---|---|---|
| 1 | A | item3 |
| 2 | A | item2 |
| 3 | A | item4 |
| 4 | B | item9 |
| 5 | B | item5 |
| 6 | B | item6 |
| 7 | C | item7 |
| 8 | C | item1 |
| 9 | C | item8 |
| 10 | C | item10 |

> **Note**
>
> The wrapper procedure generation approach to call PAL functions is still supported. For more information, refer to the older versions (prior to SAP HANA Platform 2.0 SPS 02) of this document.

## 5.12.2 Model Visualization

Model visualization transforms model contents to DOT strings which can be further visualized.

Many tree-based machine learning algorithms in PAL have tree model contents as one of their outputs. Those model contents are typically in either PMML or JSON form, or both. Visualization of such model contents enables users to have a better understanding of how good the tree model is and what role each attribute plays in classification.

Model visualization supports the transformation for the following algorithms:

- decision trees (DT)
- random decision trees (RDT)
- hybrid gradient boosting tree (HGBT)

### Prerequisites

The model table must come directly from the supported PAL procedures mentioned previously.

# _SYS_AFL.PAL_VISUALIZE_MODEL

This function reads the output model table from DT, RDT, and HGBT, and transforms it into DOT string table.

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<MODEL table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<DOT table>` | OUT |

### Input Table

Comes directly from the `<MODEL table>` output of the following calls:

- `_SYS_AFL.PAL_DECISION_TREE`
- `_SYS_AFL.PAL_RANDOM_DECISION_TREES`
- `_SYS_AFL.PAL_HGBT`

### Parameter Table

None but requires a physical table.

### Output Table

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| MODEL | 1st column | INTEGER | ROW_INDEX | Indicates the current row. |
| | 2nd column | INTEGER | TREE_INDEX | Indicates the corre-sponding tree id in the input model table. |
| | 3rd column | NVARCHAR | MODEL_CONTENT | DOT string that can be visualized. |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

Example: Visualization for Decision Trees

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_DT_DATA_TBL;
CREATE COLUMN TABLE PAL_DT_DATA_TBL (
    "OUTLOOK" VARCHAR(20),
    "TEMP" INTEGER,
```

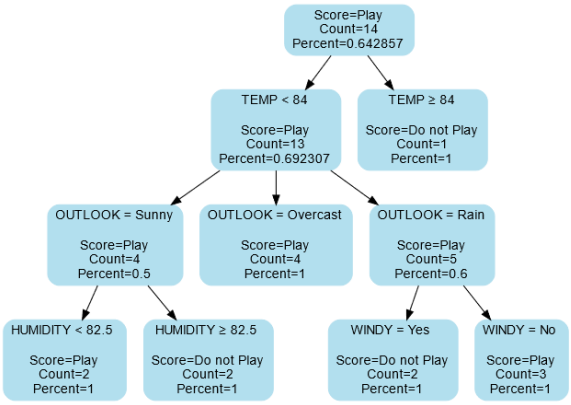
```
    "HUMIDITY" DOUBLE,
    "WINDY" VARCHAR(10),
    "CLASS" INTEGER -- for regression
);
INSERT INTO PAL_DT_DATA_TBL VALUES ('Sunny', 75, 70, 'Yes', 1);
INSERT INTO PAL_DT_DATA_TBL VALUES ('Sunny', 80, 90, 'Yes', 0);
INSERT INTO PAL_DT_DATA_TBL VALUES ('Sunny', 85, 85, 'No', 0);
INSERT INTO PAL_DT_DATA_TBL VALUES ('Sunny', 72, 95, 'No', 0);
INSERT INTO PAL_DT_DATA_TBL VALUES ('Sunny', 69, 70, 'No', 1);
INSERT INTO PAL_DT_DATA_TBL VALUES ('Overcast', 72, 90, 'Yes', 1);
INSERT INTO PAL_DT_DATA_TBL VALUES ('Overcast', 83, 78, 'No', 0);
INSERT INTO PAL_DT_DATA_TBL VALUES ('Overcast', 64, 65, 'Yes', 1);
INSERT INTO PAL_DT_DATA_TBL VALUES ('Overcast', 81, 75, 'No', 1);
INSERT INTO PAL_DT_DATA_TBL VALUES ('Rain', 71, 80, 'Yes', 0);
INSERT INTO PAL_DT_DATA_TBL VALUES ('Rain', 65, 70, 'Yes', 0);
INSERT INTO PAL_DT_DATA_TBL VALUES ('Rain', 75, 80, 'No', 1);
INSERT INTO PAL_DT_DATA_TBL VALUES ('Rain', 68, 80, 'No', 1);
INSERT INTO PAL_DT_DATA_TBL VALUES ('Rain', 70, 96, 'No', 0);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL ("NAME" VARCHAR
(50),"INT_VALUE" INTEGER,"DOUBLE_VALUE" DOUBLE,"STRING_VALUE" VARCHAR (100));
INSERT INTO #PAL_PARAMETER_TBL VALUES ('ALGORITHM', 3, NULL, NULL); -- CART
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MODEL_FORMAT', 2, NULL, NULL); -- PMML
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SPLIT_THRESHOLD', NULL, 1e-5, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MIN_RECORDS_OF_PARENT', 2, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MIN_RECORDS_OF_LEAF', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('IS_OUTPUT_RULES', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('IS_OUTPUT_CONFUSION_MATRIX', 1, NULL,
NULL);
--INSERT INTO #PAL_PARAMETER_TBL VALUES ('CATEGORICAL_VARIABLE', NULL, NULL,
'TEM');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 0.4, NULL);
--INSERT INTO #PAL_PARAMETER_TBL VALUES ('HAS_ID', 0, NULL, NULL);
--INSERT INTO #PAL_PARAMETER_TBL VALUES ('DEPENDENT_VARIABLE', NULL, NULL,
'CLASS');
DROP TABLE PAL_DT_MODEL_TBL; -- for predicted followed
CREATE COLUMN TABLE PAL_DT_MODEL_TBL (
    "ROW_INDEX" INTEGER,
    "MODEL_CONTENT" NVARCHAR(5000)
);
CALL _SYS_AFL.PAL_DECISION_TREE (PAL_DT_DATA_TBL, #PAL_PARAMETER_TBL,
PAL_DT_MODEL_TBL, ?, ?, ?, ?) WITH OVERVIEW;
TRUNCATE TABLE #PAL_PARAMETER_TBL;
CALL _SYS_AFL.PAL_VISUALIZE_MODEL(PAL_DT_MODEL_TBL, #PAL_PARAMETER_TBL, ?);
```

**Expected Results:**

DOT:

| | ROW_INDEX | MODEL_INDEX | MODEL_CONTENT |
|---|---|---|---|
| 1 | 0 | 0 | digraph G {node[shape=box, style="filled, rounded", color="lightblue2", fontname=helvet... |

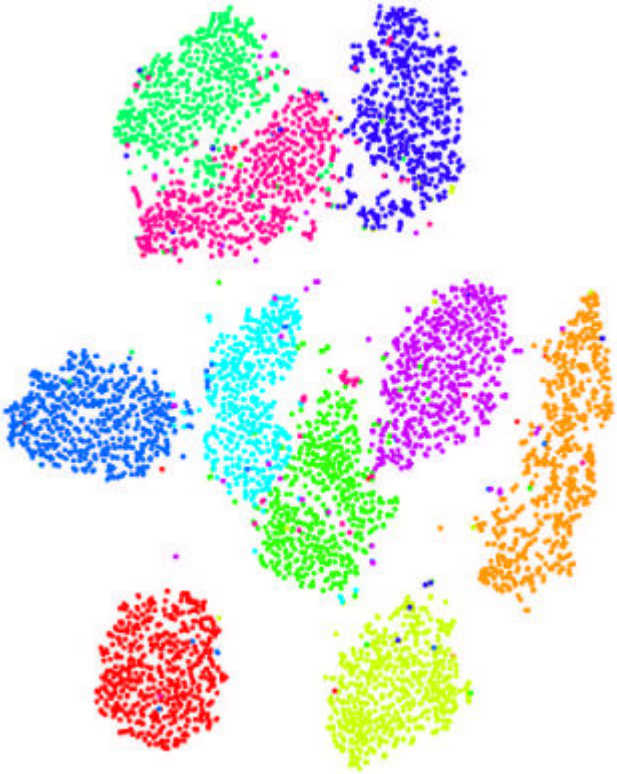
Visualized Model:




## 5.12.3 T-distributed Stochastic Neighbour Embedding

T-distributed stochastic neighbour embedding (TSNE) is a machine learning algorithm for visualization as well as dimensionality reduction.

The algorithm achieves high quality visualization by adopting unsupervised learning and non-linear method. Take following the visualization of MNIST dataset as an example. TSNE is capable of visualizing every digits (0 – 9) in cluster.



TSNE has been widely used for visualization of cancer detection, human genetic data, biomedical signal processing, and so on. The visualization could also help to understand how the deep neural network model works.

TSNE is based on the hypothesis that the distribution of original high dimensional dataset is accordance to the distribution of the 2/3-dimensional embedded space. We first introduces a notation $p_{j|i}$ denoting the conditional probability that sample $x_i$ would pick $x_j$ as its neighbour. So the $p_{j|i}$ would be relatively high for samples closed to each others while the value would be lower for distant samples. Mathematically we have $p_{j|i}$ represented by the following formula:

$$p_{j|i} = \frac{\exp\left(-\|x_i - x_j\|^2 / 2\sigma_i^2\right)}{\sum_{k \neq i} \exp\left(-\|x_i - x_k\|^2 / 2\sigma_i^2\right)}$$

where $\sigma_i$ is the variance of Gaussian that is centered on sample $x_i$. The value of $\sigma_i$ is related to the input parameter PERPLEXITY. Similarly, we denotes conditional probability $q_{j|i}$ on embedded space. Suppose $y_i$ is the counterpart of $x_i$, then we have:

$$q_{j|i} = \frac{\exp\left(-\|y_i - y_j\|^2\right)}{\sum_{k \neq i} \exp\left(-\|y_i - y_k\|^2\right)}$$

In case points $y_i$ and $y_j$ perfectly model the relation of $x_i$ and $x_j$, then the conditional probability $p_{j|i}$ and $q_{j|i}$ would be equal. TSNE aims to optimize the embedded space $Y = \{y_1, y_2, \ldots, y_n\}$ such that the distribution of $q_{j|i}$ has the lowest mismatch to the distribution of $p_{j|i}$. The optimization problem uses Kullback-Leibler divergence as the cost function. And the optimization is achieved by momentum gradient descent.

$$C = \sum_i KL(P_i || Q_i) = \sum_i \sum_j p_{j|i} \log \frac{p_{j|i}}{q_{j|i}}$$

The time complexity of the original TSNE algorithm is `O(N^2)`. Since it has to store the conditional probabilities between samples, the space complexity would be `O(N^2)`. To handle large amount of data, TSNE employs Barnes-Hut approximation which would optimize space complexity to O(N) and time complexity to O(NlogN), while reasonably sacrificing accuracy of the objective calculation. The tradeoff between objective value accuracy and calculation duration is controlled by parameter "THETA". By default, PAL TSNE uses the Barnes-Hut approximation method. The exact method should be chosen when the result is required to be accurate.

## Prerequisites

- The table for the input samples must have an ID column (1st column).
- Except for the ID column, the sample data is numerical, not categorical.
- The input data does not contain null value.
- Label column should not be included in the DATA table.

## _SYS_AFL.PAL_TSNE

This function applies TSNE algorithm.

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<RESULT table>` | OUT |
| 4 | `<STATISTICS table>` | OUT |
| 5 | `<OBJECTIVES table>` | OUT |

### Input Table

| Table | Column | Column Data Type | Description |
|---|---|---|---|
| DATA | 1st column | VARCHAR, NVARCHAR, INTEGER, or DOUBLE | ID column |

| Table | Column | Column Data Type | Description |
|---|---|---|---|
| | Feature columns | VARCHAR, NVARCHAR, INTEGER, DOUBLE or DECIMAL(p,s) | Feature data |

**Parameter Table**

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| MAX_ITER | INTEGER | 250 | Total iteration number. | |
| ETA | DOUBLE | 200.0 | Learning rate. | |
| OBJ_FREQ | INTEGER | 50 | Frequency of calculating the objective function and putting the result into OBJECTIVES table. PAL would calculate the objective function every 50 iterations by default. | Be noticed that this parameter value should not be larger than the MAX_ITER value. |
| NO_DIM | INTEGER | 2 | Dimension of the embedded space. Value other than 2 or 3 is illegal. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| THETA | DOUBLE | 0.5 | The legal value should be between 0.0 to 1.0. Setting it to 0.0 means using the "exact" method which would run O(N^2) time, otherwise TSNE would employ Barnes-Hut approximation which would run O(NlogN). This value is a trade-off between accuracy and training speed for Barnes-Hut approximation. The training speed would be faster with higher value. | |
| PERPLEXITY | DOUBLE | 30.0 | The perplexity is related to the number of nearest neighbors and $\sigma_i$ mentioned above. Larger value is suitable for large dataset. | Make sure PERPLEXITY * 3 < [no. of samples] |
| EXAGGERATION | DOUBLE | 12.0 | Value to be multiplied on $p_{ij}$ before 250 iterations. The natural clusters would be more separated with larger value, which means there would be more empty space on the map. | |
| THREAD_RATIO | DOUBLE | 0.0 | The ratio of available threads.<br>• 0: single thread<br>• 0~1: percentage | |
| SEED | INTEGER | 0 | The seed for random number generate. | |
| EXCLUDED_FEATURE | VARCHAR | No default value | Indicates the columns excluded from model training in DATA table. | |

### Output Tables

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| RESULT | 1st column | ID (in input table) column type | ID (in input table) column name | Indicates the row. |
| | 2nd column | DOUBLE | X | Coordinate value of the 1st dimension. |
| | 3rd column | DOUBLE | Y | Coordinate value of the 2nd dimension. |
| | 4th column | DOUBLE | Z | Coordinate value of the 3rd dimension. (NULL if the embedded space is 2 dimension) |
| STATISTICS | 1st column | NVARCHAR(256) | STAT_NAME | Statistics name. |
| | 2nd column | NVARCHAR(256) | STAT_VALUE | Statistics value. |
| OBJECTIVES | 1st column | INTEGER | ITER | Iteration step. |
| | 2nd column | DOUBLE | OBJ_VALUE | Objective value of the iteration. |

### Example 1: Embedded into two-dimensional space

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_TSNE_DATA_TBL;
CREATE COLUMN TABLE PAL_TSNE_DATA_TBL (
    "ID" INTEGER,
    "ATT1" DOUBLE,
    "ATT2" DOUBLE,
    "ATT3" DOUBLE,
    "ATT4" DOUBLE,
    "ATT5" DOUBLE
);
INSERT INTO PAL_TSNE_DATA_TBL VALUES(1,1.0,2.0,-10.0,-20.0,3.0);
INSERT INTO PAL_TSNE_DATA_TBL VALUES(2,4.0,5.0,-30.0,-10.0,6.0);
INSERT INTO PAL_TSNE_DATA_TBL VALUES(3,7.0,8.0,-40.0,-50.0,9.0);
INSERT INTO PAL_TSNE_DATA_TBL VALUES(4,10.0,11.0,-25.0,-15.0,12.0);
INSERT INTO PAL_TSNE_DATA_TBL VALUES(5,13.0,14.0,-12.0,-24.0,15.0);
INSERT INTO PAL_TSNE_DATA_TBL VALUES(6,16.0,17.0,-9.0,-13.0,18.0);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (100)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES('PERPLEXITY',  NULL, 1.0,  NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES('MAX_ITER', 600, NULL,  NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES('NO_DIM', 2, NULL,  NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES('THETA',  NULL, 0.5,  NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES('OBJ_FREQ', 100, NULL,  NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES('SEED', 30, NULL,  NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES('EXCLUDED_FEATURE', NULL, NULL, 'ATT3');
INSERT INTO #PAL_PARAMETER_TBL VALUES('EXCLUDED_FEATURE', NULL, NULL, 'ATT4');
```

```
CALL _SYS_AFL.PAL_TSNE(PAL_TSNE_DATA_TBL, #PAL_PARAMETER_TBL, ?, ?, ?);
```

**Expected Result**

RESULT:

| | ID | x | y | z |
|---|---|---|---|---|
| 1 | 1 | -73.19508775090924 | -59.19329199331623 | ? |
| 2 | 2 | -47.73583094845587 | -39.11497603149974 | ? |
| 3 | 3 | -12.210440255990... | -10.832997527517... | ? |
| 4 | 4 | 13.8658587605047... | 10.5395732923151... | ? |
| 5 | 5 | 47.39450513950467 | 38.82721281477113 | ? |
| 6 | 6 | 71.88099505534592 | 59.7744794452475... | ? |

STATISTICS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | method | barnes-hut |
| 2 | iter | 600 |
| 3 | objective | 0.07409617877015232 |

OBJECTIVES:

| | ITER | OBJ_VALUE |
|---|---|---|
| 1 | 100 | 53.616913120232496 |
| 2 | 200 | 52.1365210933462 |
| 3 | 300 | 0.10982461678964... |
| 4 | 400 | 0.14113676702992... |
| 5 | 500 | 0.07156496795525... |
| 6 | 600 | 0.07409617877015... |

**Example 2: Embedded into three-dimensional space**

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_TSNE_DATA_TBL;
CREATE COLUMN TABLE PAL_TSNE_DATA_TBL (
    "ID" INTEGER,
    "ATT1" DOUBLE,
    "ATT2" DOUBLE,
    "ATT3" DOUBLE,
    "ATT4" DOUBLE,
    "ATT5" DOUBLE
);
INSERT INTO PAL_TSNE_DATA_TBL VALUES(1,1.0,2.0,-10.0,-20.0,3.0);
INSERT INTO PAL_TSNE_DATA_TBL VALUES(2,4.0,5.0,-30.0,-10.0,6.0);
INSERT INTO PAL_TSNE_DATA_TBL VALUES(3,7.0,8.0,-40.0,-50.0,9.0);
INSERT INTO PAL_TSNE_DATA_TBL VALUES(4,10.0,11.0,-25.0,-15.0,12.0);
INSERT INTO PAL_TSNE_DATA_TBL VALUES(5,13.0,14.0,-12.0,-24.0,15.0);
INSERT INTO PAL_TSNE_DATA_TBL VALUES(6,16.0,17.0,-9.0,-13.0,18.0);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
```

```
    "STRING_VALUE" VARCHAR (100)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES('PERPLEXITY',  NULL, 1.0,  NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES('MAX_ITER', 500, NULL,  NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES('NO_DIM', 3, NULL,  NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES('THETA',  NULL, 0.0,  NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES('OBJ_FREQ', 50, NULL,  NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES('SEED', 30, NULL,  NULL);
CALL _SYS_AFL.PAL_TSNE(PAL_TSNE_DATA_TBL, #PAL_PARAMETER_TBL, ?, ?, ?);
```

**Expected Result**

RESULT:

| | ID | x | y | z |
|---|---|---|---|---|
| 1 | 1 | 4.875853475258551 | -189.0904973176342 | -229.536424203881 |
| 2 | 2 | -67.67545885343789 | 213.66173970516653 | 178.39762320935367 |
| 3 | 3 | -68.85291027809652 | 162.71085281028536 | 284.9662712408977 |
| 4 | 4 | -68.05610811278935 | 193.11805185463433 | 220.27543931553902 |
| 5 | 5 | 76.52462398680748 | -189.85092625849... | -227.62574972245... |
| 6 | 6 | 123.183999782257... | -190.5492207939598 | -226.47715983945... |

STATISTICS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | method | exact |
| 2 | iter | 500 |
| 3 | objective | 0.12310845438143747 |

OBJECTIVES:

| | ITER | OBJ_VALUE |
|---|---|---|
| 1 | 50 | 50.347529819455914 |
| 2 | 100 | 50.98219431523313 |
| 3 | 150 | 49.36841935562039 |
| 4 | 200 | 70.2012831718849 |
| 5 | 250 | 63.71753480822872 |
| 6 | 300 | 1.2966871014581893 |
| 7 | 350 | 0.8826361616560774 |
| 8 | 400 | 0.2605321169067199 |
| 9 | 450 | 0.17417840155221... |
| 10 | 500 | 0.12310845438143... |

## 5.12.4 Weighted Score Table

A weighted score table is a method of evaluating alternatives when the importance of each criterion differs.

In a weighted score table, each alternative is given a score for each criterion. These scores are then weighted by the importance of each criterion. All of an alternative's weighted scores are then added together to calculate its total weighted score. The alternative with the highest total score should be the best alternative.

You can use weighted score tables to make predictions about future customer behavior. You first create a model based on historical data in the data mining application, and then apply the model to new data to make the prediction. The prediction, which is the output of the model, is called a score. You can create a single score for your customers by taking into account different dimensions.

A function defined by weighted score tables is a linear combination of functions of a variable.

$$f(x_1,\ldots,x_n) = w_1 \times f_1(x_1) + \ldots + w_n \times f_n(x_n)$$

### Prerequisites

- The input data does not contain null value.
- The column of the Map Function table is sorted by the attribute order of the Input Data table.

### _SYS_AFL.PAL_WEIGHTED_TABLE

This function performs weighted table calculation. It is similar to the Volume Driver function in the Business Function Library (BFL). Volume Driver calculates only one column. Weighted table calculates multiple columns at the same time.

**Overview of All Tables**

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<MAP table>` | IN |
| 3 | `<WEIGHTS table>` | IN |
| 4 | `<PARAMETER table>` | IN |
| 5 | `<RESULT table>` | OUT |

**Input Tables**

| Table | Column | Reference for Output Table | Data Type | Description |
|---|---|---|---|---|
| DATA | 1st columns | ID | VARCHAR, NVARCHAR, INTEGER, DOUBLE, or DECIMAL(p, s) | Record ID |
| | Other columns | | | Specifies which will be used to calculate the scores |
| MAP | Columns | | VARCHAR, NVARCHAR, INTEGER, DOUBLE, or DECIMAL(p, s) | Every attribute (except ID) in the Input Data table maps to two columns in the Map Function table: Key column and Value column. The Value column must be of DOUBLE type. |
| WEIGHTS | Columns | | INTEGER, DOUBLE, or DECIMAL(p, s) | This table has three columns.<br>When the DATA table has *n* attributes (except ID), the WEIGHTS table will have *n* rows. |

### Parameter Table

Mandatory Parameters

None.

Optional Parameter

The following parameter is optional. If it is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| THREAD_RATIO | DOUBLE | 0 | Specifies the ratio of total number of threads that can be used by this function. The value range is from 0 to 1, where 0 means only using 1 thread, and 1 means using at most all the currently available threads. Values outside the range will be ignored and this function heuristically determines the number of threads to use. |

### Output Table

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| RESULT | 1st column | ID (in input table) data type | ID (in input table) column name | Record ID |
| | 2nd column | DOUBLE | SCORE | Result value |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_DATA_TBL;
CREATE COLUMN TABLE PAL_DATA_TBL(
    "ID" INTEGER,
    "GENDER" VARCHAR(10),
    "INCOME" INTEGER,
    "HEIGHT" DOUBLE
);
INSERT INTO PAL_DATA_TBL VALUES (0, 'male', 5000, 1.73);
INSERT INTO PAL_DATA_TBL VALUES (1, 'male', 9000, 1.80);
INSERT INTO PAL_DATA_TBL VALUES (2, 'female', 6000, 1.55);
INSERT INTO PAL_DATA_TBL VALUES (3, 'male', 15000, 1.65);
INSERT INTO PAL_DATA_TBL VALUES (4, 'female', 2000, 1.70);
INSERT INTO PAL_DATA_TBL VALUES (5, 'female', 12000, 1.65);
INSERT INTO PAL_DATA_TBL VALUES (6, 'male', 1000, 1.65);
INSERT INTO PAL_DATA_TBL VALUES (7, 'male', 8000, 1.60);
INSERT INTO PAL_DATA_TBL VALUES (8, 'female', 5500, 1.85);
INSERT INTO PAL_DATA_TBL VALUES (9, 'female', 9500, 1.85);
DROP TABLE PAL_MAP_TBL;
CREATE COLUMN TABLE PAL_MAP_TBL(
    "GENDER" VARCHAR(10),
    "VAL1" DOUBLE,
    "INCOME" INTEGER,
```

```
    "VAL2" DOUBLE,
    "HEIGHT" DOUBLE,
    "VAL3" DOUBLE
);
INSERT INTO PAL_MAP_TBL VALUES ('male', 2.0, 0, 0.0, 1.5, 0.0);
INSERT INTO PAL_MAP_TBL VALUES ('female', 1.5, 5500, 1.0, 1.6, 1.0);
INSERT INTO PAL_MAP_TBL VALUES (NULL, 0.0, 9000, 2.0, 1.71, 2.0);
INSERT INTO PAL_MAP_TBL VALUES (NULL, 0.0, 12000, 3.0, 1.80, 3.0);
DROP TABLE PAL_WEIGHTS_TBL;
CREATE COLUMN TABLE PAL_WEIGHTS_TBL(
    "WEIGHT" DOUBLE,
    "ISDIS" INTEGER,
    "ROWNUM" INTEGER
);
INSERT INTO PAL_WEIGHTS_TBL VALUES (0.5, 1,2);
INSERT INTO PAL_WEIGHTS_TBL VALUES (2.0, -1,4);
INSERT INTO PAL_WEIGHTS_TBL VALUES (1.0, -1,4);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL(
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR(1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 0.3, NULL);
CALL _SYS_AFL.PAL_WEIGHTED_TABLE(PAL_DATA_TBL, PAL_MAP_TBL, PAL_WEIGHTS_TBL,
"#PAL_PARAMETER_TBL", ?);
```

**Expected Result**

PAL_WT_RESULT_TBL:

| | ID | SCORE |
|---|---|---|
| 1 | 0 | 3 |
| 2 | 1 | 8 |
| 3 | 2 | 2.75 |
| 4 | 3 | 8 |
| 5 | 4 | 1.75 |
| 6 | 5 | 7.75 |
| 7 | 6 | 2 |
| 8 | 7 | 4 |
| 9 | 8 | 5.75 |
| 10 | 9 | 7.75 |

> **Note**
>
> The wrapper procedure generation approach to call PAL functions is still supported. For more information, refer to the older versions (prior to SAP HANA Platform 2.0 SPS 02) of this document.

## 5.13 Unified Interface

Unified interface contains unified classification, unified clustering, unified regression and unified exponential smoothing.

The unified interface procedures support two kinds of input data: normal and pivoted input data. While the normal data format also exists in other PAL procedures, the pivoted data format is special only in the unified interface.

Each row in the pivoted data includes only a feature of a data sample. This provides more flexibility for you to select the features you want. The following table shows an example of the pivoted data.

| ID | VARIABLE_NAME | VARIABLE_VALUE | PURPOSE |
|---|---|---|---|
| 0 | ATTR1 | 1 | 2 |
| 0 | ATTR2 | 10 | 2 |
| 0 | ATTR3 | 100 | 2 |
| 0 | ATTR4 | A | 2 |
| 0 | LABEL | 1 | 2 |
| 1 | ATTR1 | 1.1 | 1 |
| 1 | ATTR2 | 10.1 | 1 |
| 1 | ATTR3 | 100.1 | 1 |
| 1 | ATTR4 | B | 1 |
| 1 | LABEL | 2 | 1 |

In addition to the information of the feature, three additional columns are needed for pivoted data: ID, VARIABLE_NAME, and PURPOSE. The ID and VRAIABLE_NAME columns are used to group the data. The PURPOSE column is used for user-defined data partition.

### 5.13.1 Unified Classification

The unified classification interface refers to a new unified interface for classification algorithms.

The classification algorithms include:

- Decision trees
- Hybrid gradient boosting tree
- Logistic regression
- Multi-class logistic regression
- Naïve Bayes
- Random decision trees
- Support Vector Machine

## Prerequisite

All prerequisites for the original corresponding classification functions apply in unified classification interface.

## _SYS_AFL.PAL_UNIFIED_CLASSIFICATION

This function reads the input data, generates the training model and provides the evaluation statistics and metrics based on the training model and validation data.

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA Table>` | IN |
| 2 | `<PARAMETER Table>` | IN |
| 3 | `<MODEL Table>` | OUT |
| 4 | `<VARIABLE_IMPORTANCE Table>` | OUT |
| 5 | `<STATISTIC Table>` | OUT |
| 6 | `<OPTIMAL_PARAM Table>` | OUT |
| 7 | `<CONFUSION_MATRIX Table>` | OUT |
| 8 | `<METRICS Table>` | OUT |
| 9 | `<PARTITION_TYPE Table>` | OUT |
| 10 | `<PLACE_HOLDER1 Table>` | OUT |

### Signature

### Input Table

| Table | Column | Data Type | Description | Constraint |
|---|---|---|---|---|
| DATA | 1st column | INTEGER, VRACHAR, or NVARCHAR | Record ID. If this column is absent, the KEY parameter must be set to 0. | |
| | Feature column | INTEGER, DOUBLE, DECIMAL(p,s), VARCHAR, or NVARCHAR | Variable values. | |

| Table | Column | Data Type | Description | Constraint |
|---|---|---|---|---|
| | LABEL column | INTEGER, VARCHAR, or NVARCHAR | Dependent variable.<br>Normally, only VARCHAR and NVARCHAR are allowed. If INTEGER is used, CATEGORICAL_VARIABLE needs to be used to specify this column. | Usually this column is the last column except the following two scenarios:<br>• LABEL is set to be another column.<br>• The optional purpose column is needed, which means PARTITION_METHOD is 1. |
| | Purpose column (Optional) | INTEGER | An optional column used for user-defined data partition. This column is needed if PARTITION_METHOD is set to 1. Otherwise, the column does not exist. The valid values are:<br>• 1:training<br>• 2:validation | |

### Parameter Table

See details in Parameter Tables [page 1495].

### Output Table

| Table | Column | Column Name | Column Data Type |
|---|---|---|---|
| MODEL | 1st column | ROW_INDEX | INTEGER |
| | 2nd column | PART_INDEX | INTEGER |
| | 3rd column | MODEL_CONTENT | NCLOB |
| VARIABLE_IMPORTANCE | 1st column | VARIABLE_NAME | NVARCHAR(256) |
| | 2nd column | IMPORTANCE | DOUBLE |
| STATISTICS | 1st column | STAT_NAME | NVARCHAR (256) |
| | 2nd column | STAT_VALUE | NVARCHAR (1000) |
| | 3rd column | CLASS_NAME | NVARCHAR (256) |
| OPTIMAL_PARAM | 1st column | PARAM_NAME | NVARCHAR (256) |
| | 2nd column | INT_VALUE | INTEGER |

| Table | Column | Column Name | Column Data Type |
|---|---|---|---|
| | 3rd column | DOUBLE_VALUE | DOUBLE |
| | 4th column | STRING_VALUE | NVARCHAR (1000) |
| CONFUSION_MATRIX | 1st column | ACTUAL_CLASS | NVARCHAR (1000) |
| | 2nd column | PREDICTED_CLASS | NVARCHAR (1000) |
| | 3rd column | COUNT | INTEGER |
| METRICS | 1st column | NAME | NVARCHAR (256) |
| | 2nd column | X | DOUBLE |
| | 3rd column | Y | DOUBLE |
| PARTITION_TYPE | 1st column | ID column name (if there is no ID column, the name of first column of data table.) | ID column data type (if there is no ID column, the type of first column of data table.) |
| | 2nd column | TYPE | INTEGER |
| PLACE_HOLDER1 | 1st column | OBJECT | NVARCHAR(10) |
| | 2nd column | KEY | NVARCHAR(10) |
| | 3rd column | VALUE | NVARCHAR(10) |

### Examples

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

#### Example 1: Random Decision Trees with stratified data partition

```
SET SCHEMA DM_PAL;
DROP TABLE  PAL_DATA_TBL;
CREATE COLUMN TABLE PAL_DATA_TBL (
    "ID" INTEGER,
    "OUTLOOK" NVARCHAR(20),
    "TEMP" DOUBLE,
    "HUMIDITY" DOUBLE,
    "WINDY" NVARCHAR(10),
    "CLASS" NVARCHAR(20)
);
INSERT INTO PAL_DATA_TBL VALUES (1, 'Sunny', 75, 70.0, 'Yes', 'Play');
INSERT INTO PAL_DATA_TBL VALUES (2, 'Sunny', 80, 90.0, 'Yes', 'Do not Play');
INSERT INTO PAL_DATA_TBL VALUES (3, 'Sunny', 85, 91.0, 'No', 'Do not Play');
INSERT INTO PAL_DATA_TBL VALUES (4, 'Sunny', 72, 95.0, 'No', 'Do not Play');
INSERT INTO PAL_DATA_TBL VALUES (5, 'Sunny', 73, 70.0, 'No', 'Play');
INSERT INTO PAL_DATA_TBL VALUES (6, 'Overcast', 72.0, 90, 'Yes', 'Play');
INSERT INTO PAL_DATA_TBL VALUES (7, 'Overcast', 83.0, 78, 'No', 'Play');
INSERT INTO PAL_DATA_TBL VALUES (8, 'Overcast', 64.0, 65, 'Yes', 'Play');
INSERT INTO PAL_DATA_TBL VALUES (9, 'Overcast', 81.0, 75, 'No', 'Play');
INSERT INTO PAL_DATA_TBL VALUES (10, 'Rain', 71, 80.0, 'Yes', 'Do not Play');
INSERT INTO PAL_DATA_TBL VALUES (11, 'Rain', 65, 70.0, 'Yes', 'Do not Play');
INSERT INTO PAL_DATA_TBL VALUES (12, 'Rain', 75, 80.0, 'No', 'Play');
INSERT INTO PAL_DATA_TBL VALUES (13, 'Rain', 68, 80.0, 'No', 'Play');
INSERT INTO PAL_DATA_TBL VALUES (14, 'Rain', 70, 96.0, 'No', 'Play');
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
```

```
    "PARAM_NAME" NVARCHAR (100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR (100)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FUNCTION', NULL, NULL, 'RDT');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('KEY', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SEED', 2, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SPLIT_THRESHOLD', NULL, 0.0000001, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MIN_SAMPLES_LEAF', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('N_ESTIMATORS', 10, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MAX_DEPTH', 55, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PARTITION_METHOD', 2, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PARTITION_RANDOM_SEED', 2, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PARTITION_STRATIFIED_VARIABLE', NULL,
NULL, 'CLASS');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PARTITION_TRAINING_PERCENT', NULL, 0.7,
NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('OUTPUT_PARTITION_RESULT', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('NTILES', 2, NULL, NULL);
DROP TABLE PAL_MODEL_RDT_TBL;
CREATE COLUMN TABLE PAL_MODEL_RDT_TBL (
    "ROW_INDEX" INTEGER,
    "PART_INDEX" INTEGER,
    "MODEL_CONTENT" NCLOB
);
CALL _SYS_AFL.PAL_UNIFIED_CLASSIFICATION (PAL_DATA_TBL, #PAL_PARAMETER_TBL,
PAL_MODEL_RDT_TBL, ?, ?, ?, ?, ?, ?, ?)with overview;
```

### Expected Result

MODEL:

```
MODEL_CONTENT_
<PMMLversion="4.0"xmlns=http://www.dmg.o
```

MODEL_OUTPUT_INDEX_
rg/PMML-4_0xmlns:xs...

MODEL_FORMAT_PMML_version="4.0" xmlns="http://www.dmg.org

MODEL_FORMAT_INDEX_/PMML-4_0xmlns:xs...

MODEL_OUTPUT_3<PMMLversion="4.0"xmlns=http://www.dmg.org

MODEL_FORMAT_CONTENT_/PMML-4_0xmlns:xs...

MODEL_FORMAT_PMML_VERSION="4.0" xmlns="http://www.dmg.org/

MODEL_FORMAT_CONTENT_/PMML-4_0xmlns:xs...

MODEL_FORMAT_PMML version="4.0" xmlns="http://www.dmg.org/

MODEL_OUTPUT_/PMML-4_0xmlns:xs...

MODEL_FORMAT_PMML_version="4.0" xmlns="http://www.dmg.org

MODEL_OUTPUT_/PMML-4_0xmlns:xs...

MODEL_OUTPUT_TS_PMMLversion="4.0" xmlns="http://www.dmg.org

MODEL_OUTPUT_INDEX_/PMML-4_0xmlns:xs...

MODEL_OUTPUT_SPMML version="4.0" xmlns="http://www.dmg.org"

MODEL_FORMAT_INDEX_/PMML-4_0xmlns:xs...

MODEL_OUTPUT_SPMMLversion="4.0" xmlns="http://www.dmg.org/

MODEL_FORMAT_CONTENT_/PMML-4_0xmlns:xs...

MODEL_OUTPUT_PMML version="4.0" xmlns="http://www.dmg.org

MODEL_CONTENT_/PMML-4_0xmlns:xs...

MODEL_OUTPUT_PMML version="4.0" xmlns="http://www.dmg.org

MODEL_OUTPUT_/PMML-4_0xmlns:xs...

IMP:

VARIMPORTANCE_OUT_8204789644012944_2E3P2928802588967753

STATISTIC:

CONFUSION MATRIX:

| | PREDICTED_CLASS_Don't Play | PREDICTED_CLASS_Play | PREDICTED_CLASS_Don't Play | PREDICTED_CLASS_Play |
|---|---|---|---|---|

METRICS:

| | NAME | X | Y |
|---|---|---|---|
| 1 | RANDOM_CUMGAINS | 0 | 0 |
| 2 | RANDOM_CUMGAINS | 1 | 1 |
| 3 | RANDOM_LIFT | 0 | 1 |
| 4 | RANDOM_LIFT | 1 | 1 |
| 5 | RANDOM_CUMLIFT | 0 | 1 |
| 6 | RANDOM_CUMLIFT | 1 | 1 |
| 7 | PERF_CUMGAINS | 0 | 0 |
| 8 | PERF_CUMGAINS | 0.5 | 1 |
| 9 | PERF_CUMGAINS | 1 | 1 |
| 10 | PERF_LIFT | 0 | 2 |
| 11 | PERF_LIFT | 0.5 | 2 |
| 12 | PERF_LIFT | 1 | 0 |
| 13 | PERF_CUMLIFT | 0 | 2 |
| 14 | PERF_CUMLIFT | 0.5 | 2 |
| 15 | PERF_CUMLIFT | 1 | 1 |
| 16 | ROC_FPR | 0 | 0 |
| 17 | ROC_TPR | 0 | 0 |
| 18 | ROC_FPR | 1 | 0.25 |
| 19 | ROC_TPR | 1 | 0.5 |
| 20 | ROC_FPR | 2 | 0.25 |
| 21 | ROC_TPR | 2 | 0.75 |
| 22 | ROC_FPR | 3 | 0.5 |
| 23 | ROC_TPR | 3 | 0.75 |
| 24 | ROC_FPR | 4 | 1 |
| 25 | ROC_TPR | 4 | 1 |
| 26 | CUMGAINS | 0 | 0 |
| 27 | CUMGAINS | 0.5 | 0.75 |
| 28 | CUMGAINS | 1 | 1 |
| 29 | LIFT | 0 | 2 |
| 30 | LIFT | 0.5 | 1 |
| 31 | LIFT | 1 | 1 |
| 32 | CUMLIFT | 0 | 2 |
| 33 | CUMLIFT | 0.5 | 1.5 |
| 34 | CUMLIFT | 1 | 1 |

PARTITION_TYPE:

| |
|---|
| TYPE |
| 1 |
| 2 |
| 3 |
| 4 |
| 5 |
| 6 |
| 7 |

TYPE - 1 0 - 8 2 - 9 3 - 1 4 - 1 1 - 2 2 - 2 3 - 2 4 -

**Example 2: Support Vector Machine with user-defined data partition**

You can define how the dataset is divided into the training set and validation set by adding the purpose column and setting PARTITION_METHOD to be 1.

The purpose column can only be the last column with the values 1 for training or 2 for validation. When you set PARTITION_METHOD to 1, the function divides the dataset by taking values in the last column as your partition purpose.

> **Note**
>
> Do not add the purpose column if you do not use the user-defined partition feature.

```
SET SCHEMA DM_PAL;
DROP TABLE  PAL_DATA_TBL;
CREATE COLUMN TABLE PAL_DATA_TBL (
    "OUTLOOK" NVARCHAR(20),
    "TEMP" DOUBLE,
    "HUMIDITY" DOUBLE,
    "WINDY" NVARCHAR(10),
    "CLASS" NVARCHAR(20),
    "PURPOSE" INTEGER
);
INSERT INTO PAL_DATA_TBL VALUES ('Sunny', 75, 70.0, 'Yes', 'Play',1);
INSERT INTO PAL_DATA_TBL VALUES ('Sunny', 80, 90.0, 'Yes', 'Do not Play',1);
INSERT INTO PAL_DATA_TBL VALUES ('Sunny', 85, 91.0, 'No', 'Do not Play',2);
INSERT INTO PAL_DATA_TBL VALUES ('Sunny', 72, 95.0, 'No', 'Do not Play',1);
INSERT INTO PAL_DATA_TBL VALUES ('Sunny', 73, 70.0, 'No', 'Play',2);
INSERT INTO PAL_DATA_TBL VALUES ('Overcast', 72.0, 90, 'Yes', 'Play',1);
INSERT INTO PAL_DATA_TBL VALUES ('Overcast', 83.0, 78, 'No', 'Play',1);
INSERT INTO PAL_DATA_TBL VALUES ('Overcast', 64.0, 65, 'Yes', 'Play',2);
INSERT INTO PAL_DATA_TBL VALUES ('Overcast', 81.0, 75, 'No', 'Play',2);
INSERT INTO PAL_DATA_TBL VALUES ('Rain', 71, 80.0, 'Yes', 'Do not Play',1);
INSERT INTO PAL_DATA_TBL VALUES ('Rain', 65, 70.0, 'Yes', 'Do not Play',1);
```

```
INSERT INTO PAL_DATA_TBL VALUES ('Rain', 75, 80.0, 'No', 'Play',1);
INSERT INTO PAL_DATA_TBL VALUES ('Rain', 68, 80.0, 'No', 'Play',2);
INSERT INTO PAL_DATA_TBL VALUES ('Rain', 70, 96.0, 'No', 'Play',2);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" NVARCHAR (100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR (100)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FUNCTION', NULL, NULL, 'SVM');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PARTITION_METHOD', 1, NULL, NULL);
--parameters to be selected
INSERT INTO #PAL_PARAMETER_TBL VALUES ('RBF_GAMMA_VALUES',NULL,NULL,
'{0.01,0.05,0.07}');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SVM_C_VALUES',NULL,NULL, '{10,50,100}');
--parameter selection parameters
INSERT INTO #PAL_PARAMETER_TBL VALUES ('RESAMPLING_METHOD', NULL, NULL, 'cv');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('EVALUATION_METRIC', NULL, NULL,
'F1_SCORE');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FOLD_NUM', 2, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('REPEAT_TIMES', 2, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SEED', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PROGRESS_INDICATOR_ID', NULL, NULL,
'TEST');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PARAM_SEARCH_STRATEGY', NULL, NULL,
'grid');
CALL _SYS_AFL.PAL_UNIFIED_CLASSIFICATION (PAL_DATA_TBL,
#PAL_PARAMETER_TBL, ?, ?, ?, ?, ?, ?, ?, ?);
```

### Expected Result

MODEL:

MODEL_CONTENT_1 {"C":50.0,"CONTENT":"{\"impute_model\":{

MODEL_OWNER_VERSION_
\"column_statistics\":[{\"COLU...

STATISTICS:

| | STAT_NAME | STAT_VALUE | CLASS_NAME |
|---|---|---|---|
| 1 | RECALL | 0 | Do not Play |
| 2 | PRECISION | 0 | Do not Play |
| 3 | F1_SCORE | 0 | Do not Play |
| 4 | SUPPORT | 1 | Do not Play |

| | STAT_NAME | STAT_VALUE | CLASS_NAME |
|---|---|---|---|
| 5 | RECALL | 0.4 | Play |
| 6 | PRECISION | 0.66666666... | Play |
| 7 | F1_SCORE | 0.5 | Play |
| 8 | SUPPORT | 5 | Play |
| 9 | ACCURACY | 0.33333333... | ? |
| 10 | KAPPA | -0.3333333... | ? |
| 11 | MCC | -0.4472135... | ? |

OPTIMAL PARAMETER:

| | PARAM_NAME | INT_VALUE | DOUBLE_VALUE | STRING_VALUE |
|---|---|---|---|---|
| 1 | RBF_GAMMA | ? | 0.01 | ? |
| 2 | SVM_C | ? | 50 | ? |

CONFUSION MATRIX:

| | ACTUAL_CLASS | PREDICTED_CLASS | COUNT |
|---|---|---|---|
| 1 | Do not Play | Do not Play | 0 |
| 2 | Do not Play | Play | 1 |
| 3 | Play | Do not Play | 3 |
| 4 | Play | Play | 2 |

## _SYS_AFL. PAL_UNIFIED_CLASSIFICATION_PREDICT

This function uses the trained model to make calculated predictions or predicted values.

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA Table>` | IN |
| 2 | `<MODEL Table>` | IN |
| 3 | `<PARAMETER Table>` | IN |
| 4 | `<RESULT Table>` | OUT |

### Input Table

| Table | Column | Data Type | Description |
|---|---|---|---|
| DATA | 1st column | INTEGER, VRACHAR, or NVARCHAR | Data ID. This column must be the first column in the table. |
| | 2nd column | INTEGER, DOUBLE, DECIMAL(p,s), VARCHAR, or NVARCHAR | Independent variables. |

### Model Table

| Table | Column | Data Type |
|---|---|---|
| MODEL | 1st column | INTEGER |
| | 2nd column | INTEGER |
| | 3rd column | NCLOB |

### Parameter Table

See more details in Parameter Tables [page 1495].

### Output Table

| Table | Column | Column Name | Column Data Type | Description |
|---|---|---|---|---|
| RESULT | 1st column | ID column name | ID column data type | |
| | 2nd column | SCORE | NVARCHAR (256) | |
| | 3rd column | CONFIDENCE | DOUBLE | |
| | 4th column | REASON CODE | NCLOB | Only valid for random decision trees, decision trees, and hybrid gradient boosting trees. |
| PLACE_HOLDER | 1st column | OBJECT | NVARCHAR(10) | |
| | 2nd column | KEY | NVARCHAR(10) | |
| | 3rd column | VALUE | NVARCHAR(10) | |

### Example: Random Decision Trees prediction procedure

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_PREDICT_DATA_TBL;
CREATE COLUMN TABLE PAL_PREDICT_DATA_TBL (
    "ID" INTEGER,
    "OUTLOOK" VARCHAR(20),
    "TEMP" DOUBLE,
    "HUMIDITY" DOUBLE,
```

```
    "WINDY" VARCHAR(10)
);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES (0, 'Overcast', 75, -10000.0, 'Yes');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES (1, 'Rain', 78, 70.0, 'Yes');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES (2, 'Sunny', -10000.0, NULL, 'Yes');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES (3, 'Sunny', 69, 70.0, 'Yes');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES (4, 'Rain', NULL, 70.0, 'Yes');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES (5, NULL, 70, 70.0, 'Yes');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES (6, '***', 70, 70.0, 'Yes');
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (100)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FUNCTION', NULL, NULL, 'RDT');
CALL _SYS_AFL.PAL_UNIFIED_CLASSIFICATION_PREDICT (PAL_PREDICT_DATA_TBL,
PAL_MODEL_RDT_TBL, #PAL_PARAMETER_TBL, ?, ?);
```

### Expected Result

RESULT:

| | ID | SCORE | CONFIDENCE | REASON_CODE |
|---|---|---|---|---|
| 1 | 0 | Play | 1 | [{"attr":"HUMIDITY","pct":69.0,"val":0.23416666666666669},{"attr":"OUTL... |
| 2 | 1 | Play | 1 | [{"attr":"HUMIDITY","pct":93.0,"val":0.31611111111111112},{"attr":"OUTL... |
| 3 | 2 | Play | 0.6 | [{"attr":"TEMP","pct":42.0,"val":0.1488888888888889},{"attr":"OUTLOOK","... |
| 4 | 3 | Play | 1 | [{"attr":"HUMIDITY","pct":76.0,"val":0.33944444444444449},{"attr":"OUTL... |
| 5 | 4 | Play | 1 | [{"attr":"HUMIDITY","pct":93.0,"val":0.31611111111111112},{"attr":"OUTL... |
| 6 | 5 | Play | 0.8 | [{"attr":"HUMIDITY","pct":61.0,"val":0.37444444444444447},{"attr":"TEMP... |
| 7 | 6 | Play | 0.8 | [{"attr":"HUMIDITY","pct":61.0,"val":0.37444444444444447},{"attr":"TEMP... |

## _SYS_AFL. PAL_UNIFIED_CLASSIFICATION_SCORE

This function makes calculated predictions, and provides the evaluation statistics and metrics for the trained model.

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA Table>` | IN |
| 2 | `<MODEL Table>` | IN |
| 3 | `<PARAMETER Table>` | IN |
| 4 | `<RESULT Table>` | OUT |
| 5 | `<STATISTIC Table>` | OUT |
| 6 | `<CONFUSION_MATRIX Table>` | OUT |

| Position | Table Name | Parameter Type |
|---|---|---|
| 7 | `<METRICS table>` | OUT |

### Input Table

| Table | Column | Data Type | Description | Constraint |
|---|---|---|---|---|
| DATA | 1st column | INTEGER, VRACHAR, or NVARCHAR | Data ID. This column must be the first column in the table. | |
| | Feature column | INTEGER, DOUBLE, DECIMAL(p,s), VARCHAR, or NVARCHAR | Independent variables. | |
| | LABEL column | INTEGER, VARCHAR, or NVARCHAR | Dependent variable. | Usually this column is the last column except LABEL is set to be another column. |

### Model Table

| Table | Column | Data Type |
|---|---|---|
| MODEL | 1st column | INTEGER |
| | 2nd column | INTEGER |
| | 3rd column | NCLOB |

### Parameter Table

See details in Parameter Tables [page 1495].

### Output Table

| Table | Column | Column Name | Column Data Type | Description |
|---|---|---|---|---|
| RESULT | 1st column | ID column name | ID column data type | |
| | 2nd column | SCORE | NVARCHAR(256) | |
| | 3rd column | CONFIDENCE | DOUBLE | |
| | 4th column | REASON CODE | NCLOB | Only valid for random decision trees, decision trees, and hybrid gradient boosting tree. |
| STATISTICS | 1st column | STAT_NAME | NVARCHAR (256) | |
| | 2nd column | STAT_VALUE | NVARCHAR (1000) | |
| | 3rd column | CLASS_NAME | NVARCHAR (256) | |

| Table | Column | Column Name | Column Data Type | Description |
|---|---|---|---|---|
| CONFUSION_MATRIX | 1st column | ACTUAL_CLASS | NVARCHAR (1000) | |
| | 2nd column | PREDICTED_CLASS | NVARCHAR (1000) | |
| | 3rd column | COUNT | INTEGER | |
| METRICS | 1st column | NAME | NVARCHAR (256) | |
| | 2nd column | X | DOUBLE | |
| | 3rd column | Y | DOUBLE | |

### Example: Random Decision Trees scoring procedure

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_SCORING_DATA_TBL;
CREATE COLUMN TABLE PAL_SCORING_DATA_TBL (
    "ID" INTEGER,
    "OUTLOOK" NVARCHAR(20),
    "TEMP" DOUBLE,
    "HUMIDITY" DOUBLE,
    "WINDY" NVARCHAR(10),
    "CLASS" NVARCHAR(20)
);
INSERT INTO PAL_SCORING_DATA_TBL VALUES (0, 'Overcast', 75, -10000.0,
'Yes','Play');
INSERT INTO PAL_SCORING_DATA_TBL VALUES (1, 'Rain', 78, 70.0, 'Yes','Play');
INSERT INTO PAL_SCORING_DATA_TBL VALUES (2, 'Sunny', 63, NULL, 'Yes', 'Do not
Play');
INSERT INTO PAL_SCORING_DATA_TBL VALUES (3, 'Sunny', 69, 70.0, 'Yes','Do not
Play');
INSERT INTO PAL_SCORING_DATA_TBL VALUES (4, 'Rain', NULL, 70.0, 'Yes','Play');
INSERT INTO PAL_SCORING_DATA_TBL VALUES (5, NULL, 70, 70.0, 'Yes','Do not Play');
INSERT INTO PAL_SCORING_DATA_TBL VALUES (6, '***', 70, 70.0, 'Yes','Play');
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (100)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FUNCTION', NULL, NULL, 'RDT');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MAX_RESULT_NUM', 2, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('NTILES', 2, NULL, NULL);
CALL _SYS_AFL.PAL_UNIFIED_CLASSIFICATION_SCORE (PAL_SCORING_DATA_TBL,
PAL_MODEL_RDT_TBL, #PAL_PARAMETER_TBL, ?, ?,?,?);
```

### Expected Result

RESULT:

| | ID | SCORE | CONFIDENCE | REASON_CODE |
|---|---|---|---|---|
| 1 | 0 | Play | 1 | [{"attr":"HUMIDITY","pct":69.0,"val":0.23416666666666669},{"attr":"OUTL... |
| 2 | 1 | Play | 1 | [{"attr":"HUMIDITY","pct":93.0,"val":0.31611111111111112},{"attr":"OUTL... |

STATISTICS:

SAP PUBLIC
STATISTICAL REPORT
SP_17_2_2_2

| STATISTVAMME | 1 | 2 |
|---|---|---|
| 1 | 0 | 1 |
| 2 | 1 | 0 |

CONFUSION MATRIX:

| PREDICTED_CLASS | Donot Play | Play |
|---|---|---|
| ACTUAL | | |

PREDICTIVE_ANALYSIS_LIBRARY_PAL_Play_Play_

METRICS:

| | NAME | X | Y |
|---|---|---|---|
| 1 | RANDOM_CUMGAINS | 0 | 0 |
| 2 | RANDOM_CUMGAINS | 1 | 1 |
| 3 | RANDOM_LIFT | 0 | 1 |
| 4 | RANDOM_LIFT | 1 | 1 |
| 5 | RANDOM_CUMLIFT | 0 | 1 |
| 6 | RANDOM_CUMLIFT | 1 | 1 |
| 7 | PERF_CUMGAINS | 0 | 0 |
| 8 | PERF_CUMGAINS | 0.42... | 0.85... |
| 9 | PERF_CUMGAINS | 1 | 1 |
| 10 | PERF_LIFT | 0 | 2 |
| 11 | PERF_LIFT | 0.42... | 2 |
| 12 | PERF_LIFT | 1 | 0 |
| 13 | PERF_CUMLIFT | 0 | 2 |
| 14 | PERF_CUMLIFT | 0.42... | 2 |
| 15 | PERF_CUMLIFT | 1 | 1 |
| 16 | ROC_FPR | 0 | 0 |
| 17 | ROC_TPR | 0 | 0 |
| 18 | ROC_FPR | 1 | 0.14... |
| 19 | ROC_TPR | 1 | 0.42... |
| 20 | ROC_FPR | 2 | 0.28... |
| 21 | ROC_TPR | 2 | 0.57... |
| 22 | ROC_FPR | 3 | 0.42... |
| 23 | ROC_TPR | 3 | 0.57... |
| 24 | ROC_FPR | 4 | 0.42... |
| 25 | ROC_TPR | 4 | 0.71... |
| 26 | ROC_FPR | 5 | 0.57... |
| 27 | ROC_TPR | 5 | 0.85... |
| 28 | ROC_FPR | 6 | 1 |
| 29 | ROC_TPR | 6 | 1 |
| 30 | CUMGAINS | 0 | 0 |
| 31 | CUMGAINS | 0.42... | 0.57... |
| 32 | CUMGAINS | 1 | 1 |
| 33 | LIFT | 0 | 2 |
| 34 | LIFT | 0.42... | 1 |
| 35 | LIFT | 1 | 1 |
| 36 | CUMLIFT | 0 | 2 |
| 37 | CUMLIFT | 0.42... | 1.33... |
| 38 | CUMLIFT | 1 | 1 |

## 5.13.1.1 Parameters Specific to Unified Classification

The unified classification function supports almost all the parameters in the original classification function only except some of them disabled and some of them renamed. Besides, the unified classification function has some unique parameters that work in it only.

This topic lists all the parameters renamed or disabled from the original classification parameters and parameters unique to the unified classification function only. Refer to Parameter Tables [page 1495] for a complete list of parameters.

## _SYS_AFL.PAL_UNIFIED_CLASSIFICATION

Mandatory parameters

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| FUNCTION | VARCHAR or NVARCHAR | No default value | The parameter must be the same with the FUNCTION used to train the model.<br>• DT: decision trees<br>• HGBT: hybrid gradient boosting tree<br>• LOGR: logistic regression<br>• M_LOGR: multi-class logistic regression<br>• MLP: multilayer perception<br>• NB: naïve bayes<br>• RDT: random decision trees<br>• SVM: support vector machine |

Optional parameters

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| PARTITION_METHOD | INTEGER | 0 | Defines the way to divide the dataset:<br>• 0: no partition<br>• 1: user defined partition<br>• 2: stratified partition | |
| PARTITION_RANDOM_SEED | INTEGER | 0 | Indicates the seed used to initialize the random number generator:<br>• 0: uses the system time<br>• Not 0: uses the specified seed | Valid only when PARTITION_METHOD is set to 2. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| PARTITION_STRATIFIED_VARIABLE | VARCHAR | No default value | Indicates the column used for stratification. | Valid only when PARTITION_METHOD is set to 2. |
| PARTITION_TRAINING_PERCENT | DOUBLE | 0.8 | The percentage of data used for training.<br>Value: numbers between 0 and 1. | Valid only when PARTITION_METHOD is set to 2. |
| PARTITION_TRAINING_SIZE | INTEGER | No default value | Row size of data used for training.<br>Value: numbers larger or equal to 0. | If both TRAINING_PERCENT and TRAINING_SIZE are specified, TRAINING_PERCENT takes precedence. |
| OUTPUT_PARTITION_RESULT | INTEGER | 0 | Controls whether to output the partition result.<br>• 0: No<br>• 1: Yes | Valid only when PARTITION_METHOD is 1 or 2 and KEY is 1. |
| NTILES | INTEGER | 1 or 20 | Used to control the population tiles in metrics output.<br>The validation value should be at least 1 and no larger than the row size of the validation data.<br>**Note**<br>For AUC, this parameter means the maximum tiles. | If the row size of data for metrics evaluation is less than 20, the default value is 1; otherwise it is 20. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| BACKGROUND_SIZE | INTEGER | 0 | The row size of background data. | It should not be larger than the row size of train data.<br>Valid for the functions:<br>• Naive Bayes<br>• Support Vector Machine<br>• Multilayer Perceptron<br>• Multi-class Logistic Regression |
| BACKGROUND_SAMPLING_SEED | INTEGER | 0 | Specifies the seed for random number generator in the background samping.<br>• 0: Uses the current time (in second) as seed<br>• Others: Uses the specified value as seed | Valid for the functions:<br>• Naive Bayes<br>• Support Vector Machine<br>• Multilayer Perceptron<br>• Multi-class Logistic Regression |
| SELECTED_FEATURES | VARCHAR | No default value | A string to specify the features that will be processed. The pattern is "`X1`,...,`Xn`", where `Xi` is the corresponding column name in the data table. If this parameter is not specified, all the features will be processed. | Only used when you need to indicate the needed features. |
| HANDLE_MISSING_VALUE | INTEGER | 0 | Whether to handle missing values:<br>• 0: No<br>• 1: Yes | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| IMPUTATION_TYPE | INTEGER | 1 | Chooses the overall default imputation type for all columns:<br>• 0: NONE<br>• 1: MODE_MEAN<br>• 2: MODE_MEDIAN<br>• 3: MODE_ALL-ZERO<br>• 4: MODE_ALS<br>• 5: DELETE<br>NONE and DELETE: set all columns to imputation type NONE/DELETE.<br>MODE_MEAN, MODE_MEDIAN, MODE_ALLZERO, and MODE_ALS: set categorical columns to the first imputation type (the type before “”); set numerical columns to the second imputation type (the type after “”). ALLZERO means to fill all missing values in numerical columns with 0. | Only valid when HANDLE_MISSING_VALUE is 1. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| `<column name>`_IMPUTATION_TYPE | INTEGER | No default value | Sets imputation type of a specified column. This value overwrites the default imputation type deduced by the IMPUTATION_TYPE parameter:<br>• 0: NONE<br>• 1: DELETE<br>• 100: MODE<br>• 101: SPECIFIED_CATEGORICAL_VALUE<br>• 200: MEAN<br>• 201: MEDIAN<br>• 203:SPECIFIED_NUMERICAL_VALUE<br>• 204: ALS<br>For type 101 (SPECIFIED_CATEGORICAL_VALUE), use the following syntax to specify a categorical value to be filled in a column V0:<br>INSERT INTO #PAL_PARAMETER_TBL VALUES('V0_IMPUTATION_TYPE', 101, NULL, '`<CATEGORICAL_VALUE>`');<br>For type 203 (SPECIFIED_NUMERICAL_VALUE), use the following syntax to specify a numerical value to be filled in a column V1:<br>INSERT INTO #PAL_PARAMETER_TBL VALUES('V1_IMPUTA- | Only valid when HANDLE_MISSING_VALUE is 1. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| | | | TION_TYPE', 203, `<NUMERICAL_VALUE>`, NULL);<br><br>**Note**<br>A column of integer type can be set to type 101 (SPECIFIED_CATEGORICAL_VALUE) or type 203 (SPECIFIED_NUMERICAL_VALUE):<br>• when setting to type 101 (SPECIFIED_CATEGORICAL_VALUE), the string must contain a valid integer value, otherwise an error will be thrown;<br>• when setting to type 203 (SPECIFIED_NUMERICAL_VALUE), the double value will be converted to integer. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| ALS_FACTOR_NUMBER | INTEGER | 3 | Length of factor vectors, that is, the f in ALS model.<br>**Note**<br>ALS_FACTOR_NUMBER should be less than the number of numerical columns to get a meaningful result. | |
| ALS_REGULARIZATION | DOUBLE | 1e-2 | L2 regularization of the factors, that is, the λ in ALS model. | |
| ALS_MAX_ITERATION | INTEGER | 20 | Maximum number of iterations when training ALS model. | |
| ALS_SEED | INTEGER | 0 | Specifies the seed of the random number generator used in the training of ALS model:<br>• 0: Uses the current time as the seed<br>• Others: Uses the specified value as the seed. | |
| ALS_EXIT_THRESHOLD | DOUBLE | 0.0 | The training of ALS model exits if the value of cost function is not decreased more than the value of ALS_EXIT_THRESHOLD since the last check.<br>If ALS_EXIT_THRESHOLD is set to 0, there is no check and the training only exits on reaching the maximum number of iterations. Evaluations of cost function require additional calculations. You can set this parameter to 0 to avoid it. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| ALS_EXIT_INTERVAL | INTEGER | 5 | During the training of ALS model, it will calculate cost function and check the exit criteria every `<ALS_EXIT_INTERVAL>` iterations. Evaluations of cost function require additional calculations. | |
| ALS_LINEAR_SYSTEM_SOLVER | INTEGER | 0 | • 0: cholesky solver<br>• 1: cg solver (recommended when ALS_FACTOR_NUMBER is large) | |
| ALS_CG_MAX_ITERATION | INTEGER | 3 | Maximum number of iteration of cg solver. | |
| ALS_CENTERING | INTEGER | 1 | Whether to center the data by column before training ALS model.<br>• 0: no centering<br>• 1: centering. | |
| ALS_SCALING | INTEGER | 1 | Whether to scale the data by column before training ALS model.<br>• 0: no scaling<br>• 1: scaling | |

### Decision Trees

| ORIGINAL PARAMETERS | RENAMED/DISABLED PARAMETERS |
|---|---|
| ALLOW_MISSING_DEPENDENT | ALLOW_MISSING_LABEL |
| MIN_RECORDS_OF_PARENT | MIN_RECORDS_PARENT |
| MIN_RECORDS_OF_PARENT_VALUES | MIN_RECORDS_PARENT_VALUES |
| MIN_RECORDS_OF_PARENT_RANGE | MIN_RECORDS_PARENT_RANGE |
| MIN_RECORDS_OF_LEAF | MIN_RECORDS_LEAF |
| MIN_RECORDS_OF_LEAF_VALUES | MIN_RECORDS_LEAF_VALUES |
| MIN_RECORDS_OF_LEAF_RANGE | MIN_RECORDS_LEAF_RANGE |
| HAS_ID | KEY |
| DEPENDENT_VARIABLE | LABEL |
| IS_OUTPUT_RULES | Disabled |

| ORIGINAL PARAMETERS | RENAMED/DISABLED PARAMETERS |
| --- | --- |
| IS_OUTPUT_CONFUSION_MATRIX | Disabled |

### HYBRID GRADIENT BOOSTING TREE

| ORIGINAL PARAMETERS | RENAMED/DISABLED PARAMETERS |
| --- | --- |
| COL_SAMPLE_RATE_BYSPLIT | COL_SUBSAMPLE_SPLIT |
| COL_SAMPLE_RATE_BYSPLIT_RANGE | COL_SUBSAMPLE_SPLIT_RANGE |
| COL_SUBSAMPLE_SPLIT_VALUES | COL_SUBSAMPLE_SPLIT_VALUES |
| COL_SAMPLE_RATE_BYTREE | COL_SUBSAMPLE_TREE |
| COL_SAMPLE_RATE_BYTREE_RANGE | COL_SUBSAMPLE_TREE_RANGE |
| COL_SAMPLE_RATE_BYTREE_VALUES | COL_SUBSAMPLE_TREE_VALUES |
| ETA | LEARNING_RATE |
| ETA_RANGE | LEARNING_RATE_RANGE |
| ETA_VALUES | LEARNING_RATE_VALUES |
| GAMMA | SPLIT_THRESHOLD |
| GAMMA_RANGE | SPLIT_THRESHOLD_RANGE |
| GAMMA_VALUES | SPLIT_THRESHOLD_VALUES |
| ITER_NUM | N_ESTIMATORS |
| ITER_NUM_RANGE | N_ESTIMATORS_RANGE |
| ITER_NUM_VALUES | N_ESTIMATORS_VALUES |
| MIN_CHILD_HESSIAN | MIN_SAMPLES_WEIGHT_LEAF |
| MIN_CHILD_HESSIAN_RANGE | MIN_SAMPLES_WEIGHT_LEAF_RANGE |
| MIN_CHILD_HESSIAN_VALUES | MIN_SAMPLES_WEIGHT_LEAF_VALUES |
| NODE_SIZE | MIN_SAMPLES_LEAF |
| NODE_SIZE_RANGE | MIN_SAMPLES_LEAF_RANGE |
| NODE_SIZE_VALUES | MIN_SAMPLES_LEAF_VALUES |
| NODE_WEIGHT_CONSTRAINT | MAX_W_IN_SPLIT |
| NODE_WEIGHT_CONSTRAINT_RANGE | MAX_W_IN_SPLIT_RANGE |
| NODE_WEIGHT_CONSTRAINT_VALUES | MAX_W_IN_SPLIT_VALUES |
| ROW_SAMPLE_RATE | SUBSAMPLE |
| ROW_SAMPLE_RATE_RANGE | SUBSAMPLE_RANGE |
| ROW_SAMPLE_RATE_VALUES | SUBSAMPLE_VALUES |
| SKETCH_EPS | SKETCH_ESP |
| HAS_ID | KEY |
| DEPENDENT_VARIABLE | LABEL |

| ORIGINAL PARAMETERS | RENAMED/DISABLED PARAMETERS |
|---|---|
| CALCULATE_IMPORTANCE | Disabled |
| CALCULATE_CONFUSION_MATRIX | Disabled |

## MULTI-CLASS LOGISTIC REGRESSION

| ORIGINAL PARAMETERS | RENAMED/DISABLED PARAMETERS |
|---|---|
| ENET_ALPHA | ALPHA |
| ENET_ALPHA_RANGE | ALPHA_RANGE |
| ENET_ALPHA_VALUES | ALPHA_VALUES |
| ENET_LAMBDA | LAMB |
| ENET_LAMBDA_RANGE | LAMB_RANGE |
| ENET_LAMBDA_VALUES | LAMB_VALUES |
| EXIT_THRESHOLD | TOL |
| MAX_ITERATION | MAX_ITER |
| HAS_ID | KEY |
| DEPENDENT_VARIABLE | LABEL |
| PMML_EXPORT | Disabled |

## LOGISTIC REGRESSION

| ORIGINAL PARAMETERS | RENAMED/DISABLED PARAMETERS |
|---|---|
| ENET_ALPHA | ALPHA |
| ENET_ALPHA_RANGE | ALPHA_RANGE |
| ENET_ALPHA_VALUES | ALPHA_VALUES |
| ENET_LAMBDA | LAMB |
| ENET_LAMBDA_RANGE | LAMB_RANGE |
| ENET_LAMBDA_VALUES | LAMB_VALUES |
| EXIT_THRESHOLD | TOL |
| MAX_ITERATION | MAX_ITER |
| HAS_ID | KEY |
| DEPENDENT_VARIABLE | LABEL |
| PMML_EXPORT | Disabled |

## MULTILAYER PERCEPTRON

| ORIGINAL PARAMETERS | RENAMED/DISABLED PARAMETERS |
|---|---|
| HIDDEN_LAYER_ACTIVE_FUNC | ACTIVATION |

| ORIGINAL PARAMETERS | RENAMED/DISABLED PARAMETERS |
|---|---|
| HIDDEN_LAYER_ACTIVE_FUNC_VALUES | ACTIVATION_OPTIONS |
| HIDDEN_LAYER_SIZE_VALUES | HIDDEN_LAYER_SIZE_OPTIONS |
| MAX_ITERATION | MAX_ITER |
| MINI_BATCH_SIZE | BATCH_SIZE |
| MINI_BATCH_SIZE_VALUES | BATCH_SIZE_VALUES |
| MINI_BATCH_SIZE_RANGE | BATCH_SIZE_RANGE |
| MOMENTUM_FACTOR | MOMENTUM |
| MOMENTUM_FACTOR_VALUES | MOMENTUM_VALUES |
| MOMENTUM_FACTOR_RANGE | MOMENTUM_RANGE |
| OUTPUT_LAYER_ACTIVE_FUNC | OUTPUT_ACTIVATION |
| OUTPUT_LAYER_ACTIVE_FUNC_VALUES | OUTPUT_ACTIVATION_OPTIONS |
| HAS_ID | KEY |
| DEPENDENT_VARIABLE | LABEL |
| FUNCTIONALITY | Disabled |

### NAÏVE BAYES

| ORIGINAL PARAMETERS | RENAMED PARAMETERS |
|---|---|
| LAPLACE | ALPHA |
| LAPLACE_RANGE | ALPHA_RANGE |
| LAPLACE_VALUES | ALPHA_VALUES |
| HAS_ID | KEY |
| DEPENDENT_VARIABLE | LABEL |

### RANDOM DECISION TREES

| ORIGINAL PARAMETERS | RENAMED/DISABLED PARAMETERS |
|---|---|
| ALLOW_MISSING_DEPENDENT | ALLOW_MISSING_LABEL |
| NODE_SIZE | MIN_SAMPLES_LEAF |
| TREES_NUM | N_ESTIMATORS |
| TRY_NUM | MAX_FEATURES |
| HAS_ID | KEY |
| DEPENDENT_VARIABLE | LABEL |
| CALCULATE_OOB | Disabled |

### SUPPORT VECTOR MACHINE

| ORIGINAL PARAMETERS | RENAMED/DISABLED PARAMETERS |
|---|---|
| ERROR_TOL | TOL |
| HAS_ID | KEY |
| DEPENDENT_VARIABLE | LABEL |
| TYPE | Disabled |

## _SYS_AFL. PAL_UNIFIED_CLASSIFICATION_PREDICT

The unified classification prediction procedure supports all the parameters in the prediction procedure of the original PAL functions except the VERBOSE_OUTPUT parameter in some functions is renamed to VERBOSE.

Mandatory parameters

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| FUNCTION | VARCHAR or NVARCHAR | No default value | The parameter must be the same with the FUCTION used to train the model:<br>• DT: decision trees<br>• HGBT: hybrid gradient boosting tree<br>• LOGR: logistic regression<br>• M_LOGR: multi-class logistic regression<br>• MLP: multilayer perception<br>• NB: naïve bayes<br>• RDT: random decision trees<br>• SVM: support vector machine |

Optional Parameters

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| FEATURE_ATTRIBUTION_METHOD | INTEGER | 2 | Controls which method to use in model reasoning:<br>• 0: no reasoning<br>• 1: SAABAS<br>• 2: SHAP | Valid only for these functions to generate the reason code:<br>• Decision Trees<br>• Random Decision Trees<br>• Hybrid Gradient Boosting Tree |
| TOP_K_ATTRIBUTIONS | INTEGER | 10 | Outputs the attributions of top k features which contribute the most. | |
| SAMPLESIZE | INTEGER | 0 | Number of sampled combinations of features. If it equals to 0, it is determined by algorithm heuristically. | It is better to user a number which is larger than the number of the columns of the data table.<br>Available for the following functions:<br>• Naive Bayes<br>• Support Vector Machine<br>• Multilayer Perceptron<br>• Multi-class Logistic Regression |
| SEED | INTEGER | 0 | Specifies the seed for random number generator.<br>• 0: Uses the current time (in second) as seed<br>• Others: Uses the specified value as seed | Available for the following functions:<br>• Naive Bayes<br>• Support Vector Machine<br>• Multilayer Perceptron<br>• Multi-class Logistic Regression |
| HANDLE_MISSING_VALUE | INTEGER | 0 | Whether to handle missing values:<br>• 0: No<br>• 1: Yes | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| IMPUTATION_TYPE | INTEGER | 1 | Chooses the overall default imputation type for all columns:<br>• 0: NONE<br>• 1: MODE_MEAN<br>• 2: MODE_MEDIAN<br>• 3: MODE_ALL-ZERO<br>• 4: MODE_ALS<br>• 5: DELETE<br>NONE and DELETE: set all columns to imputation type NONE/DELETE.<br>MODE_MEAN, MODE_MEDIAN, MODE_ALLZERO, and MODE_ALS: set categorical columns to the first imputation type (the type before “”); set numerical columns to the second imputation type (the type after “”). ALLZERO means to fill all missing values in numerical columns with 0. | Only valid when HANDLE_MISSING_VALUE is 1. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| `<column name>`_IMPUTATION_TYPE | INTEGER | No default value | Sets imputation type of a specified column. This value overwrites the default imputation type deduced by the IMPUTATION_TYPE parameter:<br>• 0: NONE<br>• 1: DELETE<br>• 100: MODE<br>• 101: SPECIFIED_CATEGORICAL_VALUE<br>• 200: MEAN<br>• 201: MEDIAN<br>• 203:SPECIFIED_NUMERICAL_VALUE<br>• 204: ALS<br>For type 101 (SPECIFIED_CATEGORICAL_VALUE), use the following syntax to specify a categorical value to be filled in a column V0:<br>INSERT INTO #PAL_PARAMETER_TBL VALUES('V0_IMPUTATION_TYPE', 101, NULL, '`<CATEGORICAL_VALUE>`');<br>For type 203 (SPECIFIED_NUMERICAL_VALUE), use the following syntax to specify a numerical value to be filled in a column V1:<br>INSERT INTO #PAL_PARAMETER_TBL VALUES('V1_IMPUTA- | Only valid when HANDLE_MISSING_VALUE is 1. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| | | | TION_TYPE', 203, `<NUMERICAL_VALUE>`, NULL);<br><br>**Note**<br>A column of integer type can be set to type 101 (SPECIFIED_CATEGORICAL_VALUE) or type 203 (SPECIFIED_NUMERICAL_VALUE):<br>• when setting to type 101 (SPECIFIED_CATEGORICAL_VALUE), the string must contain a valid integer value, otherwise an error will be thrown;<br>• when setting to type 203 (SPECIFIED_NUMERICAL_VALUE), the double value will be converted to integer. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| ALS_FACTOR_NUMBER | INTEGER | 3 | Length of factor vectors, that is, the f in ALS model.<br>**Note**<br>ALS_FACTOR_NUMBER should be less than the number of numerical columns to get a meaningful result. | |
| ALS_REGULARIZATION | DOUBLE | 1e-2 | L2 regularization of the factors, that is, the λ in ALS model. | |
| ALS_MAX_ITERATION | INTEGER | 20 | Maximum number of iterations when training ALS model. | |
| ALS_SEED | INTEGER | 0 | Specifies the seed of the random number generator used in the training of ALS model:<br>• 0: Uses the current time as the seed<br>• Others: Uses the specified value as the seed. | |
| ALS_EXIT_THRESHOLD | DOUBLE | 0.0 | The training of ALS model exits if the value of cost function is not decreased more than the value of ALS_EXIT_THRESHOLD since the last check.<br>If ALS_EXIT_THRESHOLD is set to 0, there is no check and the training only exits on reaching the maximum number of iterations. Evaluations of cost function require additional calculations. You can set this parameter to 0 to avoid it. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| ALS_EXIT_INTERVAL | INTEGER | 5 | During the training of ALS model, it will calculate cost function and check the exit criteria every `<ALS_EXIT_INTERVAL>` iterations. Evaluations of cost function require additional calculations. | |
| ALS_LINEAR_SYSTEM_SOLVER | INTEGER | 0 | • 0: cholesky solver<br>• 1: cg solver (recommended when ALS_FACTOR_NUMBER is large) | |
| ALS_CG_MAX_ITERATION | INTEGER | 3 | Maximum number of iteration of cg solver. | |
| ALS_CENTERING | INTEGER | 1 | Whether to center the data by column before training ALS model.<br>• 0: no centering<br>• 1: centering. | |
| ALS_SCALING | INTEGER | 1 | Whether to scale the data by column before training ALS model.<br>• 0: no scaling<br>• 1: scaling | |

## _SYS_AFL. PAL_UNIFIED_CLASSIFICATION_SCORE

The unified classification score procedure supports all the parameters in the prediction procedures of the original PAL functions except the VERBOSE and VERBOSE_OUTPUT parameters.

Mandatory parameters

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| FUNCTION | VARCHAR or NVARCHAR | No default value | The parameter must be the same with the FUCTION used to train the model:<br>• DT: decision trees<br>• HGBT: hybrid gradient boosting tree<br>• LOGR: logistic regression<br>• M_LOGR: multi-class logistic regression<br>• MLP: multilayer perception<br>• NB: naïve bayes<br>• RDT: random decision trees<br>• SVM: support vector machine |

Optional Parameters

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| MAX_RESULT_NUM | INTEGER | No default value | Specifies the output number of prediction results. | |
| LABEL | VARCHAR or NVARCHAR | No default value | | The setting of the parameter should be the same with the one under the training. |
| NTILES | INTEGER | 1 or 20 | Used to control the population tiles in the metrics output.<br>The value should be at least 1 and no larger than the row size of the input data.<br>**Note**<br>For AUC, this parameter means the maximum tiles. | If the row size of data for metrics evaluation is less than 20, the default value is the 1; otherwise it is 20. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| FEATURE_ATTRIBUTION_METHOD | INTEGER | 2 | Controls which method to use in model reasoning:<br>• 0: no reasoning<br>• 1: SAABAS<br>• 2: SHAP | Valid only for these functions to generate the reason code:<br>• Decision Trees<br>• Random Decision Trees<br>• Hybrid Gradient Boosting Tree |
| TOP_K_ATTRIBUTIONS | INTEGER | 10 | Outputs the attributions of top k features which contribute the most. | |
| SAMPLESIZE | INTEGER | 0 | Number of sampled combinations of features. If it equals to 0, it is determined by algorithm heuristically. | It is better to user a number which is larger than the number of the columns of the data table.<br>Available for the following functions:<br>• Naive Bayes<br>• Support Vector Machine<br>• Multilayer Perceptron<br>• Multi-class Logistic Regression |
| SEED | INTEGER | 0 | Specifies the seed for random number generator.<br>• 0: Uses the current time (in second) as seed<br>• Others: Uses the specified value as seed | Available for the following functions:<br>• Naive Bayes<br>• Support Vector Machine<br>• Multilayer Perceptron<br>• Multi-class Logistic Regression |
| HANDLE_MISSING_VALUE | INTEGER | 0 | Whether to handle missing values:<br>• 0: No<br>• 1: Yes | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| IMPUTATION_TYPE | INTEGER | 1 | Chooses the overall default imputation type for all columns:<br>• 0: NONE<br>• 1: MODE_MEAN<br>• 2: MODE_MEDIAN<br>• 3: MODE_ALL-ZERO<br>• 4: MODE_ALS<br>• 5: DELETE<br>NONE and DELETE: set all columns to imputation type NONE/DELETE.<br>MODE_MEAN, MODE_MEDIAN, MODE_ALLZERO, and MODE_ALS: set categorical columns to the first imputation type (the type before “”); set numerical columns to the second imputation type (the type after “”). ALLZERO means to fill all missing values in numerical columns with 0. | Only valid when HANDLE_MISSING_VALUE is 1. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| `<column name>`_IMPUTATION_TYPE | INTEGER | No default value | Sets imputation type of a specified column. This value overwrites the default imputation type deduced by the IMPUTATION_TYPE parameter:<br>• 0: NONE<br>• 1: DELETE<br>• 100: MODE<br>• 101: SPECIFIED_CATEGORICAL_VALUE<br>• 200: MEAN<br>• 201: MEDIAN<br>• 203:SPECIFIED_NUMERICAL_VALUE<br>• 204: ALS<br>For type 101 (SPECIFIED_CATEGORICAL_VALUE), use the following syntax to specify a categorical value to be filled in a column V0:<br>INSERT INTO #PAL_PARAMETER_TBL VALUES('V0_IMPUTATION_TYPE', 101, NULL, '`<CATEGORICAL_VALUE>`');<br>For type 203 (SPECIFIED_NUMERICAL_VALUE), use the following syntax to specify a numerical value to be filled in a column V1:<br>INSERT INTO #PAL_PARAMETER_TBL VALUES('V1_IMPUTA- | Only valid when HANDLE_MISSING_VALUE is 1. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| | | | TION_TYPE', 203, `<NUMERICAL_VALUE>`, NULL);<br><br>**Note**<br>A column of integer type can be set to type 101 (SPECIFIED_CATEGORICAL_VALUE) or type 203 (SPECIFIED_NUMERICAL_VALUE):<br>• when setting to type 101 (SPECIFIED_CATEGORICAL_VALUE), the string must contain a valid integer value, otherwise an error will be thrown;<br>• when setting to type 203 (SPECIFIED_NUMERICAL_VALUE), the double value will be converted to integer. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| ALS_FACTOR_NUMBER | INTEGER | 3 | Length of factor vectors, that is, the f in ALS model.<br>**Note**<br>ALS_FACTOR_NUMBER should be less than the number of numerical columns to get a meaningful result. | |
| ALS_REGULARIZATION | DOUBLE | 1e-2 | L2 regularization of the factors, that is, the λ in ALS model. | |
| ALS_MAX_ITERATION | INTEGER | 20 | Maximum number of iterations when training ALS model. | |
| ALS_SEED | INTEGER | 0 | Specifies the seed of the random number generator used in the training of ALS model:<br>• 0: Uses the current time as the seed<br>• Others: Uses the specified value as the seed. | |
| ALS_EXIT_THRESHOLD | DOUBLE | 0.0 | The training of ALS model exits if the value of cost function is not decreased more than the value of ALS_EXIT_THRESHOLD since the last check.<br>If ALS_EXIT_THRESHOLD is set to 0, there is no check and the training only exits on reaching the maximum number of iterations. Evaluations of cost function require additional calculations. You can set this parameter to 0 to avoid it. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| ALS_EXIT_INTERVAL | INTEGER | 5 | During the training of ALS model, it will calculate cost function and check the exit criteria every `<ALS_EXIT_INTERVAL>` iterations. Evaluations of cost function require additional calculations. | |
| ALS_LINEAR_SYSTEM_SOLVER | INTEGER | 0 | • 0: cholesky solver<br>• 1: cg solver (recommended when ALS_FACTOR_NUMBER is large) | |
| ALS_CG_MAX_ITERATION | INTEGER | 3 | Maximum number of iteration of cg solver. | |
| ALS_CENTERING | INTEGER | 1 | Whether to center the data by column before training ALS model.<br>• 0: no centering<br>• 1: centering. | |
| ALS_SCALING | INTEGER | 1 | Whether to scale the data by column before training ALS model.<br>• 0: no scaling<br>• 1: scaling | |

## 5.13.1.2 Data Partition

A data partition process is provided in the train procedure implementation. The input data can be divided into the train set and the validation set. The train set is used for generating the training model and the validation set is used to evaluate the quality of the model.

You can execute data partition by setting the following parameters.

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| PARTITION_METHOD | INTEGER | 0 | Defines the way to divide the dataset:<br>• 0: no partition<br>• 1: user defined partition<br>• 2: stratified partition | If the parameter is set to be 1, an additional purpose column in data table is needed. |
| PARTITION_RANDOM_SEED | INTEGER | 0 | Indicates the seed used to initialize the random number generator:<br>• 0: Uses the system time<br>• Not 0: Uses the specified seed | Valid only when PARTITION_METHOD is set to 2. |
| PARTITION_STRATIFIED_VARIABLE | VARCHAR | No default value | Indicates which column is used for stratification. | Valid only when PARTITION_METHOD is set to 2. |
| PARTITION_TRAINING_PERCENT | DOUBLE | 0.8 | The percentage of data used for training.<br>Value: 0 - 1 | Valid only when PARTITION_METHOD is set to 2. |
| PARTITION_TRAINING_SIZE | INTEGER | No default value | Row size of data used for training.<br>Value: ≥ 0 | If both TRAINING_PERCENT and TRAINING_SIZE are specified, TRAINING_PERCENT takes precedence. |
| OUTPUT_PARTITION_RESULT | INTEGER | 0 | Controls whether to output the partition result.<br>• 0: No<br>• 1: Yes | Valid only when PARTITION_METHOD is 1 or 2 and KEY is 1. |

## 5.13.1.3 Model Evaluation

Unified classification provides three output tables as the evaluating results of the training model: statistics table, confusion matrix table, and the metrics table. They are generated in both the train and scoring procedures. In the training procedure, the model evaluation works only if the PARTITION_METHOD is 1 or 2.

### Statistics Table

In the statistics table, the following statistics are supported:

- AUC: See Area Under Curve (AUC) [page 174] for more details.
- RECALL, PRECISION, F1-SCORE, SUPPORT: See Confusion Matrix [page 183] for more details.
- ACCURACY: The percentage of samples that are correctly classified.
- KAPPA (Cohen's kappa): Compares an observed accuracy with an expected accuracy given by a random system. KAPPA is calculated by the following equation:

  $$\text{KAPPA} = \frac{p_o - p_e}{1 - p_e}$$

  Where $p_o$ is the observed accuracy and $p_e$ is the expected accuracy. In the binary case, $p_o$ and $p_e$ can be calculated by the following equations:
  N = TP+TN+FP+FN

  $$p_o = \frac{TP+TN}{N}$$

  $$p_e = \frac{(TP+FN)*(TP+FP)-(FP+TN)*(FN+TN)}{N^2}$$

- MCC (Matthews Correlation Coefficient): In a binary case, the statistic of MCC is calculated by the following equation:

  $$\text{MCC} = \frac{TP*TN-FP*FN}{\sqrt{(TP+FP)*(TP+FN)*(TN+FP)*(TN+FN)}}$$

  Where TP, TN, FP and FN represent True Positive, True Negative, False Positive and False Negative respectively.

**Confusion Matrix Table**

The confusion matrix table provides the result of confusion matrix calculation based on the validation set. See Confusion Matrix [page 183] for more details.

**Metrics Table**

> **Note**
>
> The metrics table of Support Vector Machine is empty due to the limitations of the SVM algorithm.

In the metrics table, the following metrics are output:

- ROC: See Area Under Curve (AUC) [page 174] for more details.
- LIFT: The result of LIFT is a curve. The curve can be obtained by plotting the interval lift at several threshold of the population. The calculation formulas for interval lift are listed below:

  $$\text{Lift} = \frac{PPV}{Positive\ Rate}$$

  $$\text{PPV} = \frac{TP}{TP+FP}$$

  $$\text{Positive Rate} = \frac{TP+FN}{TP+FP+TN+FN}$$

- CUMULATIVE LIFT: The result of CUMULATIVE LIFT is a curve. The curve can be obtained by plotting the lift at several threshold of the population. The calculation formulas for lift are listed below:

$$\text{CUM_Lift} = \frac{CUM_PPV}{Positive\ Rate}$$

$$\text{CUM_PPV} = \frac{CUM_TP}{CUM_TP+CUM_FP}$$

$$\text{Positive Rate} = \frac{TP+FN}{TP+FP+TN+FN}$$

- CUMULATIVE GAINS: The result of CUMULATIVE GAINS is a curve. The curve can be obtained by plotting the true positive rate (TPR) at several threshold of the population. The calculation formula for TPR is listed below:

$$\text{TPR} = \frac{TP}{TP+FN}$$

## Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

**Example 1: Random Decision Trees Train Procedure**

```
SET SCHEMA DM_PAL;
DROP TABLE  PAL_DATA_TBL;
CREATE COLUMN TABLE PAL_DATA_TBL (
    "OUTLOOK" NVARCHAR(20),
    "TEMP" DOUBLE,
    "HUMIDITY" DOUBLE,
    "WINDY" NVARCHAR(10),
    "CLASS" NVARCHAR(20)
);
INSERT INTO PAL_DATA_TBL VALUES ('Sunny', 75, 70.0, 'Yes', 'Play');
INSERT INTO PAL_DATA_TBL VALUES ('Sunny', 80, 90.0, 'Yes', 'Do not Play');
INSERT INTO PAL_DATA_TBL VALUES ('Sunny', 85, 91.0, 'No', 'Do not Play');
INSERT INTO PAL_DATA_TBL VALUES ('Sunny', 72, 95.0, 'No', 'Do not Play');
INSERT INTO PAL_DATA_TBL VALUES ('Sunny', 73, 70.0, 'No', 'Play');
INSERT INTO PAL_DATA_TBL VALUES ('Overcast', 72.0, 90, 'Yes', 'Play');
INSERT INTO PAL_DATA_TBL VALUES ('Overcast', 83.0, 78, 'No', 'Play');
INSERT INTO PAL_DATA_TBL VALUES ('Overcast', 64.0, 65, 'Yes', 'Play');
INSERT INTO PAL_DATA_TBL VALUES ('Overcast', 81.0, 75, 'No', 'Play');
INSERT INTO PAL_DATA_TBL VALUES ('Rain', 71, 80.0, 'Yes', 'Do not Play');
INSERT INTO PAL_DATA_TBL VALUES ('Rain', 65, 70.0, 'Yes', 'Do not Play');
INSERT INTO PAL_DATA_TBL VALUES ('Rain', 75, 80.0, 'No', 'Play');
INSERT INTO PAL_DATA_TBL VALUES ('Rain', 68, 80.0, 'No', 'Play');
INSERT INTO PAL_DATA_TBL VALUES ('Rain', 70, 96.0, 'No', 'Play');
DROP TABLE PAL_PARAMETER_TBL;
CREATE  COLUMN TABLE PAL_PARAMETER_TBL (
    "PARAM_NAME" NVARCHAR (100),
```

```
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR (100)
);
INSERT INTO PAL_PARAMETER_TBL VALUES ('FUNCTION', NULL, NULL, 'RDT');
INSERT INTO PAL_PARAMETER_TBL VALUES ('SEED', 1234, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('SPLIT_THRESHOLD', NULL, 0.0000001, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('MIN_SAMPLES_LEAF', 1, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('N_ESTIMATORS', 10, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('MAX_DEPTH', 55, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('PARTITION_METHOD', 2, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('PARTITION_RANDOM_SEED', 1234, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('PARTITION_STRATIFIED_VARIABLE', NULL,
NULL, 'CLASS');
INSERT INTO PAL_PARAMETER_TBL VALUES ('PARTITION_TRAINING_PERCENT', NULL, 0.8,
NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('NTILES', 2, NULL, NULL);
DROP TABLE PAL_MODEL_RDT_TBL;
CREATE COLUMN TABLE PAL_MODEL_RDT_TBL (
    "ROW_INDEX" INTEGER,
    "PART_INDEX" INTEGER,
    "MODEL_CONTENT" NCLOB
);
DROP TABLE PAL_STAT_TBL;
CREATE COLUMN TABLE PAL_STAT_TBL (
    "STAT_NAME" NVARCHAR(256),
    "STAT_VALUE" NVARCHAR(1000),
    "CLASS_NAME" NVARCHAR(256)
);
DROP TABLE PAL_CMATRIX_TBL;
CREATE COLUMN TABLE PAL_CMATRIX_TBL (
    "ACTUAL_CLASS" NVARCHAR(256),
    "PREDICTED_CLASS" NVARCHAR(256),
    "COUNT" INTEGER
);
DROP TABLE PAL_METRICS_TBL;
CREATE COLUMN TABLE PAL_METRICS_TBL (
    "NAME" NVARCHAR(256),
    "X" DOUBLE,
    "Y" DOUBLE
);
DO
BEGIN
    lt_data = SELECT * FROM PAL_DATA_TBL;
    lt_ctrl = SELECT * FROM PAL_PARAMETER_TBL;
    CALL _SYS_AFL.PAL_UNIFIED_CLASSIFICATION (:lt_data, :lt_ctrl,
lt_model,lt_imp,lt_stat, lt_opt,lt_cm,lt_metrics,lt_partition,lt_ph1);
    INSERT INTO PAL_MODEL_RDT_TBL SELECT * FROM :lt_model;
    INSERT INTO PAL_STAT_TBL SELECT * FROM :lt_stat;
    INSERT INTO PAL_CMATRIX_TBL SELECT * FROM :lt_cm;
    INSERT INTO PAL_METRICS_TBL SELECT * FROM :lt_metrics;
END;
SELECT * FROM PAL_STAT_TBL;
SELECT * FROM PAL_CMATRIX_TBL;
SELECT * FROM PAL_METRICS_TBL;
```

### Expected Result

STATISTICS:

| | STAT_NAME | STAT_VALUE | CLASS_NAME |
|---|---|---|---|
| 1 | AUC | 0.6111111111111112 | ? |
| 2 | RECALL | 1 | Do not Play |
| 3 | PRECISION | 0.5 | Do not Play |
| 4 | F1_SCORE | 0.6666666666666666 | Do not Play |
| 5 | SUPPORT | 1 | Do not Play |
| 6 | RECALL | 0.5 | Play |
| 7 | PRECISION | 1 | Play |
| 8 | F1_SCORE | 0.6666666666666666 | Play |
| 9 | SUPPORT | 2 | Play |
| 10 | ACCURACY | 0.6666666666666666 | ? |
| 11 | KAPPA | 0.4 | ? |
| 12 | MCC | 0.5 | ? |

CONFUSION MATRIX:

| | ACTUAL_CLASS | PREDICTED_CLASS | COUNT |
|---|---|---|---|
| 1 | Do not Play | Do not Play | 1 |
| 2 | Do not Play | Play | 0 |
| 3 | Play | Do not Play | 1 |
| 4 | Play | Play | 1 |

METRICS:

| | NAME | X | Y |
|---|---|---|---|
| 1 | RANDOM_CUMGAINS | 0 | 0 |
| 2 | RANDOM_CUMGAINS | 1 | 1 |
| 3 | RANDOM_LIFT | 0 | 1 |
| 4 | RANDOM_LIFT | 1 | 1 |
| 5 | RANDOM_CUMLIFT | 0 | 1 |
| 6 | RANDOM_CUMLIFT | 1 | 1 |
| 7 | PERF_CUMGAINS | 0 | 0 |
| 8 | PERF_CUMGAINS | 0.33... | 0.66... |
| 9 | PERF_CUMGAINS | 1 | 1 |
| 10 | PERF_LIFT | 0 | 2 |
| 11 | PERF_LIFT | 0.33... | 2 |
| 12 | PERF_LIFT | 1 | 0 |
| 13 | PERF_CUMLIFT | 0 | 2 |
| 14 | PERF_CUMLIFT | 0.33... | 2 |
| 15 | PERF_CUMLIFT | 1 | 1 |
| 16 | ROC_FPR | 0 | 0 |
| 17 | ROC_TPR | 0 | 0 |
| 18 | ROC_FPR | 1 | 0 |
| 19 | ROC_TPR | 1 | 0.33... |
| 20 | ROC_FPR | 2 | 0.33... |
| 21 | ROC_TPR | 2 | 0.33... |
| 22 | ROC_FPR | 3 | 0.66... |
| 23 | ROC_TPR | 3 | 0.66... |
| 24 | ROC_FPR | 4 | 0.66... |
| 25 | ROC_TPR | 4 | 1 |
| 26 | ROC_FPR | 5 | 1 |
| 27 | ROC_TPR | 5 | 1 |
| 28 | CUMGAINS | 0 | 0 |
| 29 | CUMGAINS | 0.33... | 0.33... |
| 30 | CUMGAINS | 1 | 1 |
| 31 | LIFT | 0 | 2 |
| 32 | LIFT | 0.33... | 1 |
| 33 | LIFT | 1 | 1 |
| 34 | CUMLIFT | 0 | 2 |
| 35 | CUMLIFT | 0.33... | 1 |
| 36 | CUMLIFT | 1 | 1 |

### Example 2: Random Decision Trees Scoring Procedure

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_SCORING_DATA_TBL;
CREATE COLUMN TABLE PAL_SCORING_DATA_TBL (
    "ID" INTEGER,
    "OUTLOOK" NVARCHAR(20),
    "TEMP" DOUBLE,
    "HUMIDITY" DOUBLE,
    "WINDY" NVARCHAR(10),
    "CLASS" NVARCHAR(20)
);
INSERT INTO PAL_SCORING_DATA_TBL VALUES (0, 'Overcast', 75, -10000.0,
'Yes','Play');
INSERT INTO PAL_SCORING_DATA_TBL VALUES (1, 'Rain', 78, 70.0, 'Yes','Play');
INSERT INTO PAL_SCORING_DATA_TBL VALUES (2, 'Sunny', 63, NULL, 'Yes', 'Do not
Play');
```

```
INSERT INTO PAL_SCORING_DATA_TBL VALUES (3, 'Sunny', 69, 70.0, 'Yes','Do not
Play');
INSERT INTO PAL_SCORING_DATA_TBL VALUES (4, 'Rain', NULL, 70.0, 'Yes','Play');
INSERT INTO PAL_SCORING_DATA_TBL VALUES (5, NULL, 70, 70.0, 'Yes','Do not Play');
INSERT INTO PAL_SCORING_DATA_TBL VALUES (6, '***', 70, 70.0, 'Yes','Play');
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (100)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FUNCTION', NULL, NULL, 'RDT');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MAX_RESULT_NUM', 2, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('NTILES', 2, NULL, NULL);
CALL _SYS_AFL.PAL_UNIFIED_CLASSIFICATION_SCORE (PAL_SCORING_DATA_TBL,
PAL_MODEL_RDT_TBL, #PAL_PARAMETER_TBL, ?, ?,?,?);
```

Expected Result

STATISTICS:

| | STAT_NAME | STAT_VALUE | CLASS_NAME |
|---|---|---|---|
| 1 | AUC | 0.8367346938775511 | ? |
| 2 | RECALL | 0.3333333333333333 | Do not Play |
| 3 | PRECISION | 1 | Do not Play |
| 4 | F1_SCORE | 0.5 | Do not Play |
| 5 | SUPPORT | 3 | Do not Play |
| 6 | RECALL | 1 | Play |
| 7 | PRECISION | 0.6666666666666666 | Play |
| 8 | F1_SCORE | 0.8 | Play |
| 9 | SUPPORT | 4 | Play |
| 10 | ACCURACY | 0.7142857142857143 | ? |
| 11 | KAPPA | 0.36363636363636... | ? |
| 12 | MCC | 0.47140452079103... | ? |

CONFUSION MATRIX:

| | ACTUAL_CLASS | PREDICTED_CLASS | COUNT |
|---|---|---|---|
| 1 | Do not Play | Do not Play | 1 |
| 2 | Do not Play | Play | 2 |
| 3 | Play | Do not Play | 0 |
| 4 | Play | Play | 4 |

METRICS:

| | NAME | X | Y |
|---|---|---|---|
| 1 | RANDOM_CUMGAINS | 0 | 0 |
| 2 | RANDOM_CUMGAINS | 1 | 1 |
| 3 | RANDOM_LIFT | 0 | 1 |
| 4 | RANDOM_LIFT | 1 | 1 |
| 5 | RANDOM_CUMLIFT | 0 | 1 |
| 6 | RANDOM_CUMLIFT | 1 | 1 |
| 7 | PERF_CUMGAINS | 0 | 0 |
| 8 | PERF_CUMGAINS | 0.42... | 0.85... |
| 9 | PERF_CUMGAINS | 1 | 1 |
| 10 | PERF_LIFT | 0 | 2 |
| 11 | PERF_LIFT | 0.42... | 2 |
| 12 | PERF_LIFT | 1 | 0 |
| 13 | PERF_CUMLIFT | 0 | 2 |
| 14 | PERF_CUMLIFT | 0.42... | 2 |
| 15 | PERF_CUMLIFT | 1 | 1 |
| 16 | ROC_FPR | 0 | 0 |
| 17 | ROC_TPR | 0 | 0 |
| 18 | ROC_FPR | 1 | 0 |
| 19 | ROC_TPR | 1 | 0.14... |
| 20 | ROC_FPR | 2 | 0 |
| 21 | ROC_TPR | 2 | 0.28... |
| 22 | ROC_FPR | 3 | 0 |
| 23 | ROC_TPR | 3 | 0.42... |
| 24 | ROC_FPR | 4 | 0.57... |
| 25 | ROC_TPR | 4 | 1 |
| 26 | ROC_FPR | 5 | 0.71... |
| 27 | ROC_TPR | 5 | 1 |
| 28 | ROC_FPR | 6 | 0.85... |
| 29 | ROC_TPR | 6 | 1 |
| 30 | ROC_FPR | 7 | 1 |
| 31 | ROC_TPR | 7 | 1 |
| 32 | CUMGAINS | 0 | 0 |
| 33 | CUMGAINS | 0.42... | 0.71... |
| 34 | CUMGAINS | 1 | 1 |
| 35 | LIFT | 0 | 2 |
| 36 | LIFT | 0.42... | 2 |
| 37 | LIFT | 1 | 0 |
| 38 | CUMLIFT | 0 | 2 |
| 39 | CUMLIFT | 0.42... | 1.66... |
| 40 | CUMLIFT | 1 | 1 |

### 5.13.1.4 Local Interpretability of Models

In the process of machine learning predictive modeling, you can also view why the predicative results are made.

Based on the Shapley Values in game theory, SHAP (Shapley Additive Explanations) was proposed to explain individual predictions.

When it comes to specified machine learning model, the implementations of SHAP values have variant versions.

- For tree-based machine learning models, a tree SHAP explainer (tree SHAP) is provided. This method can approximately compute the Shapley values in polynomial time and satisfy the interpretation consistency. Additionally, an approximating method, Saabas is also implemented to calculate the attribution values, which takes only the single decision path into account and could run in a logarithmic time (over the number of nodes).
- For nonlinear and non-tree based models, for example, Support Vector Mahcine, a kernel SHAP explainer (kernel SHAP) is provided. Kernel SHAP uses samplings with a kernel to estimate the SHAP values. The samplings called background data are sampled from the train dataset in the trainining process and stored in the model table.

The following table gives an overview of the supported SHAP versions for different functions.

| FUNCTION NAME | METHOD |
|---|---|
| Decision Trees | • tree SHAP<br>• Saabas |
| Random Decision Trees | • tree SHAP<br>• Saabas |
| Hybrid Gradient Boosting Tree | • tree SHAP<br>• Saabas |
| Naive Bayes | kernel SHAP |
| Support Vector Machine | kernel SHAP |
| Multilayer Perceptron | kernel SHAP |

## _SYS_AFL.PAL_UNIFIED_CLASSIFICATION

### Parameter Table

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| BACKGROUND_SIZE | INTEGER | 0 | The row size of background data. | It should not be larger than the row size of train data. Valid for the functions:<br>• Naive Bayes<br>• Support Vector Machine<br>• Multilayer Perceptron |
| BACKGROUND_SAMPLING_SEED | INTEGER | 0 | Specifies the seed for random number generator in the background samping.<br>• 0: Uses the current time (in second) as seed<br>• Others: Uses the specified value as seed | Valid for the functions:<br>• Naive Bayes<br>• Support Vector Machine<br>• Multilayer Perceptron |

### Examples

Assume that:

- DM_PAL is a schema belonging to USER1;
- USER1 has been assigned the AFL__SYS_AFL_AFLPAL_EXECUTE or AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION role.

**Example 1: tree SHAP ( Random Decision Trees )**

```
SET SCHEMA DM_PAL;
DROP TABLE  PAL_DATA_TBL;
CREATE COLUMN TABLE PAL_DATA_TBL (
    "OUTLOOK" NVARCHAR(20),
    "TEMP" DOUBLE,
    "HUMIDITY" DOUBLE,
    "WINDY" NVARCHAR(10),
    "CLASS" NVARCHAR(20)
);
INSERT INTO PAL_DATA_TBL VALUES ('Sunny', 75, 70.0, 'Yes', 'Play');
INSERT INTO PAL_DATA_TBL VALUES ('Sunny', 80, 90.0, 'Yes', 'Do not Play');
INSERT INTO PAL_DATA_TBL VALUES ('Sunny', 85, 91.0, 'No', 'Do not Play');
INSERT INTO PAL_DATA_TBL VALUES ('Sunny', 72, 95.0, 'No', 'Do not Play');
INSERT INTO PAL_DATA_TBL VALUES ('Sunny', 73, 70.0, 'No', 'Play');
INSERT INTO PAL_DATA_TBL VALUES ('Overcast', 72.0, 90, 'Yes', 'Play');
INSERT INTO PAL_DATA_TBL VALUES ('Overcast', 83.0, 78, 'No', 'Play');
INSERT INTO PAL_DATA_TBL VALUES ('Overcast', 64.0, 65, 'Yes', 'Play');
INSERT INTO PAL_DATA_TBL VALUES ('Overcast', 81.0, 75, 'No', 'Play');
INSERT INTO PAL_DATA_TBL VALUES ('Rain', 71, 80.0, 'Yes', 'Do not Play');
INSERT INTO PAL_DATA_TBL VALUES ('Rain', 65, 70.0, 'Yes', 'Do not Play');
INSERT INTO PAL_DATA_TBL VALUES ('Rain', 75, 80.0, 'No', 'Play');
INSERT INTO PAL_DATA_TBL VALUES ('Rain', 68, 80.0, 'No', 'Play');
INSERT INTO PAL_DATA_TBL VALUES ('Rain', 70, 96.0, 'No', 'Play');
DROP TABLE #PAL_PARAMETER_TBL;
CREATE  LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" NVARCHAR (100),
    "INT_VALUE" INTEGER,
```

```
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR (100)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FUNCTION', NULL, NULL, 'RDT');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SEED', 2, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MAX_DEPTH', 10, NULL, NULL);
DROP TABLE PAL_MODEL_RDT_TBL;
CREATE COLUMN TABLE PAL_MODEL_RDT_TBL (
    "ROW_INDEX" INTEGER,
    "PART_INDEX" INTEGER,
    "MODEL_CONTENT" NCLOB
);
CALL _SYS_AFL.PAL_UNIFIED_CLASSIFICATION (PAL_DATA_TBL, #PAL_PARAMETER_TBL,
PAL_MODEL_RDT_TBL, ?, ?, ?, ?, ?, ?, ?) WITH OVERVIEW;
```

**Example 2: Background data based kernel SHAP ( Support Vector Mahcine )**

```
ET SCHEMA DM_PAL;
DROP TABLE  PAL_DATA_TBL;
CREATE COLUMN TABLE PAL_DATA_TBL (
    "OUTLOOK" NVARCHAR(20),
    "TEMP" DOUBLE,
    "HUMIDITY" DOUBLE,
    "WINDY" NVARCHAR(10),
    "CLASS" NVARCHAR(20)
);
INSERT INTO PAL_DATA_TBL VALUES ('Sunny', 75, 70.0, 'Yes', 'Play');
INSERT INTO PAL_DATA_TBL VALUES ('Sunny', 80, 90.0, 'Yes', 'Do not Play');
INSERT INTO PAL_DATA_TBL VALUES ('Sunny', 85, 91.0, 'No', 'Do not Play');
INSERT INTO PAL_DATA_TBL VALUES ('Sunny', 72, 95.0, 'No', 'Do not Play');
INSERT INTO PAL_DATA_TBL VALUES ('Sunny', 73, 70.0, 'No', 'Play');
INSERT INTO PAL_DATA_TBL VALUES ('Overcast', 72.0, 90, 'Yes', 'Play');
INSERT INTO PAL_DATA_TBL VALUES ('Overcast', 83.0, 78, 'No', 'Play');
INSERT INTO PAL_DATA_TBL VALUES ('Overcast', 64.0, 65, 'Yes', 'Play');
INSERT INTO PAL_DATA_TBL VALUES ('Overcast', 81.0, 75, 'No', 'Play');
INSERT INTO PAL_DATA_TBL VALUES ('Rain', 71, 80.0, 'Yes', 'Do not Play');
INSERT INTO PAL_DATA_TBL VALUES ('Rain', 65, 70.0, 'Yes', 'Do not Play');
INSERT INTO PAL_DATA_TBL VALUES ('Rain', 75, 80.0, 'No', 'Play');
INSERT INTO PAL_DATA_TBL VALUES ('Rain', 68, 80.0, 'No', 'Play');
INSERT INTO PAL_DATA_TBL VALUES ('Rain', 70, 96.0, 'No', 'Play');
DROP TABLE #PAL_PARAMETER_TBL;
CREATE  LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" NVARCHAR (100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR (100)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FUNCTION', NULL, NULL, 'SVM');
--parameters to be selected
INSERT INTO #PAL_PARAMETER_TBL VALUES ('RBF_GAMMA_VALUES',NULL,NULL,
'{0.01,0.05,0.07}');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SVM_C_VALUES',NULL,NULL, '{10,50,100}');
--generate background data
INSERT INTO #PAL_PARAMETER_TBL VALUES ('BACKGROUND_SIZE', 5, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('BACKGROUND_SAMPLING_SEED', 1234, NULL,
NULL);
--parameter selection parameters
INSERT INTO #PAL_PARAMETER_TBL VALUES ('RESAMPLING_METHOD', NULL, NULL, 'cv');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('EVALUATION_METRIC', NULL, NULL,
'F1_SCORE');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FOLD_NUM', 2, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('REPEAT_TIMES', 2, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SEED', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PROGRESS_INDICATOR_ID', NULL, NULL,
'TEST');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PARAM_SEARCH_STRATEGY', NULL, NULL,
'grid');
```

```
DROP TABLE PAL_MODEL_SVM_TBL;
CREATE COLUMN TABLE PAL_MODEL_SVM_TBL (
    "ROW_INDEX" INTEGER,
    "PART_INDEX" INTEGER,
    "MODEL_CONTENT" NCLOB
);
CALL _SYS_AFL.PAL_UNIFIED_CLASSIFICATION (PAL_DATA_TBL, #PAL_PARAMETER_TBL,
PAL_MODEL_SVM_TBL, ?, ?, ?, ?, ?, ?, ?) WITH OVERVIEW;
```

### Expected Result

MODEL:

| | ROW_INDEX | PART_INDEX | MODEL_CONTENT |
|---|---|---|---|
| 1 | 0 | 0 | {"C":50.0,"CONTENT":"{\"impute_model\":{\"column_statistics\":[{\"COLUMN_NAME\":\"ID\",\"IMPUTATION... |
| 2 | -10 | 1 | {"cate_vals":[{"cateVal":["Sunny","Rain","Overcast","Sunny"]},{"cateVal":["No","No","Yes","No"]},{"cateVal":["... |

## _SYS_AFL.PAL_UNIFIED_CLASSIFICATION_PREDICT

### Parameter Table

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| TOP_K_ATTRIBUTIONS | INTEGER | 10 | Outputs the attributions of top k features which contribute the most. | |
| FEATURE_ATTRIBUTION_METHOD | INTEGER | 2 | Controls which method to use in model reasoning.<br>• 0: no reasoning<br>• 1: Saabas<br>• 2: tree SHAP | Valid for the functions:<br>• Decision Trees<br>• Random Decision Trees<br>• Hybrid Gradient Boosting Tree |
| SAMPLESIZE | INTEGER | 0 | Number of sampled combinations of features. If it equals to 0, then it will be determined by algorithm heuristically. | It is better to user a number which is larger than the number of the columns of the data table. |
| SEED | INTEGER | 0 | Specifies the seed for random number generator.<br>• 0: Uses the current time (in second) as seed<br>• Others: Uses the specified value as seed | Valid for the functions:<br>• Naive Bayes<br>• Support Vector Machine<br>• Multilayer Perceptron |

### Examples

Assume that:

- DM_PAL is a schema belonging to USER1;
- USER1 has been assigned the AFL__SYS_AFL_AFLPAL_EXECUTE or AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION role.

**Example 1: tree SHAP ( Random Decision Trees )**

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_PREDICT_DATA_TBL;
CREATE COLUMN TABLE PAL_PREDICT_DATA_TBL (
    "ID" INTEGER,
    "OUTLOOK" VARCHAR(20),
    "TEMP" DOUBLE,
    "HUMIDITY" DOUBLE,
    "WINDY" VARCHAR(10)
);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES (0, 'Overcast', 75, -10000.0, 'Yes');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES (1, 'Rain', 78, 70.0, 'Yes');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES (2, 'Sunny', -10000.0, NULL, 'Yes');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES (3, 'Sunny', 69, 70.0, 'Yes');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES (4, 'Rain', NULL, 70.0, 'Yes');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES (5, NULL, 70, 70.0, 'Yes');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES (6, '***', 70, 70.0, 'Yes');
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (100)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FUNCTION', NULL, NULL, 'RDT');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FEATURE_ATTRIBUTION_METHOD', 2, NULL,
NULL);
CALL _SYS_AFL.PAL_UNIFIED_CLASSIFICATION_PREDICT (PAL_PREDICT_DATA_TBL,
PAL_MODEL_RDT_TBL, #PAL_PARAMETER_TBL, ?, ?);
```

**Expected Result**

RESULT:

| | ID | SCORE | CONFIDENCE | REASON_CODE |
|---|---|---|---|---|
| 1 | 0 | Play | 0.88 | [{"attr":"OUTLOOK","pct":39.0,"val":-0.14659878547378546},{"attr":"HUMI... |
| 2 | 1 | Play | 0.68 | [{"attr":"HUMIDITY","pct":45.0,"val":-0.12520580147365862},{"attr":"WIN... |
| 3 | 2 | Do n... | 0.66 | [{"attr":"OUTLOOK","pct":40.0,"val":-0.10814982239982243},{"attr":"HUMI... |
| 4 | 3 | Play | 0.7 | [{"attr":"HUMIDITY","pct":47.0,"val":-0.16330387271458693},{"attr":"OUT... |
| 5 | 4 | Play | 0.6 | [{"attr":"HUMIDITY","pct":36.0,"val":-0.11016680607752036},{"attr":"WIN... |
| 6 | 5 | Play | 0.62 | [{"attr":"OUTLOOK","pct":54.0,"val":-0.17987860750360751},{"attr":"HUMI... |
| 7 | 6 | Play | 0.8 | [{"attr":"OUTLOOK","pct":54.0,"val":-0.17987860750360751},{"attr":"HUMI... |

**Example 2: Background data based kernel SHAP ( Support Vector Machine )**

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_PREDICT_DATA_TBL;
CREATE COLUMN TABLE PAL_PREDICT_DATA_TBL (
    "ID" INTEGER,
    "OUTLOOK" VARCHAR(20),
    "TEMP" DOUBLE,
    "HUMIDITY" DOUBLE,
    "WINDY" VARCHAR(10)
);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES (0, 'Overcast', 75, -10000.0, 'Yes');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES (1, 'Rain', 78, 70.0, 'Yes');
```

```
INSERT INTO PAL_PREDICT_DATA_TBL VALUES (2, 'Sunny', -10000.0, NULL, 'Yes');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES (3, 'Sunny', 69, null, 'Yes');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES (4, 'Rain', NULL, 70.0, 'Yes');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES (5, NULL, 70, 70.0, 'Yes');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES (6, '***', 70, 70.0, 'Yes');
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (100)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FUNCTION', NULL, NULL, 'SVM');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SEED', 1234, NULL, NULL);
CALL _SYS_AFL.PAL_UNIFIED_CLASSIFICATION_PREDICT (PAL_PREDICT_DATA_TBL,
PAL_MODEL_SVM_TBL, #PAL_PARAMETER_TBL, ?, ?);
```

### Expected Result

RESULT:

| | ID | SCORE | CONFIDENCE | REASON_CODE |
|---|---|---|---|---|
| 1 | 0 | Play | 0.5523881435072807 | [{"attr":"OUTLOOK","val":-0.07581587883586852,"pct":58.0},{"attr":"HUMI... |
| 2 | 1 | Play | 0.5752462038295143 | [{"attr":"HUMIDITY","val":-0.0954393307068418,"pct":46.9},{"attr":"WIND... |
| 3 | 2 | Play | 0.5523881435072807 | [{"attr":"TEMP","val":-0.09915666757459749,"pct":55.9},{"attr":"WINDY","... |
| 4 | 3 | Do n... | 0.27983967576039... | [{"attr":"WINDY","val":-0.06485851785533359,"pct":48.1},{"attr":"OUTLO... |
| 5 | 4 | Play | 0.622402647297219 | [{"attr":"HUMIDITY","val":-0.08120410786894798,"pct":49.6},{"attr":"WIN... |
| 6 | 5 | Play | 0.630031235619437 | [{"attr":"HUMIDITY","val":-0.10385686567797867,"pct":46.9},{"attr":"WIN... |
| 7 | 6 | Play | 0.603788580661892 | [{"attr":"HUMIDITY","val":-0.08515713859399454,"pct":41.6},{"attr":"WIN... |

## _SYS_AFL.PAL_UNIFIED_CLASSIFICATION_SCORE

### Parameter Table

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| TOP_K_ATTRIBUTIONS | INTEGER | 10 | Outputs the attributions of top k features which contribute the most. | |
| FEATURE_ATTRIBUTION_METHOD | INTEGER | 2 | Controls which method to use in model reasoning.<br>• 0: no reasoning<br>• 1: Saabas<br>• 2: tree SHAP | Valid for the functions:<br>• Decision Trees<br>• Random Decision Trees<br>• Hybrid Gradient Boosting Tree |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| SAMPLESIZE | INTEGER | 0 | Number of sampled combinations of features. If it equals to 0, then it will be determined by algorithm heuristically. | It is better to user a number which is larger than the number of the columns of the data table. |
| SEED | INTEGER | 0 | Specifies the seed for random number generator.<br>• 0: Uses the current time (in second) as seed<br>• Others: Uses the specified value as seed | Valid for the functions:<br>• Naive Bayes<br>• Support Vector Machine<br>• Multilayer Perceptron |

### Examples

Assume that:

- DM_PAL is a schema belonging to USER1;
- USER1 has been assigned the AFL__SYS_AFL_AFLPAL_EXECUTE or AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION role.

**Example 1: tree SHAP ( Random Decision Trees )**

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_SCORING_DATA_TBL;
CREATE COLUMN TABLE PAL_SCORING_DATA_TBL (
    "ID" INTEGER,
    "OUTLOOK" NVARCHAR(20),
    "TEMP" DOUBLE,
    "HUMIDITY" DOUBLE,
    "WINDY" NVARCHAR(10),
    "CLASS" NVARCHAR(20)
);
INSERT INTO PAL_SCORING_DATA_TBL VALUES (0, 'Overcast', 75, -10000.0,
'Yes','Play');
INSERT INTO PAL_SCORING_DATA_TBL VALUES (1, 'Rain', 78, 70.0, 'Yes','Play');
INSERT INTO PAL_SCORING_DATA_TBL VALUES (2, 'Sunny', 63, NULL, 'Yes', 'Do not
Play');
INSERT INTO PAL_SCORING_DATA_TBL VALUES (3, 'Sunny', 69, 70.0, 'Yes','Do not
Play');
INSERT INTO PAL_SCORING_DATA_TBL VALUES (4, 'Rain', NULL, 70.0, 'Yes','Play');
INSERT INTO PAL_SCORING_DATA_TBL VALUES (5, NULL, 70, 70.0, 'Yes','Do not Play');
INSERT INTO PAL_SCORING_DATA_TBL VALUES (6, '***', 70, 70.0, 'Yes','Play');
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (100)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FUNCTION', NULL, NULL, 'RDT');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FEATURE_ATTRIBUTION_METHOD', 1, NULL,
NULL);
CALL _SYS_AFL.PAL_UNIFIED_CLASSIFICATION_SCORE (PAL_SCORING_DATA_TBL,
PAL_MODEL_RDT_TBL, #PAL_PARAMETER_TBL, ?, ?, ?, ?);
```

### Expected Result

RESULT:

| | ID | SCORE | CONFIDENCE | REASON_CODE |
|---|---|---|---|---|
| 1 | 0 | Play | 0.88 | [{"attr":"OUTLOOK","pct":48.0,"val":-0.1777222222222222},{"attr":"HUMI... |
| 2 | 1 | Play | 0.68 | [{"attr":"HUMIDITY","pct":48.0,"val":-0.1340573315573316},{"attr":"WIND... |
| 3 | 2 | Do n... | 0.66 | [{"attr":"OUTLOOK","pct":41.0,"val":-0.13387190587190589},{"attr":"HUMI... |
| 4 | 3 | Play | 0.7 | [{"attr":"HUMIDITY","pct":48.0,"val":-0.21994766344766339},{"attr":"OUT... |
| 5 | 4 | Play | 0.6 | [{"attr":"HUMIDITY","pct":40.0,"val":-0.1173214008214008},{"attr":"WIND... |
| 6 | 5 | Play | 0.62 | [{"attr":"OUTLOOK","pct":62.0,"val":-0.21047979797979802},{"attr":"WIN... |
| 7 | 6 | Play | 0.8 | [{"attr":"HUMIDITY","pct":59.0,"val":-0.096139971139971137},{"attr":"WI... |

### Example 2: Background data based kernel SHAP ( Support Vector Machine )

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_SCORING_DATA_TBL;
CREATE COLUMN TABLE PAL_SCORING_DATA_TBL (
    "ID" INTEGER,
    "OUTLOOK" NVARCHAR(20),
    "TEMP" DOUBLE,
    "HUMIDITY" DOUBLE,
    "WINDY" NVARCHAR(10),
    "CLASS" NVARCHAR(20)
);
INSERT INTO PAL_SCORING_DATA_TBL VALUES (0, 'Overcast', 75, -10000.0,
'Yes','Play');
INSERT INTO PAL_SCORING_DATA_TBL VALUES (1, 'Rain', 78, 70.0, 'Yes','Play');
INSERT INTO PAL_SCORING_DATA_TBL VALUES (2, 'Sunny', 63, NULL, 'Yes', 'Do not
Play');
INSERT INTO PAL_SCORING_DATA_TBL VALUES (3, 'Sunny', 69, 70.0, 'Yes','Do not
Play');
INSERT INTO PAL_SCORING_DATA_TBL VALUES (4, 'Rain', NULL, 70.0, 'Yes','Play');
INSERT INTO PAL_SCORING_DATA_TBL VALUES (5, NULL, 70, 70.0, 'Yes','Do not Play');
INSERT INTO PAL_SCORING_DATA_TBL VALUES (6, '***', 70, 70.0, 'Yes','Play');
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (100)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FUNCTION', NULL, NULL, 'SVM');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MAX_RESULT_NUM', 2, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('NTILES', 2, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SEED', 1234, NULL, NULL);
CALL _SYS_AFL.PAL_UNIFIED_CLASSIFICATION_SCORE (PAL_SCORING_DATA_TBL,
PAL_MODEL_SVM_TBL, #PAL_PARAMETER_TBL, ?, ?,?,?);
```

### Expected Result

RESULT:

| | ID | SCORE | CONFIDENCE | REASON_CODE |
|---|---|---|---|---|
| 1 | 0 | Play | 0.5523881435072807 | [{"attr":"OUTLOOK","val":-0.07581587883586852,"pct":58.0},{"attr":"HUMI... |
| 2 | 1 | Play | 0.5752462038295143 | [{"attr":"HUMIDITY","val":-0.0954393307068418,"pct":46.9},{"attr":"WIND... |

## 5.13.1.5 Model Compression

The training process may cost a large size of memory. Model compression aims to reduce the size of the model with minimum loss of the accuracy.

Model compression is provided for Support Vector Machine, Random Decision Trees, and Hybrid Gradient Boosting Tree. The following tables list the parameters for using model compression in Support Vector Machine, Random Decision Trees, and Hybrid Gradient Boosting Tree respectively.

Support Vector Machine

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| COMPRESSION | INTEGER | 1 | Specifies if the model is stored in compressed format.<br>• 0 : no<br>• 1: yes | |
| MAX_BITS | INTEGER | 12 | The maximum number of bits to quantize continuous features. Equivalent to use 2^MAX_BITS bins. Must be less than 31. | Valid only when the value of COMPRESSION is 1. |
| MAX_QUANTIZATION_ITER | INTEGER | 1000 | The maximum iteration steps for quantization. | Valid only when the value of COMPRESSION is 1. |

Random Decision Trees

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| COMPRESSION | INTEGER | 0 | Specifies if the model is stored in compressed format.<br>• 0 : no<br>• 1: yes | |
| MAX_BITS | INTEGER | 12 | The maximum number of bits to quantize continuous features. Equivalent to use 2^MAX_BITS bins. Must be less than 31. | Valid only when the value of COMPRESSION is 1. |
| QUANTIZE_RATE | DOUBLE | 0.005 | Quantize a feature if the largest class frequency of which is less than QUANTIZE_RATE. | Valid only when the value of COMPRESSION is 1. |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| FITTINGS_QUANTIZATION | INTEGER | 0 | Indicate whether to quantize the fitting values.<br>• 0 : no<br>• 1: yes | Valid only when the value of COMPRESSION is 1. |

**Hybrid Gradient Boosting Tree**

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| COMPRESSION | INTEGER | 0 | Indicates whether result model should be compressed:<br>• 0. no;<br>• 1. yes. | |
| MAX_BITS | INTEGER | 12 | Maximum number of bits to quantize continuous features. Equivalent to use $2^{MAX_BITS}$ bins.<br>Must be less than 31. | Valid only when the value of COMPRESSION is 1. |

## Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

**Example: Random Decision Trees Model Compression**

```
SET SCHEMA DM_PAL;
DROP TABLE  PAL_DATA_TBL;
CREATE COLUMN TABLE PAL_DATA_TBL (
    "OUTLOOK" VARCHAR(20),
    "TEMP" DOUBLE,
    "HUMIDITY" DOUBLE,
    "WINDY" VARCHAR(10),
    "CLASS" VARCHAR(20)
);
INSERT INTO PAL_DATA_TBL VALUES ('Sunny', 75, 70.0, 'Yes', 'Play');
INSERT INTO PAL_DATA_TBL VALUES ('Sunny', 80, 90.0, 'Yes', 'Do not Play');
INSERT INTO PAL_DATA_TBL VALUES ('Sunny', 85, 91.0, 'No', 'Do not Play');
INSERT INTO PAL_DATA_TBL VALUES ('Sunny', 72, 95.0, 'No', 'Do not Play');
INSERT INTO PAL_DATA_TBL VALUES ('Sunny', 73, 70.0, 'No', 'Play');
INSERT INTO PAL_DATA_TBL VALUES ('Overcast', 72.0, 90, 'Yes', 'Play');
INSERT INTO PAL_DATA_TBL VALUES ('Overcast', 83.0, 78, 'No', 'Play');
INSERT INTO PAL_DATA_TBL VALUES ('Overcast', 64.0, 65, 'Yes', 'Play');
INSERT INTO PAL_DATA_TBL VALUES ('Overcast', 81.0, 75, 'No', 'Play');
INSERT INTO PAL_DATA_TBL VALUES ('Rain', 71, 80.0, 'Yes', 'Do not Play');
```

```
INSERT INTO PAL_DATA_TBL VALUES ('Rain', 65, 70.0, 'Yes', 'Do not Play');
INSERT INTO PAL_DATA_TBL VALUES ('Rain', 75, 80.0, 'No', 'Play');
INSERT INTO PAL_DATA_TBL VALUES ('Rain', 68, 80.0, 'No', 'Play');
INSERT INTO PAL_DATA_TBL VALUES ('Rain', 70, 96.0, 'No', 'Play');
DROP TABLE PAL_PARAMETER_TBL;
CREATE  COLUMN TABLE PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (100)
);
INSERT INTO PAL_PARAMETER_TBL VALUES ('FUNCTION', NULL, NULL, 'RDT');
INSERT INTO PAL_PARAMETER_TBL VALUES ('SEED', 2, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('N_ESTIMATORS', 200, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('MAX_FEATURES', 3, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('COMPRESSION', 1, NULL,
NULL);                      ------parameter for model compression
INSERT INTO PAL_PARAMETER_TBL VALUES ('PARTITION_METHOD', 0, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('PARTITION_STRATIFIED_VARIABLE', NULL,
NULL, 'CLASS');
INSERT INTO PAL_PARAMETER_TBL VALUES ('PARTITION_TRAINING_PERCENT', NULL, 0.7,
NULL);
DROP TABLE PAL_MODEL_TBL;
CREATE COLUMN TABLE PAL_MODEL_TBL (
    "ROW_INDEX" INTEGER,
    "PART_INDEX" INTEGER,
    "MODEL_CONTENT" NCLOB
);
DO
BEGIN
    lt_data = SELECT * FROM PAL_DATA_TBL;
    lt_ctrl = SELECT * FROM PAL_PARAMETER_TBL;
    CALL _SYS_AFL.PAL_UNIFIED_CLASSIFICATION (:lt_data, :lt_ctrl, lt_model,
lt_imp,lt_stat, lt_opt,lt_cm,lt_metrics,lt_partition,lt_ph1);
    INSERT INTO PAL_MODEL_TBL SELECT * FROM :lt_model;
END;
SELECT * FROM PAL_MODEL_TBL;
```

### Expected Result

MODEL:

| | ROW_INDEX | PART_INDEX | MODEL_CONTENT |
|---|---|---|---|
| 1 | 0 | -1 | {"atFields":[{"dataType":"string","name":"OUTLOOK","opType":"cat","valu... |
| 2 | 1 | 0 | {"cw":"XXI11F\u0015\u0015\u0014RJ>\u0010\|!IIq\u000f\bt1hbRS8\u001... |
| 3 | 2 | 1 | {"cw":"R xj$\u000f\f5\u001c\r\u0007 \u0004\u0018t\u0000C\u000bh>/!... |
| 4 | 3 | 2 | {"cw":"&,\u0011\u00190CZCZD:\u0012\u0011hl<w\u000bg8kn\u0011W... |
| 5 | 4 | 2 | 00\u0001\u0001\u0000\u0001\u0004\u0000\u0003\u0001\u0001\u0001... |
| 6 | 5 | 2 | \u0000\u0002\u0006\u0000\f\u0002\u0006\u0002\u0004\u0000\u0002\... |
| 7 | 6 | 2 | 00\u0005\u0001\u0000\u0001\u0000\u0007\u0007\u0004\u0000\u0003... |
| 8 | 7 | 3 | {"at":0,"cvs":["\u0002\u0001","\u0000","\u0001\u0000","\u0002","\u0001... |
| 9 | 8 | 3 | 6\u0007\t\n\t\n\u0006\t\u0006\t","np":"\u0002\u0002\u0002\u0002\u0... |
| 10 | 9 | 4 | {"at":1,"cvs":["77.5","72.5","84","71","66.5","64.5","68.5","80","69.5","76.5","... |
| 11 | 10 | 4 | u0002","trees":"\u0000\u0001\u0006\u0007\b\t\f\r\u000e\u000f\u0010\... |
| 12 | 11 | 5 | {"at":2,"cvs":["80","85","90.5","92.5","87.5","75","95.5","79","80.5","77.5","7... |
| 13 | 12 | 6 | {"at":3,"cvs":["\u0001","\u0000"],"dicts":[[{"p":0.5,"v":0},{"p":0.5,"v":1}],[{"p... |

## 5.13.1.6 Missing Value Handling

Missing values in each column of the table can be handled by one of the following imputation types:

- MODE: Fills all missing values by the mode of the column, that is, the value that appears most often in the column. This type only suits column with categorical variables.
- MEAN: Fills all missing values by the mean of the column, that is, the average of all non-missing values in the column. This type only suits column with numerical variables.
- MEDIAN: Fills all missing values by the median of the column, that is, the value that separates the higher half of the non-missing values from the lower half. If there is an even number of observations, then the median is given by the mean of the two middle values. This type only suits column with numerical variables.
- ALS: Fills each missing value by the value imputed by a matrix completion model trained using alternating least squares method. This type only suits column with numerical variables.
- SPECIFIED_CATEGORICAL_VALUE: Fills all missing values by a specified categorical value. This type only suits column with categorical variables.
- SPECIFIED_NUMERICAL_VALUE: Fills all missing values by a specified numerical value. This type only suits column with numerical variables.
- DELETE: Deletes all rows with missing values. The entire row in table will be deleted.
- NONE: Does nothing. Leave the column untouched.

More detailed information is provided with the guidance of Missing Value Handling algorithm (preprocessing algorithms section).

**Parameter Table**

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| HANDLE_MISS-ING_VALUE | INTEGER | 0 | Whether to handle missing values:<br>• 0: No<br>• 1: Yes | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| IMPUTATION_TYPE | INTEGER | 1 | Chooses the overall default imputation type for all columns:<br>• 0: NONE<br>• 1: MODE_MEAN<br>• 2: MODE_MEDIAN<br>• 3: MODE_ALL-ZERO<br>• 4: MODE_ALS<br>• 5: DELETE<br>NONE and DELETE: set all columns to imputation type NONE/DELETE.<br>MODE_MEAN, MODE_MEDIAN, MODE_ALLZERO, and MODE_ALS: set categorical columns to the first imputation type (the type before “”); set numerical columns to the second imputation type (the type after “”). ALLZERO means to fill all missing values in numerical columns with 0. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| `<column name>`_IMPUTATION_TYPE | INTEGER | No default value | Sets imputation type of a specified column. This value overwrites the default imputation type deduced by the IMPUTATION_TYPE parameter:<br>• 0: NONE<br>• 1: DELETE<br>• 100: MODE<br>• 101: SPECIFIED_CATEGORICAL_VALUE<br>• 200: MEAN<br>• 201: MEDIAN<br>• 203:SPECIFIED_NUMERICAL_VALUE<br>• 204: ALS<br>For type 101 (SPECIFIED_CATEGORICAL_VALUE), use the following syntax to specify a categorical value to be filled in a column V0:<br>INSERT INTO #PAL_PARAMETER_TBL VALUES('V0_IMPUTATION_TYPE', 101, NULL, '`<CATEGORICAL_VALUE>`');<br>For type 203 (SPECIFIED_NUMERICAL_VALUE), use the following syntax to specify a numerical value to be filled in a column V1:<br>INSERT INTO #PAL_PARAMETER_TBL VALUES('V1_IMPUTA- | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| | | | TION_TYPE', 203, `<NUMERICAL_VALUE>`, NULL);<br><br>**Note**<br>A column of integer type can be set to type 101 (SPECIFIED_CATEGORICAL_VALUE) or type 203 (SPECIFIED_NUMERICAL_VALUE):<br>• when setting to type 101 (SPECIFIED_CATEGORICAL_VALUE), the string must contain a valid integer value, otherwise an error will be thrown;<br>• when setting to type 203 (SPECIFIED_NUMERICAL_VALUE), the double value will be converted to integer. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| ALS_FACTOR_NUMBER | INTEGER | 3 | Length of factor vectors, that is, the f in ALS model.<br>**Note**<br>ALS_FACTOR_NUMBER should be less than the number of numerical columns to get a meaningful result. | |
| ALS_REGULARIZATION | DOUBLE | 1e-2 | L2 regularization of the factors, that is, the λ in ALS model. | |
| ALS_MAX_ITERATION | INTEGER | 20 | Maximum number of iterations when training ALS model. | |
| ALS_SEED | INTEGER | 0 | Specifies the seed of the random number generator used in the training of ALS model:<br>• 0: Uses the current time as the seed<br>• Others: Uses the specified value as the seed. | |
| ALS_EXIT_THRESHOLD | DOUBLE | 0.0 | The training of ALS model exits if the value of cost function is not decreased more than the value of ALS_EXIT_THRESHOLD since the last check.<br>If ALS_EXIT_THRESHOLD is set to 0, there is no check and the training only exits on reaching the maximum number of iterations. Evaluations of cost function require additional calculations. You can set this parameter to 0 to avoid it. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| ALS_EXIT_INTERVAL | INTEGER | 5 | During the training of ALS model, it will calculate cost function and check the exit criteria every `<ALS_EXIT_INTERVAL>` iterations. Evaluations of cost function require additional calculations. | |
| ALS_LINEAR_SYSTEM_SOLVER | INTEGER | 0 | • 0: cholesky solver<br>• 1: cg solver (recommended when ALS_FACTOR_NUMBER is large) | |
| ALS_CG_MAX_ITERATION | INTEGER | 3 | Maximum number of iteration of cg solver. | |
| ALS_CENTERING | INTEGER | 1 | Whether to center the data by column before training ALS model.<br>• 0: no centering<br>• 1: centering. | |
| ALS_SCALING | INTEGER | 1 | Whether to scale the data by column before training ALS model.<br>• 0: no scaling<br>• 1: scaling | |

### Examples

Assume that:

- DM_PAL is a schema belonging to USER1;
- USER1 has been assigned the AFL__SYS_AFL_AFLPAL_EXECUTE or AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION role.

#### Example 1: Multi-Class Logistic Regression (Training)

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_META_DATA_TBL;
CREATE COLUMN TABLE PAL_META_DATA_TBL(
    VARIABLE_NAME NVARCHAR(100),
    VARIABLE_TYPE NVARCHAR(100)
);
INSERT INTO PAL_META_DATA_TBL VALUES ('ATTR1', 'CONTINUOUS');
INSERT INTO PAL_META_DATA_TBL VALUES ('ATTR2', 'CONTINUOUS');
INSERT INTO PAL_META_DATA_TBL VALUES ('ATTR3', 'CATEGORICAL');
INSERT INTO PAL_META_DATA_TBL VALUES ('ATTR4', 'CATEGORICAL');
INSERT INTO PAL_META_DATA_TBL VALUES ('LABEL', 'TARGET');
DROP TABLE PAL_DATA_TBL;
CREATE COLUMN TABLE PAL_DATA_TBL (
```

```
    DATA_ID INTEGER,
    VARIABLE_NAME NVARCHAR(100),
    VAR_VALUE NVARCHAR(100),
    PURPOSE  INTEGER
);
INSERT INTO PAL_DATA_TBL VALUES(0,'ATTR1','1',2);
INSERT INTO PAL_DATA_TBL VALUES(0,'ATTR2','10',2);
INSERT INTO PAL_DATA_TBL VALUES(0,'ATTR3','100',2);
INSERT INTO PAL_DATA_TBL VALUES(0,'ATTR4','A',2);
INSERT INTO PAL_DATA_TBL VALUES(0,'LABEL','1',2);
INSERT INTO PAL_DATA_TBL VALUES(1,'ATTR1',1.1,1);
INSERT INTO PAL_DATA_TBL VALUES(1,'ATTR2',10.1,1);
INSERT INTO PAL_DATA_TBL VALUES(1,'ATTR3',100.1,1);
INSERT INTO PAL_DATA_TBL VALUES(1,'ATTR4','A',1);
INSERT INTO PAL_DATA_TBL VALUES(1,'LABEL',1,1);
INSERT INTO PAL_DATA_TBL VALUES(2,'ATTR1',1.2,1);
INSERT INTO PAL_DATA_TBL VALUES(2,'ATTR2',10.2,1);
INSERT INTO PAL_DATA_TBL VALUES(2,'ATTR3',100,1);
INSERT INTO PAL_DATA_TBL VALUES(2,'ATTR4','A',1);
INSERT INTO PAL_DATA_TBL VALUES(2,'LABEL',1,1);
INSERT INTO PAL_DATA_TBL VALUES(3,'ATTR1',1.3,1);
INSERT INTO PAL_DATA_TBL VALUES(3,'ATTR2',10.4,1);
INSERT INTO PAL_DATA_TBL VALUES(3,'ATTR3',100,1);
INSERT INTO PAL_DATA_TBL VALUES(3,'ATTR4','A',1);
INSERT INTO PAL_DATA_TBL VALUES(3,'LABEL',1,1);
INSERT INTO PAL_DATA_TBL VALUES(4,'ATTR1',1.2,1);
INSERT INTO PAL_DATA_TBL VALUES(4,'ATTR2',10.3,1);
INSERT INTO PAL_DATA_TBL VALUES(4,'ATTR3',100,1);
INSERT INTO PAL_DATA_TBL VALUES(4,'ATTR4','AB',1);
INSERT INTO PAL_DATA_TBL VALUES(4,'LABEL',1,1);
INSERT INTO PAL_DATA_TBL VALUES(5,'ATTR1',4,1);
INSERT INTO PAL_DATA_TBL VALUES(5,'ATTR2',40,1);
INSERT INTO PAL_DATA_TBL VALUES(5,'ATTR3',400,1);
INSERT INTO PAL_DATA_TBL VALUES(5,'ATTR4','AB',1);
INSERT INTO PAL_DATA_TBL VALUES(5,'LABEL',2,1);
INSERT INTO PAL_DATA_TBL VALUES(6,'ATTR1',4.1,2);
INSERT INTO PAL_DATA_TBL VALUES(6,'ATTR2',40,2);
INSERT INTO PAL_DATA_TBL VALUES(6,'ATTR3',400,2);
INSERT INTO PAL_DATA_TBL VALUES(6,'ATTR4','AB',2);
INSERT INTO PAL_DATA_TBL VALUES(6,'LABEL',2,2);
INSERT INTO PAL_DATA_TBL VALUES(7,'ATTR1',4.2,1);
INSERT INTO PAL_DATA_TBL VALUES(7,'ATTR2',40.2,1);
INSERT INTO PAL_DATA_TBL VALUES(7,'ATTR3',400,1);
INSERT INTO PAL_DATA_TBL VALUES(7,'ATTR4','AB',1);
INSERT INTO PAL_DATA_TBL VALUES(7,'LABEL',2,1);
INSERT INTO PAL_DATA_TBL VALUES(8,'ATTR1',4.3,1);
INSERT INTO PAL_DATA_TBL VALUES(8,'ATTR2',40.4,1);
INSERT INTO PAL_DATA_TBL VALUES(8,'ATTR3',400,1);
INSERT INTO PAL_DATA_TBL VALUES(8,'ATTR4','AB',1);
INSERT INTO PAL_DATA_TBL VALUES(8,'LABEL',2,1);
INSERT INTO PAL_DATA_TBL VALUES(9,'ATTR1',4.2,1);
INSERT INTO PAL_DATA_TBL VALUES(9,'ATTR2',40.3,1);
INSERT INTO PAL_DATA_TBL VALUES(9,'ATTR3',400,1);
INSERT INTO PAL_DATA_TBL VALUES(9,'ATTR4','AB',1);
INSERT INTO PAL_DATA_TBL VALUES(9,'LABEL',2,1);
INSERT INTO PAL_DATA_TBL VALUES(10,'ATTR1',9,1);
--INSERT INTO PAL_DATA_TBL VALUES(10,'ATTR2',90,1);
INSERT INTO PAL_DATA_TBL VALUES(10,'ATTR3',900,1);
INSERT INTO PAL_DATA_TBL VALUES(10,'ATTR4','B',1);
INSERT INTO PAL_DATA_TBL VALUES(10,'LABEL',3,1);
INSERT INTO PAL_DATA_TBL VALUES(11,'ATTR1',9.1,1);
INSERT INTO PAL_DATA_TBL VALUES(11,'ATTR2',90.1,1);
INSERT INTO PAL_DATA_TBL VALUES(11,'ATTR3',900,1);
INSERT INTO PAL_DATA_TBL VALUES(11,'ATTR4','A',1);
INSERT INTO PAL_DATA_TBL VALUES(11,'LABEL',3,1);
INSERT INTO PAL_DATA_TBL VALUES(12,'ATTR1',9.2,2);
--INSERT INTO PAL_DATA_TBL VALUES(12,'ATTR2',90.2,2);
INSERT INTO PAL_DATA_TBL VALUES(12,'ATTR3',900,2);
```

```
INSERT INTO PAL_DATA_TBL VALUES(12,'ATTR4','B',2);
INSERT INTO PAL_DATA_TBL VALUES(12,'LABEL',3,2);
INSERT INTO PAL_DATA_TBL VALUES(13,'ATTR1',9.3,1);
INSERT INTO PAL_DATA_TBL VALUES(13,'ATTR2',90.4,1);
INSERT INTO PAL_DATA_TBL VALUES(13,'ATTR3',900,1);
INSERT INTO PAL_DATA_TBL VALUES(13,'ATTR4','A',1);
INSERT INTO PAL_DATA_TBL VALUES(13,'LABEL',3,1);
INSERT INTO PAL_DATA_TBL VALUES(14,'ATTR1',9.2,1);
INSERT INTO PAL_DATA_TBL VALUES(14,'ATTR2',90.3,1);
INSERT INTO PAL_DATA_TBL VALUES(14,'ATTR3',900,1);
INSERT INTO PAL_DATA_TBL VALUES(14,'ATTR4','A',1);
INSERT INTO PAL_DATA_TBL VALUES(14,'LABEL',3,1);
DROP TABLE PAL_PARAMETER_TBL;
CREATE  COLUMN TABLE PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (100)
);
INSERT INTO PAL_PARAMETER_TBL VALUES ('FUNCTION', NULL, NULL, 'M_LOGR');
INSERT INTO PAL_PARAMETER_TBL VALUES ('PARTITION_METHOD', 2, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('HANDLE_MISSING_VALUE', 1, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('PARTITION_STRATIFIED_VARIABLE', NULL,
NULL, 'LABEL');
INSERT INTO PAL_PARAMETER_TBL VALUES ('PARTITION_TRAINING_PERCENT', NULL, 0.7,
NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('OUTPUT_PARTITION_RESULT', 1, NULL,
NULL);
DROP TABLE PAL_MODEL_TBL;
CREATE COLUMN TABLE PAL_MODEL_TBL (
    "ROW_INDEX" INTEGER,
    "TREE_INDEX" INTEGER,
    "MODEL_CONTENT" NVARCHAR(5000)
);
DROP TABLE PAL_IMP_TBL;
CREATE COLUMN TABLE PAL_IMP_TBL (
    "VARIABLE_NAME" NVARCHAR(256),
    "IMPORTANCE" DOUBLE
);
DROP TABLE PAL_STAT_TBL;
CREATE COLUMN TABLE PAL_STAT_TBL (
    "STAT_NAME" NVARCHAR(256),
    "STAT_VALUE" NVARCHAR(1000),
    "CLASS_NAME" NVARCHAR(256)
);
DROP TABLE PAL_OPT_PARAM_TBL;
CREATE  COLUMN TABLE PAL_OPT_PARAM_TBL (
    "PARAM_NAME" VARCHAR (100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (100)
);
DROP TABLE PAL_CMATRIX_TBL;
CREATE COLUMN TABLE PAL_CMATRIX_TBL (
    "ACTUAL_CLASS" NVARCHAR(256),
    "PREDICTED_CLASS" NVARCHAR(256),
    "COUNT" INTEGER
);
DROP TABLE PAL_METRICS_TBL;
CREATE COLUMN TABLE PAL_METRICS_TBL (
    "NAME" NVARCHAR(256),
    "X" DOUBLE,
    "Y" DOUBLE
);
DROP TABLE PAL_PARTITION_TBL;
CREATE COLUMN TABLE PAL_PARTITION_TBL (
    "ID" INTEGER,
    "TYPE" INTEGER
```

```
);
DO
BEGIN
    lt_meta_data = SELECT * FROM PAL_META_DATA_TBL;
    lt_data = SELECT * FROM PAL_DATA_TBL;
    lt_ctrl = SELECT * FROM PAL_PARAMETER_TBL;
    CALL _SYS_AFL.PAL_UNIFIED_CLASSIFICATION_PIVOT
(:lt_meta_data, :lt_data, :lt_ctrl, lt_model, lt_imp,lt_stat,
lt_opt,lt_cm,lt_metrics,lt_partition,lt_ph1);
    INSERT INTO PAL_MODEL_TBL SELECT * FROM :lt_model;
    INSERT INTO PAL_IMP_TBL SELECT * FROM :lt_imp;
    INSERT INTO PAL_STAT_TBL SELECT * FROM :lt_stat;
    INSERT INTO PAL_OPT_PARAM_TBL SELECT * FROM :lt_opt;
    INSERT INTO PAL_CMATRIX_TBL SELECT * FROM :lt_cm;
    INSERT INTO PAL_METRICS_TBL SELECT * FROM :lt_metrics;
    INSERT INTO PAL_PARTITION_TBL SELECT * FROM :lt_partition;
END;
SELECT * FROM PAL_MODEL_TBL;
```

**Example 2: Multi-Class Logistic Regression (Prediction)**

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_PREDICT_DATA_TBL;
CREATE COLUMN TABLE PAL_PREDICT_DATA_TBL (
    DATA_ID INTEGER,
    VARIABLE_NAME NVARCHAR(100),
    VAR_VALUE NVARCHAR(100)
);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(0,'ATTR1','1');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(0,'ATTR2','10');
--INSERT INTO PAL_PREDICT_DATA_TBL VALUES(0,'ATTR3','100');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(0,'ATTR4','A');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(1,'ATTR1',1.1);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(1,'ATTR2',10.1);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(1,'ATTR3',100.1);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(1,'ATTR4','A');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(2,'ATTR1',1.2);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(2,'ATTR2',10.2);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(2,'ATTR3',100);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(2,'ATTR4','A');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(3,'ATTR1',1.3);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(3,'ATTR2',10.4);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(3,'ATTR3',100);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(3,'ATTR4','A');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(4,'ATTR1',4);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(4,'ATTR2',40);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(4,'ATTR3',400);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(4,'ATTR4','AB');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(5,'ATTR1',9.2);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(5,'ATTR2',90.2);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(5,'ATTR3',900);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(5,'ATTR4','B');
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (100)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('VERBOSE', 0, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('HANDLE_MISSING_VALUE', 1, NULL, NULL);
CALL _SYS_AFL.PAL_UNIFIED_CLASSIFICATION_PREDICT_PIVOT(PAL_PREDICT_DATA_TBL,
PAL_MODEL_TBL, #PAL_PARAMETER_TBL, ?, ?);
```

**Example 3: Multi-Class Logistic Regression (Scoring)**

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_SCORING_DATA_TBL;
CREATE COLUMN TABLE PAL_SCORING_DATA_TBL (
    DATA_ID INTEGER,
    VARIABLE_NAME NVARCHAR(100),
    VAR_VALUE NVARCHAR(100)
);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(0,'ATTR1','1');
INSERT INTO PAL_SCORING_DATA_TBL VALUES(0,'ATTR2','10');
INSERT INTO PAL_SCORING_DATA_TBL VALUES(0,'ATTR3','100');
--INSERT INTO PAL_SCORING_DATA_TBL VALUES(0,'ATTR4','A');
INSERT INTO PAL_SCORING_DATA_TBL VALUES(0,'LABEL','1');
INSERT INTO PAL_SCORING_DATA_TBL VALUES(1,'ATTR1',1.1);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(1,'ATTR2',10.1);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(1,'ATTR3',100.1);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(1,'ATTR4','A');
INSERT INTO PAL_SCORING_DATA_TBL VALUES(1,'LABEL','1');
INSERT INTO PAL_SCORING_DATA_TBL VALUES(2,'ATTR1',1.2);
--INSERT INTO PAL_SCORING_DATA_TBL VALUES(2,'ATTR2',10.2);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(2,'ATTR3',100);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(2,'ATTR4','A');
INSERT INTO PAL_SCORING_DATA_TBL VALUES(2,'LABEL','1');
INSERT INTO PAL_SCORING_DATA_TBL VALUES(3,'ATTR1',1.3);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(3,'ATTR2',10.4);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(3,'ATTR3',100);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(3,'ATTR4','A');
INSERT INTO PAL_SCORING_DATA_TBL VALUES(3,'LABEL','1');
--INSERT INTO PAL_SCORING_DATA_TBL VALUES(4,'ATTR1',4);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(4,'ATTR2',40);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(4,'ATTR3',400);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(4,'ATTR4','AB');
INSERT INTO PAL_SCORING_DATA_TBL VALUES(4,'LABEL','2');
INSERT INTO PAL_SCORING_DATA_TBL VALUES(5,'ATTR1',9.2);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(5,'ATTR2',90.2);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(5,'ATTR3',900);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(5,'ATTR4','B');
INSERT INTO PAL_SCORING_DATA_TBL VALUES(5,'LABEL','3');
CALL _SYS_AFL.PAL_UNIFIED_CLASSIFICATION_SCORE_PIVOT(PAL_SCORING_DATA_TBL,
PAL_MODEL_TBL, #PAL_PARAMETER_TBL, ?, ?,?,?);
```

## 5.13.1.7 Compatibility

Unified classification interface and the original PAL procedures can use models getting from each other for prediction and scoring.

Unified classification interface can use the models getting from the original PAL procedures for prediction and scoring without any additional processing. However, you need to remove the second column in some unified classification interface models before using them in the original PAL procedures for prediction.

The following table lists if the model needs the remove processing.

| Model Source | Model Destination | Remove Processing |
|---|---|---|
| Unified Classification Interface | Decision Trees | Yes |
| | Logistic Regression | Yes |

| Model Source | Model Destination | Remove Processing |
|---|---|---|
| | Multi-class Logistic Regression | Yes |
| | Multilayer perceptron | Yes |
| | Naïve Bayes | Yes |
| | Support Vector Machine | Yes |
| | Random Decision Trees | No |
| | Hybrid Gradient Boosting Tree | No |

## Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

This example uses the unified classification model in the original Support Vector Machine procedure:

```
SET SCHEMA DM_PAL;
-------------------------Unified Classification Training
Procedure------------------------
DROP TABLE  PAL_DATA_TBL;
CREATE COLUMN TABLE PAL_DATA_TBL (
    "OUTLOOK" NVARCHAR(20),
    "TEMP" DOUBLE,
    "HUMIDITY" DOUBLE,
    "WINDY" NVARCHAR(10),
    "CLASS" NVARCHAR(20),
    "PURPOSE" INTEGER
);
INSERT INTO PAL_DATA_TBL VALUES ('Sunny', 75, 70.0, 'Yes', 'Play',1);
INSERT INTO PAL_DATA_TBL VALUES ('Sunny', 80, 90.0, 'Yes', 'Do not Play',1);
INSERT INTO PAL_DATA_TBL VALUES ('Sunny', 85, 91.0, 'No', 'Do not Play',2);
INSERT INTO PAL_DATA_TBL VALUES ('Sunny', 72, 95.0, 'No', 'Do not Play',1);
INSERT INTO PAL_DATA_TBL VALUES ('Sunny', 73, 70.0, 'No', 'Play',2);
INSERT INTO PAL_DATA_TBL VALUES ('Overcast', 72.0, 90, 'Yes', 'Play',1);
INSERT INTO PAL_DATA_TBL VALUES ('Overcast', 83.0, 78, 'No', 'Play',1);
INSERT INTO PAL_DATA_TBL VALUES ('Overcast', 64.0, 65, 'Yes', 'Play',2);
INSERT INTO PAL_DATA_TBL VALUES ('Overcast', 81.0, 75, 'No', 'Play',2);
INSERT INTO PAL_DATA_TBL VALUES ('Rain', 71, 80.0, 'Yes', 'Do not Play',1);
INSERT INTO PAL_DATA_TBL VALUES ('Rain', 65, 70.0, 'Yes', 'Do not Play',1);
INSERT INTO PAL_DATA_TBL VALUES ('Rain', 75, 80.0, 'No', 'Play',1);
INSERT INTO PAL_DATA_TBL VALUES ('Rain', 68, 80.0, 'No', 'Play',2);
INSERT INTO PAL_DATA_TBL VALUES ('Rain', 70, 96.0, 'No', 'Play',2);
DROP TABLE PAL_PARAMETER_TBL;
CREATE  COLUMN TABLE PAL_PARAMETER_TBL (
    "PARAM_NAME" NVARCHAR (100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR (100)
);
INSERT INTO PAL_PARAMETER_TBL VALUES ('FUNCTION', NULL, NULL, 'SVM');
INSERT INTO PAL_PARAMETER_TBL VALUES ('PARTITION_METHOD', 1, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('COMPRESSION', 0, NULL, NULL);
--parameters to be selected
```

```
INSERT INTO PAL_PARAMETER_TBL VALUES ('RBF_GAMMA_VALUES',NULL,NULL,
'{0.01,0.05,0.07}');
INSERT INTO PAL_PARAMETER_TBL VALUES ('SVM_C_VALUES',NULL,NULL, '{10,50,100}');
--parameter selection parameters
INSERT INTO PAL_PARAMETER_TBL VALUES ('RESAMPLING_METHOD', NULL, NULL, 'cv');
INSERT INTO PAL_PARAMETER_TBL VALUES ('EVALUATION_METRIC', NULL, NULL,
'F1_SCORE');
INSERT INTO PAL_PARAMETER_TBL VALUES ('FOLD_NUM', 2, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('REPEAT_TIMES', 2, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('SEED', 1, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('PROGRESS_INDICATOR_ID', NULL, NULL,
'TEST');
INSERT INTO PAL_PARAMETER_TBL VALUES ('PARAM_SEARCH_STRATEGY', NULL, NULL,
'grid');
DROP TABLE PAL_MODEL_SVM_TBL;
CREATE COLUMN TABLE PAL_MODEL_SVM_TBL (
    "ROW_INDEX" INTEGER,
    "PART_INDEX" INTEGER,
    "MODEL_CONTENT" NCLOB
);
DROP TABLE PAL_STAT_TBL;
CREATE COLUMN TABLE PAL_STAT_TBL (
    "STAT_NAME" NVARCHAR(256),
    "STAT_VALUE" NVARCHAR(1000),
    "CLASS_NAME" NVARCHAR(256)
);
DROP TABLE PAL_OPT_PARAM_TBL;
CREATE  COLUMN TABLE PAL_OPT_PARAM_TBL (
    "PARAM_NAME" VARCHAR (100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (100)
);
DROP TABLE PAL_CMATRIX_TBL;
CREATE COLUMN TABLE PAL_CMATRIX_TBL (
    "ACTUAL_CLASS" NVARCHAR(256),
    "PREDICTED_CLASS" NVARCHAR(256),
    "COUNT" INTEGER
);
DO
BEGIN
    lt_data = SELECT * FROM PAL_DATA_TBL;
    lt_ctrl = SELECT * FROM PAL_PARAMETER_TBL;
    CALL _SYS_AFL.PAL_UNIFIED_CLASSIFICATION (:lt_data, :lt_ctrl, lt_model,
lt_imp,lt_stat, lt_opt,lt_cm,lt_metrics,lt_partition,lt_ph1);
    INSERT INTO PAL_MODEL_SVM_TBL SELECT * FROM :lt_model;
    INSERT INTO PAL_STAT_TBL SELECT * FROM :lt_stat;
    INSERT INTO PAL_OPT_PARAM_TBL SELECT * FROM :lt_opt;
    INSERT INTO PAL_CMATRIX_TBL SELECT * FROM :lt_cm;
END;
-------------------------Support  Vector Machine Prediction
Procedure------------------------
DROP TABLE PAL_MODEL_SVM2_TBL;
CREATE COLUMN TABLE PAL_MODEL_SVM2_TBL (
    "ROW_INDEX" INTEGER,
    "MODEL_CONTENT" NCLOB
);
INSERT INTO PAL_MODEL_SVM2_TBL(ROW_INDEX,MODEL_CONTENT)
    SELECT ROW_INDEX, MODEL_CONTENT from PAL_MODEL_SVM_TBL;   ------------------
Remove Processing-----------------------
DROP TABLE PAL_PREDICT_DATA_TBL;
CREATE COLUMN TABLE PAL_PREDICT_DATA_TBL (
    "ID" INTEGER,
    "OUTLOOK" VARCHAR(20),
    "TEMP" DOUBLE,
    "HUMIDITY" DOUBLE,
    "WINDY" VARCHAR(10)
);
```

```
INSERT INTO PAL_PREDICT_DATA_TBL VALUES (0, 'Overcast', 75, -10000.0, 'Yes');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES (1, 'Rain', 78, 70.0, 'Yes');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES (2, 'Sunny', 63, 70, 'Yes');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES (3, 'Sunny', 69, 70.0, 'Yes');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES (4, 'Rain', 69, 70.0, 'Yes');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES (5, 'Rain', 70, 70.0, 'Yes');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES (6, 'Rain', 70, 70.0, 'Yes');
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (100)
);
CALL
_SYS_AFL.PAL_SVM_PREDICT(PAL_PREDICT_DATA_TBL,PAL_MODEL_SVM2_TBL,#PAL_PARAMETER_T
BL,?);
```

**Expected Result**

RESULT:

| | ID | SCORE | PROBABILITY |
|---|---|---|---|
| 1 | 0 | Do not Play | 0.14468152850275945 |
| 2 | 1 | Play | 0.5055492601493515 |
| 3 | 2 | Do not Play | 0.19245835216243548 |
| 4 | 3 | Do not Play | 0.31972558681631685 |
| 5 | 4 | Do not Play | 0.2767939834616818 |
| 6 | 5 | Do not Play | 0.2998325032144807 |
| 7 | 6 | Do not Play | 0.2998325032144807 |

## 5.13.1.8 Parameter Tables

This topic lists all parameters for unified classification.

### _SYS_AFL.PAL_UNIFIED_CLASSIFICATION

#### UNIFIED CLASSIFICATION

Mandatory

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION |
|---|---|---|---|
| FUNCTION | VARCHAR | No default value | • SVM: support vector machine<br>• RDT: random decision trees<br>• NB: naïve bayes<br>• MLP: multilayer perceptron<br>• M_LOGR: multi-class logistic regression<br>• LOGR: logistic regression<br>• HGBT: hybrid gradient boosting tree<br>• DT: decision trees |

Optional

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| KEY | INTEGER | 0 | Specifies if the first column of the DATA table is used as ID column:<br>• 0: No<br>• 1: Yes | |
| PARTITION_METHOD | INTEGER | 0 | Defines the way to divide the dataset:<br>• 0: no partition<br>• 1: user defined partition<br>• 2: stratified partition | |
| PARTITION_RANDOM_SEED | INTEGER | 0 | Indicates the seed used to initialize the random number generator:<br>• 0: Uses the system time<br>• Not 0: Uses the specified seed | Valid only when PARTITION_METHOD is set to 2. |
| PARTITION_STRATIFIED_VARIABLE | VARCHAR | No default value | Indicates which column is used for stratification. | Valid only when PARTITION_METHOD is set to 2. |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| PARTITION_TRAINING_PERCENT | DOUBLE | 0.8 | The percentage of data used for training.<br>Value: 0 - 1 | |
| PARTITION_TRAINING_SIZE | INTEGER | No default value | Row size of data used for training.<br>Value: ≥ 0 | If both TRAINING_PERCENT and TRAINING_SIZE are specified, TRAINING_PERCENT takes precedence. |
| OUTPUT_PARTITION_RESULT | INTEGER | 0 | Controls whether to output the partition result.<br>• 0: No<br>• 1: Yes | Valid only when PARTITION_METHOD is 1 or 2 and KEY is 1. |
| NTILES | INTEGER | 1 or 20 | Used to control the population tiles in metrics output.<br>The validation value should be at least 1 and no larger than the row size of the validation data.<br>**Note**<br>For AUC, this parameter means the maximum tiles. | If the row size of data for metrics evaluation is less than 20, the default value is 1; otherwise it is 20. |
| BACKGROUND_SIZE | INTEGER | 0 | The row size of background data. | It should not be larger than the row size of train data.<br>Valid for the functions:<br>• Naive Bayes<br>• Support Vector Machine<br>• Multilayer Perceptron<br>• Multi-class Logistic Regression |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| BACKGROUND_SAMPLING_SEED | INTEGER | 0 | Specifies the seed for random number generator in the background samping.<br>• 0: Uses the current time (in second) as seed<br>• Others: Uses the specified value as seed | Valid for the functions:<br>• Naive Bayes<br>• Support Vector Machine<br>• Multilayer Perceptron<br>• Multi-class Logistic Regression |
| SELECTED_FEATURES | VARCHAR | No default value | A string to specify the features that will be processed. The pattern is "`X1`,...,`Xn`", where `Xi` is the corresponding column name in the data table. If this parameter is not specified, all the features will be processed. | Only used when you need to indicate the needed features. |
| HANDLE_MISSING_VALUE | INTEGER | 0 | Whether to handle missing values:<br>• 0: No<br>• 1: Yes | |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| IMPUTATION_TYPE | INTEGER | 1 | Chooses the overall default imputation type for all columns:<br>• 0: NONE<br>• 1: MODE_MEAN<br>• 2: MODE_MEDIAN<br>• 3: MODE_ALL-ZERO<br>• 4: MODE_ALS<br>• 5: DELETE<br>NONE and DELETE: set all columns to imputation type NONE/DELETE.<br>MODE_MEAN, MODE_MEDIAN, MODE_ALLZERO, and MODE_ALS: set categorical columns to the first imputation type (the type before “”); set numerical columns to the second imputation type (the type after “”). ALLZERO means to fill all missing values in numerical columns with 0. | Only valid when HANDLE_MISSING_VALUE is 1. |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| <column name>_IMPUTATION_TYPE | INTEGER | No default value | Sets imputation type of a specified column. This value overwrites the default imputation type deduced by the IMPUTATION_TYPE parameter:<br>• 0: NONE<br>• 1: DELETE<br>• 100: MODE<br>• 101: SPECIFIED_CATEGORICAL_VALUE<br>• 200: MEAN<br>• 201: MEDIAN<br>• 203:SPECIFIED_NUMERICAL_VALUE<br>• 204: ALS<br>For type 101 (SPECIFIED_CATEGORICAL_VALUE), use the following syntax to specify a categorical value to be filled in a column V0:<br>INSERT INTO #PAL_PARAMETER_TBL VALUES('V0_IMPUTATION_TYPE', 101, NULL, '<CATEGORICAL_VALUE>');<br>For type 203 (SPECIFIED_NUMERICAL_VALUE), use the following syntax to specify a numerical value to be filled in a column V1:<br>INSERT INTO #PAL_PARAMETER_TBL VALUES('V1_IMPUTA- | Only valid when HANDLE_MISSING_VALUE is 1. |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| | | | TION_TYPE', 203, `<NUMERICAL_VALUE>`, NULL);<br><br>**Note**<br>A column of integer type can be set to type 101 (SPECIFIED_CATEGORICAL_VALUE) or type 203 (SPECIFIED_NUMERICAL_VALUE):<br>• when setting to type 101 (SPECIFIED_CATEGORICAL_VALUE), the string must contain a valid integer value, otherwise an error will be thrown;<br>• when setting to type 203 (SPECIFIED_NUMERICAL_VALUE), the double value will be converted to integer. | |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| ALS_FACTOR_NUMBER | INTEGER | 3 | Length of factor vectors, that is, the f in ALS model.<br>**Note**<br>ALS_FACTOR_NUMBER should be less than the number of numerical columns to get a meaningful result. | |
| ALS_REGULARIZATION | DOUBLE | 1e-2 | L2 regularization of the factors, that is, the λ in ALS model. | |
| ALS_MAX_ITERATION | INTEGER | 20 | Maximum number of iterations when training ALS model. | |
| ALS_SEED | INTEGER | 0 | Specifies the seed of the random number generator used in the training of ALS model:<br>• 0: Uses the current time as the seed<br>• Others: Uses the specified value as the seed. | |
| ALS_EXIT_THRESHOLD | DOUBLE | 0.0 | The training of ALS model exits if the value of cost function is not decreased more than the value of ALS_EXIT_THRESHOLD since the last check.<br>If ALS_EXIT_THRESHOLD is set to 0, there is no check and the training only exits on reaching the maximum number of iterations. Evaluations of cost function require additional calculations. You can set this parameter to 0 to avoid it. | |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| ALS_EXIT_INTERVAL | INTEGER | 5 | During the training of ALS model, it will calculate cost function and check the exit criteria every `<ALS_EXIT_INTERVAL>` iterations. Evaluations of cost function require additional calculations. | |
| ALS_LINEAR_SYSTEM_SOLVER | INTEGER | 0 | • 0: cholesky solver<br>• 1: cg solver (recommended when ALS_FACTOR_NUMBER is large) | |
| ALS_CG_MAX_ITERATION | INTEGER | 3 | Maximum number of iteration of cg solver. | |
| ALS_CENTERING | INTEGER | 1 | Whether to center the data by column before training ALS model.<br>• 0: no centering<br>• 1: centering. | |
| ALS_SCALING | INTEGER | 1 | Whether to scale the data by column before training ALS model.<br>• 0: no scaling<br>• 1: scaling | |

## DECISION TREES

Mandatory

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION |
|---|---|---|---|
| ALGORITHM | INTEGER | No default value | Specifies the algorithm used to grow a decision tree:<br>• 1: C45<br>• 2: CHAID<br>• 3: CART |

Optional

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| THREAD_RATIO | DOUBLE | -1 | The ratio of available threads for multi-thread tasks:<br>• 0: single thread<br>• 0–1: uses the specified percentage of available threads. PAL uses all available threads if the number is 1.0.<br>• -1: uses all of available threads. | |
| ALLOW_MISSING_LABEL | INTEGER | 1 | Specifies if missing target value is allowed:<br>• 0: Not allowed. If missing target presents, an error occurs.<br>• 1: Allowed. The datum with missing target is removed. | |
| PERCENTAGE | DOUBLE | 1.0 | Specifies the percentage of the training data used to build the tree model.<br>For example, if you set this parameter to 0.7, then 70% of the training data are used to build the tree model and 30% are used to prune the tree model. | |
| MIN_RECORDS_PARENT | INTEGER | 2 | Specifies the stop condition. If the number of records in one node is less than the specified value, the algorithm stops splitting. | |
| MIN_RECORDS_LEAF | INTEGER | 1 | Promises the minimum number of records in a leaf. | |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| MAX_DEPTH | INTEGER | Unlimited | Specifies the stop condition. If the depth of the tree model is greater than the specified value, the algorithm stops splitting. | |
| SPLIT_THRESHOLD | DOUBLE | 1e-5 for C45 and CART<br>0.05 for CHAID | Specifies the stop condition for a node:<br>• C45: The information gain ratio of the best split is less than this value.<br>• CHAID: The p-value of the best split is greater than or equal to this value.<br>• CART: The reduction of Gini index or relative MSE of the best split is less than this value.<br>The smaller the SPLIT_THRESHOLD value is, the larger a C45 or CART tree grows. On the contrary, CHAID grows a larger tree with larger SPLIT_THRESHOLD value. | |
| `<target_value_name>`_PRIOR_ | DOUBLE | Detected from input data | Specifies the prior probability of each class label. | Valid only for classification. |
| DISCRETIZATION_TYPE | INTEGER | 0 | Specifies the strategy for discretizing continuous attributes:<br>• 0: MDLPC<br>• 1: Equal Frequency | Valid only for C45 and CHAID. |
| `<column_name>`_BIN_ | INTEGER | 10 | Specifies the number of bins for discretisation. | Only valid when DISCRETIZATION_TYPE is 1. |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| MAX_BRANCH | INTEGER | 10 | Specifies the maximum number of branches. | Valid only for CHAID. |
| MERGE_THRESHOLD | DOUBLE | 0.05 | Specifies the merge condition for CHAID. If the metric value is greater than or equal to the specified value, the algorithm merges two branches. | Valid only for CHAID. |
| USE_SURROGATE | INTEGER | 1 | Indicates whether to use surrogate split when NULL values are encountered:<br>• 0: Not to use surrogate split.<br>• 1: Use surrogate split. | Valid only for CART. |
| MODEL_FORMAT | INTEGER | 1 | Specifies the tree model format for storage:<br>• 1: JSON<br>• 2: PMML | |
| CATEGORICAL_VARIABLE | VARCHAR or NVARCHAR | Detected from input data | Indicates which variables are treated as categorical:<br>• STRING: categorical<br>• INTEGER and DOUBLE: continuous | Valid only for INTEGER variables; omitted otherwise. |
| LABEL | VARCHAR or NVARCHAR | The last column name of the DATA table | Specifies the variable used as dependent. | Cannot be the first column name if KEY is 1. |

Parameter search and selection

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| RESAMPLING_METHOD | VARCHAR, or NVARCHAR | No default | The resampling method for model evaluation or parameter search:<br>• cv: cross validation<br>• stratified_cv: stratified cross validation<br>• bootstrap: bootstrap method<br>• stratified_bootstrap: stratified bootstrap method | |
| FOLD_NUM | INTEGER | No default | The fold number for cross validation. | Valid only for cross validation; mandatory for cross validation. |
| REPEAT_TIMES | INTEGER | 1 | The number of repeated times for model evaluation or parameter search. | |
| EVALUATION_METRIC | VARCHAR, or NVARCHAR | ERROR_RATE | The evaluation metric, candidates are:<br>• RMSE<br>• MAE<br>• ERROR_RATE<br>• NLL<br>• AUC | Multi-classification does not use RMSE or MAE |
| TIME_OUT | INTEGER | 0 | The time allocated (seconds) for program running. 0 indicates unlimited. | |
| PARAM_SEARCH_STRATEGY | VARCHAR, or NVARCHAR | No default | The search strategy for parameters:<br>• random: random search<br>• grid: grid search | If not specified, only model evaluation is carried out. |
| RANDOM_SEARCH_TIMES | INTEGER | No default | Specifies the number of search times for random search. | Mandatory for random search. |
| PROGRESS_INDICATOR_ID | NVARCHAR | No default | Sets an ID of progress indicator for model evaluation or parameter selection. | No progress indicator is active if no value is provided. |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| DISCRETIZATION_TYPE_VALUES | VARCHAR, or NVARCHAR | No default | The discrete set for the DISCRETIZATION_TYPE candidate. | |
| DISCRETIZATION_TYPE_RANGE | VARCHAR, or NVARCHAR | No default | The range for the DISCRETIZATION_TYPE candidate. | |
| MIN_RECORDS_LEAF_VALUES | VARCHAR, or NVARCHAR | No default | The discrete set for the MIN_RECORDS_OF_LEAF candidate. | |
| MIN_RECORDS_LEAF_RANGE | VARCHAR, or NVARCHAR | No default | The range for the MIN_RECORDS_OF_LEAF candidate. | |
| MIN_RECORDS_PARENT_VALUES | VARCHAR, or NVARCHAR | No default | The discrete set for the MIN_RECORDS_ PARENT candidate. | |
| MIN_RECORDS_PARENT_RANGE | VARCHAR, or NVARCHAR | No default | The range for the MIN_RECORDS_ PARENT candidate. | |
| MAX_DEPTH_VALUES | VARCHAR, or NVARCHAR | No default | The discrete set for the MAX_DEPTH candidate. | |
| MAX_DEPTH_RANGE | VARCHAR, or NVARCHAR | No default | The range for the MAX_DEPTH candidate. | |
| SPLIT_THRESHOLD_VALUES | VARCHAR, or NVARCHAR | No default | The discrete set for the SPLIT_THRESHOLD candidate. | |
| SPLIT_THRESHOLD_RANGE | VARCHAR, or NVARCHAR | No default | The range for the SPLIT_THRESHOLD candidate. | |
| MAX_BRANCH_VALUES | VARCHAR, or NVARCHAR | No default | The discrete set for the MAX_BRANCH candidate. | |
| MAX_BRANCH_RANGE | VARCHAR, or NVARCHAR | No default | The range for the MAX_BRANCH candidate. | |
| MERGE_THRESHOLD_VALUES | VARCHAR, or NVARCHAR | No default | The discrete set for the MERGE_THRESHOLD candidate. | |
| MERGE_THRESHOLD_RANGE | VARCHAR, or NVARCHAR | No default | The range for the MERGE_THRESHOLD candidate. | |

## HYBRID GRADIENT BOOSTING TREE

Optional

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| THREAD_RATIO | DOUBLE | -1 | The ratio of available threads.<br>• 0: single thread<br>• 0–1: uses the specified percentage of available threads. PAL uses all available threads if the number is 1.0.<br>• -1: uses all of available threads. | |
| SEED | INTEGER | 0 | The seed for random number generating.<br>• 0: current time<br>• Others: the seed | |
| SPLIT_METHOD | VARCHAR or NVARCHAR | histogram | The method to find split point for numeric feature.<br>• exact: the exact method, trying all possible points<br>• sketch: the sketch method, accounting for the distribution of the sum of hessian<br>• sampling: samples the split point randomly<br>• histogram:builds histogram upon data and uses it as the split point | The exact method comparably has the highest test accuracy, but costs more time. The other three methods have relative higher computational efficiency but might lead to lower test accuracy, and are considered to be adopted as the training data set is huge.<br>Valid only for numeric features. |
| SKETCH_ESP | DOUBLE | 0.1 | The epsilon of the sketch method. It indicates that the sum of hessian between two split points is not larger than this value. That is, the number of bins is approximately 1/eps. | The less is this value, the more split points are tried. |
| N_ESTIMATORS | INTEGER | 10 | Total iteration number, which is equivalent to the number of trees in the final model. | |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| MAX_DEPTH | INTEGER | 6 | The maximum depth of each tree. | |
| MIN_SAMPLES_LEAF | INTEGER | 1 | The minimum number of data in a leaf. | |
| LEARNING_RATE | DOUBLE | 0.3 | Learning rate of each iteration.<br>Range: (0, 1] | |
| SPLIT_THRESHOLD | DOUBLE | 0 | The minimum loss change value to make a split in tree growth (gamma in the equation). | |
| MIN_SAMPLES_WEIGHT_LEAF | DOUBLE | 1 | The minimum summation of sample weights (hessian) in the leaf node. | |
| MAX_W_IN_SPLIT | DOUBLE | 0 | The maximum weight constraint assigned to each tree node. 0 for no constraint. | |
| SUBSAMPLE | DOUBLE | 1 | The sample rate of row (data points).<br>Range: (0, 1] | |
| COL_SUBSAMPLE_SPLIT | DOUBLE | 1 | The sample rate of feature set in each split.<br>Range: (0, 1] | |
| COL_SUBSAMPLE_TREE | DOUBLE | 1 | The sample rate of feature set in each tree growth.<br>Range: (0, 1] | |
| LAMB | DOUBLE | 1 | L2 regularization.<br>Range: [0, 1] | |
| ALPHA | DOUBLE | 0 | L1 regularization.<br>Range: [0, 1] | |
| BASE_SCORE | DOUBLE | 0.5 | Initial prediction score of all instances.<br>Global bias for sufficient number of iterations.<br>Range: (0, 1), only for binary classification. | |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| CATEGORICAL_VARIABLE | VARCHAR or NVARCHAR | Detected from input data | Indicates which variables are treated as categorical. The default behavior is:<br>• STRING: categorical<br>• INTEGER and DOUBLE: continuous | Valid only for INTEGER variables; omitted otherwise. |
| LABEL | VARCHAR or NVARCHAR | The last column name of the DATA table | Specifies the variable used as dependent. | Cannot be the first column name if KEY is 1. |
| COMPRESSION | INTEGER | 0 | Indicates whether result model should be compressed:<br>• 0: no;<br>• 1: yes. | |
| MAX_BITS | INTEGER | 12 | Maximum number of bits to quantize continuous features. Equivalent to use $2^{MAX_BITS}$ bins.<br>Must be less than 31. | Valid only when the value of COMPRESSION is 1. |
| FEATURE_GROUPING | INTEGER | 1 | Applies feature grouping by grouping sparse features that only contain one significant value in each row.<br>• 0: no<br>• 1: yes | |
| TOLERANT_RATE | DOUBLE | 0.0001 | While applying feature grouping, features are still merged when there are rows containing more than one significant value. This parameter specifies the rate of such rows that are allowed. | Only available when FEATURE_GROUPING is set to 1. |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| DEFAULT_MISSING_DIRECTION | INTEGER | 1 | Defines the default direction where the missing value goes to while splitting.<br>• 0: left<br>• 1: right | |
| REPLACE_MISSING | INTEGER | 1 | Replaces the missing value by another value in the feature.<br>If it is a continuous feature, the value is the mean value;<br>If it is a categorical feature, the value is the most frequent one.<br>• 0: no<br>• 1: yes | |
| START_FROM_AVERAGE | INTEGER | 0 | Indicates whether to adopt the prior distribution as the initial point. To be specific, use average value if it is a regression problem.<br>• 0: no;<br>• 1: yes. | BASE_SCORE is ignored if this parameter is set to 1. |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| OBJ_FUNC | INTEGER | • 0 for regression problem<br>• 5 for binary classification<br>• 7 for multiple classification | For regression:<br>• 0: Squared error<br>• 1: Squared log error<br>• 2: Pseudo Huber error<br>• 3: Gamma<br>• 4: Tweedie<br>For binary classification:<br>• 5: Logistic<br>• 6: Hinge<br>For multiple classification:<br>• 7: Softmax | |
| TWEEDIE_POWER | DOUBLE | 1.5 | For Tweedie object function; must be in range [1.0, 2.0]. | Only valid when OBJ_FUNC equals 4. |
| VALIDATION_SET_RATE | DOUBLE | 0.0 | Specifies the rate of validation set that is sampled from the data set. If 0.0 is set, then no early stop will be applied. | |
| STRATIFIED_VALIDATION_SET | INTEGER | 0 | Indicates weather or not to apply the stratified method while sampling validation set. | Valid only when VALIDATION_SET_RATE is greater than 0. |
| TOLERANT_ITER_NUM | INTEGER | 10 | Indicates how many consecutive deteriorated iterations are observed before applying early stop. | Valid only when VALIDATION_SET_RATE is greater than 0. |

Parameter search and selection

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| RESAMPLING_METHOD | NVARCHAR | No default value | Specifies the resampling method for model evaluation or parameter selection.<br>• cv<br>• cv_sha<br>• cv_hyperband<br>• stratified_cv<br>• stratified_cv_sha<br>• stratified_cv_hyperband<br>• bootstrap<br>• bootstrap_sha<br>• bootstrap_hyperband<br>• stratified_bootstrap<br>• stratified_bootstrap_sha<br>• stratified_bootstrap_hyperband<br>stratified_* can only apply to classification algorithms. If no value is specified for this parameter, neither model evaluation nor parameter selection is activated. | Options like stratified_* can only be applied to classification algorithms. Options like *_sha and *_hyperband can only be used in parameter selection, can not be used in model evaluation. |
| EVALUATION_METRIC | VARCHAR or NVARCHAR | ERROR_RATE (for classification), | The metrics used for cross-validation. | Classification: RMSE, MAE, NLL, ERROR_RATE, AUC<br>It is mandatory if RESAMPLING_METHOD is set. |
| REF_METRIC | VARCHAR or NVARCHAR | ERROR_RATE (for binary classification), | The metric(s) for reference. Could be of multi-lines, that is, more than one REF_METRIC. | The metric that is identical to EVALUATION_METRIC is omitted |
| FOLD_NUM | INTEGER | No default value | Specifies the fold number for the cross validation method. | Mandatory and valid only when RESAMPLING_METHOD is set to cv or stratified_cv. |
| REPEAT_TIMES | INTEGER | 1 | Specifies the number of repeat times for resampling. | |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| PARAM_SEARCH_STRATEGY | NVARCHAR | No default value | Specifies the parameter search method:<br>• grid<br>• random | |
| RANDOM_SEARCH_TIMES | INTEGER | No default value | Specifies the number of times to randomly select candidate parameters for selection. | Mandatory and valid when PARAM_SEARCH_STRATEGY is set to random. |
| REDUCTION_RATE | DOUBLE | 3.0 | Specifies reduction rate in SHA or Hyperband method. | Only valid when RESAMPLING_METHOD is specified as *_sha or *_hyperband. |
| RESOURCE | NVARCHAR | No default value | Specifies resource definition for SHA or Hyperband method.<br>• Not specified, size of data set is used as resource by default.<br>• 'ITER_NUM' | Only valid when RESAMPLING_METHOD is specified as *_sha or *_hyperband. |
| MIN_RESOURCE_RATE | DOUBLE | 0.0 or 1/MAX_RESOURCE | Specifies the minimum resource rate that should be used in SHA or Hyperband iteration. | Only valid when RESAMPLING_METHOD is specified as *_sha or *_hyperband. Default value 0.0 for default RESOURCE and default value 1/MAX_RESOURCE for RESOURCE = ITER_NUM |
| MAX_RESOURCE | INTEGER | No default value | Specifies the maximum number of ITER_NUM that should be used in SHA or Hyperband method. | Only valid when RESAMPLING_METHOD is specified as *_sha or *_hyperband. It is valid and mandatory if RESOURCE is set. |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| AGGRESSIVE_ELIMINATION | INTEGER | 0 | Specifies whether to apply aggressive elimination while using SHA method. Aggressive elimination happens when the data size and parameters size to be searched does not match and there are still bunch of parameters to be searched while data size reaches its upper limits. If aggressive elimination is applied, lower bound of limit of data size will be used multiple times first to reduce number of parameters.<br>• 0: does not apply aggressive elimination<br>• 1: applies aggressive elimination | Only valid when RESAMPLING_METHOD is specified as *_sha. |
| TIMEOUT | INTEGER | 0 | Specifies maximum running time for model evaluation or parameter selection, in seconds. No timeout when 0 is specified. | |
| PROGRESS_INDICATOR_ID | NVARCHAR | No default value | Sets an ID of progress indicator for model evaluation or parameter selection. No progress indicator is active if no value is provided. | |
| N_ESTIMATORS_RANGE | VARCHAR or NVARCHAR | Null | | |
| N_ESTIMATORS_VALUES | VARCHAR or NVARCHAR | Null | | |
| MAX_DEPTH_RANGE | VARCHAR or NVARCHAR | Null | | |
| MAX_DEPTH_VALUES | VARCHAR or NVARCHAR | Null | | |
| MIN_SAMPLES_LEAF_RANGE | VARCHAR or NVARCHAR | Null | | |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| MIN_SAMPLES_LEAF_VALUES | VARCHAR or NVARCHAR | Null | | |
| LEARNING_RATE_RANGE | VARCHAR or NVARCHAR | Null | | |
| LEARNING_RATE_VALUES | VARCHAR or NVARCHAR | Null | | |
| SPLIT_THRESHOLD_RANGE | VARCHAR or NVARCHAR | Null | | |
| SPLIT_THRESHOLD_VALUES | VARCHAR or NVARCHAR | Null | | |
| MIN_SAMPLES_WEIGHT_LEAF_RANGE | VARCHAR or NVARCHAR | Null | | |
| MIN_SAMPLES_WEIGHT_LEAF_VALUES | VARCHAR or NVARCHAR | Null | | |
| MAX_W_IN_SPLIT_RANGE | VARCHAR or NVARCHAR | Null | | |
| MAX_W_IN_SPLIT_VALUES | VARCHAR or NVARCHAR | Null | | |
| SUBSAMPLE_RANGE | VARCHAR or NVARCHAR | Null | | |
| SUBSAMPLE_VALUES | VARCHAR or NVARCHAR | Null | | |
| COL_SUBSAMPLE_SPLIT_RANGE | VARCHAR or NVARCHAR | Null | | |
| COL_SUBSAMPLE_SPLIT_VALUES | VARCHAR or NVARCHAR | Null | | |
| COL_SUBSAMPLE_TREE_RANGE | VARCHAR or NVARCHAR | Null | | |
| COL_SUBSAMPLE_TREE_VALUES | VARCHAR or NVARCHAR | Null | | |
| LAMB_RANGE | VARCHAR or NVARCHAR | Null | | |
| LAMB_VALUES | VARCHAR or NVARCHAR | Null | | |
| ALPHA_RANGE | VARCHAR or NVARCHAR | Null | | |
| ALPHA_VALUES | VARCHAR or NVARCHAR | Null | | |
| BASE_SCORE_RANGE | VARCHAR or NVARCHAR | Null | | |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| BASE_SCORE_VALUES | VARCHAR or NVARCHAR | Null | | |

### LOGISTIC REGRESSION

Mandatory

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION |
|---|---|---|---|
| CLASS_MAP0 | VARCHAR | No default value | Specifies the dependent variable value which will be mapped to 0.<br>Only valid when the column type of dependent variables is VARCHAR or NVARCHAR. |
| CLASS_MAP1 | VARCHAR | No default value | Specifies the dependent variable value which will be mapped to 1.<br>Only valid when the column type of dependent variables is VARCHAR or NVARCHAR. |

Optional

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| METHOD | INTEGER | -1 | • -1: Auto selection, based on given data and parameters<br>• 0: Newton iteration method<br>• 2: Cyclical coordinate descent method to fit elastic net regularized logistic regression.<br>• 3: LBFGS method (recommended when having many independent variables).<br>• 4: Stochastic gradient descent method (recommended when dealing with very large dataset).<br>• 6: Proximal gradient descent method to fit elastic net regularized logistic regression. | |
| THREAD_RATIO | DOUBLE | -1 | The ratio of available threads.<br>• 0: single thread<br>• 0–1: uses the specified percentage of available threads. PAL uses all available threads if the number is 1.0.<br>• -1: uses all of available threads. | |
| MAX_PASS_NUMBER | INTEGER | 1 | The maximum number of passes over the data. | Only valid when METHOD is 4. |
| SGD_BATCH_NUMBER | INTEGER | 1 | The batch number of Stochastic gradient descent. | Only valid when METHOD is 4. |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| PRECOMPUTE | INTEGER | 1 | Whether to compute Gram matrix:<br>• 0: No<br>• 1: Yes | Only valid when METHOD is 2. |
| LAMB | DOUBLE | 0 | Penalized weight. The value should be equal to or greater than 0. | Only valid when METHOD is 0, 2, 3, or 6.<br>**Note**<br>If you want to solve Ridge penalty problem for METHOD 0, set ENET_ALPHA to 0.0. |
| ALPHA | DOUBLE | 1 | The elastic net mixing parameter. The value range is between 0 and 1 inclusively.<br>• 0: Ridge penalty<br>• 1: LASSO penalty | Only valid when METHOD is 0, 2, 3, or 6. |
| HANDLE_MISSING | INTEGER | 1 | Whether to handle missing values:<br>• 0: No<br>• 1: Yes | |
| MAX_ITER | INTEGER | 100, 1000, or 100000 | When METHOD is 0 or 3, the convex optimizer may return suboptimal results after the maximum number of iterations. When METHOD is 2, if convergence is not reached after the maximum number of passes over training data, an error will be generated. | When METHOD is 2, MAX_ITERATION defaults to 100000; when METHOD is 6, it defaults to 1000; otherwise it defaults to 100. |
| STANDARDIZE | INTEGER | 1 | Controls whether to standardize the data to have zero mean and unit variance.<br>• 0: No<br>• 1: Yes | |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| TOL | DOUBLE | 1.0e-6 or 1.0e-7 | Convergence threshold for exiting iterations. | When METHOD is 2, TOL defaults to 1.0E-7; Otherwise it defaults to 1.0E-6. |
| EPSILON | DOUBLE | 1.0e-5 or 1.0e-6 | The parameter determines the accuracy with which the solution is to be found.<br>When METHOD is 3, the condition is: ||g|| < EPSILON * max {1, ||x||}, where g is gradient of objective function, x is solve of current iteration, and ||·|| denotes the L2 norm.<br>When METHOD is 0, the condition is: ||x-x'|| < EPSILON * sqrt(n), where x is the solve of current iteration, x' is the previous iteration, and n is the number of features. | When METHOD is 0, EPSILON defaults to 1.0e-6. When METHOD is 3, EPSILON defaults to 1.0e-5. |
| LBFGS_M | INTEGER | 6 | Number of past updates needed to be kept. | Only available when METHOD is 3. |
| STAT_INF | INTEGER | 0 | • 0: Does not proceed statistical inference<br>• 1: Proceeds statistical inference | |
| SELECTED_FEATURES | VARCHAR | No default value | A string to specify the features that will be processed. The pattern is "X1,...,Xn", where Xi is the corresponding column name in the data table. If this parameter is not specified, all the features will be processed. | Only used when you need to indicate the needed features. |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| CATEGORICAL_VARIABLE | VARCHAR or NVARCHAR | No default value | Indicates which variables are treated as categorical. The default behavior is:<br>• STRING: categorical<br>• INTEGER and DOUBLE: continuous | Valid only for INTEGER variables; omitted otherwise. |
| LABEL | VARCHAR or NVARCHAR | The last column name of the DATA table | Specifies the variable used as dependent. | Cannot be the first column name if KEY is 1. |

Parameter search and selection

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| RESAMPLING_METHOD | NVARCHAR | No default value | Specifies the resampling method for model evaluation or parameter selection.<br>• cv<br>• cv_sha<br>• cv_hyperband<br>• bootstrap<br>• bootstrap_sha<br>• bootstrap_hyperband<br>• stratified_cv<br>• stratified_cv_sha<br>• stratified_cv_hyperband<br>• stratified_bootstrap<br>• stratified_bootstrap_sha<br>• stratified_bootstrap_hyperband<br>If no value is specified for this parameter, neither model evaluation nor parameter selection is activated. | |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| RESOURCE | NVARCHAR | No default value | Resource type used in successive-halving and hyperband algorithm for parameter selection.<br>• MAX_ITERATION<br>• MAX_PASS_NUMBER | Mandatory and valid only when RESAMPLING_METHOD parameter has values with sha or hyperband suffix. The value MAX_PASS_NUMBER is only valid when actual used method is Stochastic Gradient Descent; MAX_ITERATION is used for other methods. When user set METHOD as -1 (Auto Selection), the value to be used depends on the actual used method; For current implementation users should always choose MAX_ITERATION in this case. |
| MAX_RESOURCE | MAX_RESOURCE | No default value | Maximum allowed resource budget for single hyper-parameter candidate. Valid value must be larger than 0. | Mandatory and valid only when RESOURCE parameter is set. |
| MIN_RESOURCE_RATE | DOUBLE | 0 | The rate between minimum allowed resource and maximum allowed resource. Valid value is between [0, 1). | Mandatory and valid only when RESOURCE parameter is set. |
| REDUCTION_RATE | DOUBLE | 3.0 | This parameter specifies the reduction rate of available size of hyper-parameter candidates. For each round, the available parameter candidate size will be divided by value of this parameter. Thus valid value for this parameter must be larger than 1.0. | Valid only when RESOURCE parameter is set. |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| AGGRESSIVE_ELIMINATION | INTEGER | 0 | Boolean value indicates whether to perform aggressive elimination behavior for successive-halving algorithm. When set to 1, it will eliminate more parameter candidates than expected, which is defined by REDUCTION_RATE parameter; This can accelerate the run-time performance but may result in sub-optimal hyper-parameter candidate.<br>• 0: Does not perform aggressive elimination;<br>• 1: Performs aggressive elimiination | Valid only when RESAMPLING_METHOD parameter has value with suffix sha. |
| PS_VERBOSE | INTEGER | 1 | Boolean value indicates whether to output optimal hyper-parameter and all involved hyper-parameter candidates related evaluation statistics in Statistics output table.<br>• 0: Does not output evaluation statistics<br>• 1: Outputs evaluation statistics | Valid only when RESAMPLING_METHOD is set. |
| EVALUATION_METRIC | NVARCHAR | No default value | Specifies the evaluation metric for model evaluation or parameter selection.<br>• ACCURACY<br>• F1_SCORE<br>• AUC<br>• NLL | |
| FOLD_NUM | INTEGER | No default value | Specifies the fold number for the cross validation method. | Mandatory and valid only when RESAMPLING_METHOD is set to cv or stratified_cv. |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| REPEAT_TIMES | INTEGER | 1 | Specifies the number of repeat times for re-sampling. | |
| PARAM_SEARCH_STRATEGY | NVARCHAR | No default value | Specifies the method to activate parameter selection.<br>• grid<br>• random | This parameter is only valid when RESAMPLING_METHOD does not have value with suffix hyperband; Otherwise, this parameter is always considered to have value random. |
| RANDOM_SEARCH_TIMES | INTEGER | No default value | Specifies the number of times to randomly select candidate parameters for selection. | Mandatory and valid when PARAM_SEARCH_STRATEGY is set to random. |
| SEED | INTEGER | 0 | Specifies the seed for random generation. Use system time when 0 is specified. | |
| TIMEOUT | INTEGER | 0 | Specifies maximum running time for model evaluation or parameter selection, in seconds. No timeout when 0 is specified. | |
| PROGRESS_INDICATOR_ID | NVARCHAR | No default value | Sets an ID of progress indicator for model evaluation or parameter selection. No progress indicator is active if no value is provided. | |
| LAMB_VALUES | NVARCHAR | No default value | Specifies values of LAMB to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| LAMB_RANGE | NVARCHAR | No default value | Specifies the range of LAMB to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| ALPHA_VALUES | NVARCHAR | No default value | Specifies values of ALPHA to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| ALPHA_RANGE | NVARCHAR | No default value | Specifies the range of ALPHA to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |

### MULTI-CLASS LOGISTIC REGRESSION

Optional

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| MAX_ITER | INTEGER | 100 | Maximum number of iterations of the optimization. | |
| MODEL_FORMAT | INTEGER | 1 | Specifies the tree model format for storage:<br>• 1: JSON<br>• 2: PMML | |
| SELECTED_FEATURES | VARCHAR | No default value | A string to specify the features that will be processed. The pattern is "X1,...,Xn", where Xi is the corresponding column name in the data table. If this parameter is not specified, all the features will be processed. | Only used when you need to indicate the needed features. |
| METHOD | INTEGER | 0 | • 0: lbfgs<br>• 1: cycle coordinate descent | |
| LAMB | DOUBLE | 0 | Penalized weight. The value should be equal to or greater than 0. | |
| ALPHA | DOUBLE | 1 | The elastic net mixing parameter. The value range is between 0 and 1 inclusively.<br>• 0: Ridge penalty<br>• 1: LASSO penalty | METHOD 0 only solve Ridge penalty problem. |
| TOL | DOUBLE | 1.00E-05 | Convergence threshold for exiting iterations. | |
| STANDARDIZE | INTEGER | 1 | Controls whether to standardize the data to have zero mean and unit variance.<br>• 0: No<br>• 1: Yes | |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| STAT_INF | INTEGER | 0 | Controls whether to proceed statistical inference.<br>• 0: Does not proceed statistical inference.<br>• 1: Proceeds statistical inference. | |
| CATEGORICAL_VARIABLE | VARCHAR or NVARCHAR | No default value | Indicates which variables are treated as categorical. The default behavior is:<br>• STRING: categorical<br>• INTEGER and DOUBLE: continuous | Valid only for INTEGER variables; omitted otherwise. |
| LABEL | VARCHAR or NVARCHAR | The last column name of the DATA table | Specifies the variable used as dependent. | Cannot be the first column name if KEY is 1. |

Parameter search and selection

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| RESAMPLING_METHOD | NVARCHAR | No default value | Specifies the resampling method for model evaluation or parameter selection.<br>• cv<br>• cv_sha<br>• cv_hyperband<br>• bootstrap<br>• bootstrap_sha<br>• bootstrap_hyperband<br>• stratified_cv<br>• stratified_cv_sha<br>• stratified_cv_hyperband<br>• stratified_bootstrap<br>• tratified_bootstrap_sha<br>• stratified_bootstrap_hyperband<br>If no value is specified for this parameter, neither model evaluation nor parameter selection is activated. | |
| RESOURCE | NVARCHAR | No default value | Resource type used in successive-halving and hyperband algorithm for parameter selection.<br>• MAX_ITERATION | Mandatory and valid only when RESAMPLING_METHOD parameter has values with sha or hyperband suffix. |
| MAX_RESOURCE | INTEGER | No default value | Maximum allowed resource budget for single hyper-parameter candidate. Valid value must be larger than 0. | Mandatory and valid only when RESOURCE parameter is set. |
| MIN_RESOURCE_RATE | DOUBLE | 0 | The rate between minimum allowed resource and maximum allowed resource. Valid value is between [0, 1). | Mandatory and valid only when RESOURCE parameter is set. |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| REDUCTION_RATE | DOUBLE | 3.0 | This parameter specifies the reduction rate of available size of hyper-parameter candidates. For each round, the available parameter candidate size will be divided by value of this parameter. Thus valid value for this parameter must be larger than 1.0. | Valid only when RESOURCE parameter is set. |
| AGGRESSIVE_ELIMINATION | INTEGER | 0 | Boolean value indicates whether to perform aggressive elimination behavior for successive-halving algorithm. When set to 1, it will eliminate more parameter candidates than expected, which is defined by REDUCTION_RATE parameter; This can accelerate the run-time performance but may result in sub-optimal hyper-parameter candidate.<br>• 0: Does not perform aggressive elimination;<br>• 1: Performs aggressive elimiination | Valid only when RESAMPLING_METHOD parameter has value with suffix sha. |
| PS_VERBOSE | INTEGER | 1 | Boolean value indicates whether to output optimal hyper-parameter and all involved hyper-parameter candidates related evaluation statistics in Statistics output table.<br>• 0: Does not output evaluation statistics<br>• 1: Outputs evaluation statistics | Valid only when RESAMPLING_METHOD is set. |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| EVALUATION_METRIC | NVARCHAR | No default value | Specifies the evaluation metric for model evaluation or parameter selection.<br>• ACCURACY<br>• F1_SCORE<br>• AUC<br>• NLL | Only ordinal regression use ERROR_RATE. |
| FOLD_NUM | INTEGER | No default value | Specifies the fold number for the cross validation method. | Mandatory and valid only when RESAMPLING_METHOD is set to cv or stratified_cv. |
| REPEAT_TIMES | INTEGER | 1 | Specifies the number of repeat times for resampling. | |
| PARAM_SEARCH_STRATEGY | NVARCHAR | No default value | Specifies the method to activate parameter selection.<br>• grid<br>• random | This parameter is only valid when RESAMPLING_METHOD does not have value with suffix hyperband; Otherwise, this parameter is always considered to have value random. |
| RANDOM_SEARCH_TIMES | INTEGER | No default value | Specifies the number of times to randomly select candidate parameters for selection. | Mandatory and valid when PARAM_SEARCH_STRATEGY is set to random. |
| SEED | INTEGER | 0 | Specifies the seed for random generation. Use system time when 0 is specified. | |
| TIMEOUT | INTEGER | 0 | Specifies maximum running time for model evaluation or parameter selection, in seconds. No timeout when 0 is specified. | |
| PROGRESS_INDICATOR_ID | NVARCHAR | No default value | Sets an ID of progress indicator for model evaluation or parameter selection. No progress indicator is active if no value is provided. | |
| LAMB_VALUES | NVARCHAR | No default value | Specifies values of LAMB to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| LAMB_RANGE | NVARCHAR | No default value | Specifies the range of LAMB to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| ALPHA_VALUES | NVARCHAR | No default value | Specifies values of ALPHA to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| ALPHA_RANGE | NVARCHAR | No default value | Specifies the range of ALPHA to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |

### MULTILAYER PERCEPTRON

Mandatory

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION |
|---|---|---|---|
| ACTIVATION | INTEGER | No default value | Active function code for the hidden layer.<br>• TANH = 1<br>• LINEAR = 2<br>• SIGMOID_ASYMMETRIC = 3<br>• SIGMOID_SYMMETRIC = 4<br>• GAUSSIAN_ASYMMETRIC = 5<br>• GAUSSIAN_SYMMETRIC = 6<br>• ELLIOT_ASYMMETRIC = 7<br>• ELLIOT_SYMMETRIC = 8<br>• SIN_ASYMMETRIC = 9<br>• SIN_SYMMETRIC = 10<br>• COS_ASYMMETRIC = 11<br>• COS_SYMMETRIC = 12<br>• ReLU = 13 |
| OUTPUT_ACTIVATION | INTEGER | No default value | Active function code for the output layer. The code is the same as ACTIVATION. |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION |
|---|---|---|---|
| HIDDEN_LAYER_SIZE | NVARCHAR | No default value | Specifies the size of each hidden layer in the format of `2, 3, 4`. The value 0 will be ignored, for example, `2, 0, 3` is equal to `2, 3`. |

Optional

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| MAX_ITER | INTEGER | 100 | Maximum number of iterations. | |
| TRAINING_STYLE | INTEGER | 1 | Specifies the training style:<br>• 0: Batch<br>• 1:Stochastic | |
| LEARNING_RATE | DOUBLE | None | Specifies the learning rate. | Mandatory and valid only when TRAINING_STYLE = 1. |
| MOMENTUM | DOUBLE | None | Specifies the momentum factor. | Mandatory and valid only when TRAINING_STYLE = 1. |
| BATCH_SIZE | INTEGER | 1 | Specifies the size of mini batch. | Valid only when TRAINING_STYLE = 1. |
| NORMALIZATION | INTEGER | 0 | Specifies the normalization type:<br>• 0: None<br>• 1: Z-transform<br>• 2: Scalar | |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| WEIGHT_INIT | INTEGER | 0 | Specifies the weight initial value:<br>• 0: All zeros<br>• 1: Normal distribution<br>• 2: Uniform distribution in range (0, 1)<br>• 3: Variance scale normal<br>• 4: Variance scale uniform | |
| THREAD_RATIO | DOUBLE | -1 | The ratio of available threads.<br>• 0: single thread<br>• 0–1: uses the specified percentage of available threads. PAL uses all available threads if the number is 1.0.<br>• -1: uses all of available threads. | |
| CATEGORICAL_VARIABLE | VARCHAR or NVARCHAR | No default value | Indicates which variables are treated as categorical. The default behavior is:<br>• STRING: categorical<br>• INTEGER and DOUBLE: continuous | Valid only for INTEGER variables; omitted otherwise. |
| LABEL | VARCHAR or NVARCHAR | The last column name of the DATA table | Specifies the variable used as dependent. | Cannot be the first column name if KEY is 1. |

Parameter search and selection

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| RESAMPLING_METHOD | NVARCHAR | No default value | Specifies the resampling method for model evaluation or parameter selection.<br>• cv<br>• cv_sha<br>• cv_hyperband<br>• bootstrap<br>• bootstrap_sha<br>• bootstrap_hyperband<br>• stratified_bootstrap<br>• stratified_bootstrap_sha<br>• stratified_bootstrap_hyperband<br>• stratified_cv<br>• stratified_cv_sha<br>• stratified_cv_hyperband<br>If no value is specified for this parameter, neither model evaluation nor parameter selection is activated. | |
| REDUCTION_RATE | DOUBLE | 3.0 | Specifies reduction rate in SHA or hyperband method. | Only valid when RESAMPLING_METHOD is specified as *_sha or *_hyberband. |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| AGGRESSIVE_ELIMINATION | INTEGER | | Specifies whether to apply aggressive elimination while using SHA method. It happens when the size of data and that of parameters to be searched does not match. And there are still bunch of parameters to be searched while data size reaches its upper limits. If aggressive elimination is applied, lower bound of limit of data size will be used multiple times first to reduce number of parameters.<br>• 0: not apply<br>• 1: apply | Only valid when RESAMPLING_METHOD is specified as *_sha. |
| EVALUATION_METRIC | NVARCHAR | No default value | Specifies the evaluation metric for model evaluation or parameter selection.<br>• ACCURACY<br>• F1_SCORE<br>• AUC_1VsRest<br>• AUC_pairwise | |
| FOLD_NUM | INTEGER | No default value | Specifies the fold number for the cross validation method. | Mandatory and valid only when RESAMPLING_METHOD is set to cv or stratified_cv. |
| REPEAT_TIMES | INTEGER | 1 | Specifies the number of repeat times for resampling. | |
| PARAM_SEARCH_STRATEGY | NVARCHAR | No default value | Specifies the method to activate parameter selection.<br>• grid<br>• random | When *_sha is specified in RESAMPLING_METHOD, PARAM_SEARCH_STRATEGY is mandatory. When *_hyperband is specified in RESAMPLING_METHOD, PARAM_SEARCH_STRATEGY is set to random whatever other value is provided. |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| RANDOM_SEARCH_TIMES | INTEGER | No default value | Specifies the number of times to randomly select candidate parameters for selection. | Mandatory and valid when PARAM_SEARCH_STRATEGY is set to random. |
| REDUCTION_RATE | DOUBLE | 3.0 | Specifies reduction rate in SHA or hyperband method. | Only valid when RESAMPLING_METHOD is specified as *_sha or *_hyberband. |
| AGGRESSIVE_ELIMINATION | INTEGER | | Specifies whether to apply aggressive elimination while using SHA method. It happens when the size of data and that of parameters to be searched does not match. And there are still bunch of parameters to be searched while data size reaches its upper limits. If aggressive elimination is applied, lower bound of limit of data size will be used multiple times first to reduce number of parameters.<br>• 0: not apply<br>• 1: apply | Only valid when RESAMPLING_METHOD is specified as *_sha. |
| SEED | INTEGER | 0 | Specifies the seed for random generation. Use system time when 0 is specified. | |
| TIMEOUT | INTEGER | 0 | Specifies maximum running time for model evaluation or parameter selection, in seconds. No timeout when 0 is specified. | |
| PROGRESS_INDICATOR_ID | NVARCHAR | No default value | Sets an ID of progress indicator for model evaluation or parameter selection. No progress indicator is active if no value is provided. | |
| ACTIVATION_OPTIONS | NVARCHAR | No default value | Specifies values of ACTIVATION to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| OUTPUT_ACTIVATION_OPTIONS | NVARCHAR | No default value | Specifies values of OUTPUT_ACTIVATION to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| HIDDEN_LAYER_SIZE_OPTIONS | NVARCHAR | No default value | Specifies values of HIDDEN_LAYER_SIZE to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| LEARNING_RATE_VALUES | NVARCHAR | No default value | Specifies values of LEARNING_RATE to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified and TRAINING_STYLE = 1. |
| LEARNING_RATE_RANGE | NVARCHAR | No default value | Specifies the range of LEARNING_RATE to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified and TRAINING_STYLE = 1. |
| MOMENTUM_VALUES | NVARCHAR | No default value | Specifies values of MOMENTUM to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified and TRAINING_STYLE = 1. |
| MOMENTUM_RANGE | NVARCHAR | No default value | Specifies the range of MOMENTUM to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified and TRAINING_STYLE = 1. |
| BATCH_SIZE_VALUES | NVARCHAR | No default value | Specifies values of BATCH_SIZE to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified and TRAINING_STYLE = 1. |
| BATCH_SIZE_RANGE | NVARCHAR | No default value | Specifies the range of BATCH_SIZE to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified and TRAINING_STYLE = 1. |

### NAÏVE BAYES

Optional

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| MODEL_FORMAT | INTEGER | 1 | Specifies the tree model format for storage:<br>• 1: JSON<br>• 2: PMML | |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| ALPHA | DOUBLE | 0 | Enables or disables Laplace smoothing.<br>• 0: Disables Laplace smoothing<br>• Positive value: Enables Laplace smoothing for discrete values<br>**Note**<br>The ALPHA value is only stored by JSON format models. If the PMML format is chosen, you may need to set the ALPHA value again in the predicting phase. | |
| DISCRETIZATION | INTEGER | 0 | • 0: Disables discretization.<br>• Other values: Uses supervised discretization to all the continuous attributes. | |
| THREAD_RATIO | DOUBLE | -1 | The ratio of available threads.<br>• 0: single thread<br>• 0–1: uses the specified percentage of available threads. PAL uses all available threads if the number is 1.0.<br>• -1: uses all of available threads. | |
| CATEGORICAL_VARIABLE | VARCHAR or NVARCHAR | No default value | Indicates which variables are treated as categorical. The default behavior is:<br>• STRING: categorical<br>• INTEGER and DOUBLE: continuous | Valid only for INTEGER variables; omitted otherwise. |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| LABEL | VARCHAR or NVARCHAR | The last column name of the DATA table | Specifies the variable used as dependent. | Cannot be the first column name if KEY is 1. |

Parameter search and selection

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| RESAMPLING_METHOD | NVARCHAR | No default value | Specifies resampling method for model evaluation or parameter selection.<br>• cv<br>• cv_sha<br>• cv_hyperband<br>• stratified_cv<br>• stratified_cv_sha<br>• stratified_cv_hyperband<br>• bootstrap<br>• bootstrap_sha<br>• bootstrap_hyperband<br>• stratified_bootstrap<br>• stratified_bootstrap_sha<br>• stratified_bootstrap_sha<br>If no value is specified for this parameter, neither model evaluation nor parameter selection is activated. | Options like stratified_* can only be applied to classification algorithms. Options like *_sha and *_hyperband can only be used in parameter selection, can not be used in model evaluation. |
| EVALUATION_METRIC | NVARCHAR | No default value | Specifies the evaluation metric for model evaluation or parameter selection.<br>Naive Bayes supports the following evaluation metrics:<br>• ACCURACY<br>• F1_SCORE<br>• AUC | Mandatory. |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| FOLD_NUM | INTEGER | No default value | Specifies the fold number for the cross validation method. | Mandatory and valid only when RESAMPLING_METHOD is set to cv or stratified_cv. |
| REPEAT_TIMES | INTEGER | 1 | Specifies the number of repeat times for resampling. | |
| PARAM_SEARCH_STRATEGY | NVARCHAR | No default value | Specifies the parameter search method:<br>• grid<br>• random | |
| RANDOM_SEARCH_TIMES | INTEGER | No default value | Specifies the number of times to randomly select candidate parameters for selection. | Mandatory and valid when PARAM_SEARCH_STRATEGY is set to random. |
| REDUCTION_RATE | DOUBLE | 3.0 | Specifies reduction rate in SHA or hyperband method. | Only valid when RESAMPLING_METHOD is specified as *_sha or *_hyberband. |
| AGGRESSIVE_ELIMINATION | INTEGER | 0 | Specifies whether to apply aggressive elimination while using SHA method. Aggressive elimination happens when the data size and parameters size to be searched does not match and there are still bunch of parameters to be searched while data size reaches its upper limits. If aggressive elimination is applied, lower bound of limit of data size will be used multiple times first to reduce number of parameters.<br>• 0: does not apply aggressive elimination<br>• 1: applies aggressive elimination | Only valid when RESAMPLING_METHOD is specified as *_sha. |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| SEED | INTEGER | 0 | Specifies the seed for random generation. Use system time when 0 is specified. | |
| TIMEOUT | INTEGER | 0 | Specifies maximum running time for model evaluation or parameter selection, in seconds. No timeout when 0 is specified. | |
| PROGRESS_INDICATOR_ID | NVARCHAR | No default value | Sets an ID of progress indicator for model evaluation or parameter selection. No progress indicator is active if no value is provided. | |
| ALPHA_RANGE | NVARCHAR | No default value | Specifies candidate ALPHA values for parameter selection. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| ALPHA_VALUES | NVARCHAR | No default value | Specifies the range for candidate ALPHA values for parameter selection. | Only valid when PARAM_SEARCH_STRATEGY is specified. |

### RANDOM DECISION TREES

Optional

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| THREAD_RATIO | DOUBLE | -1 | The ratio of available threads.<br>• 0: single thread<br>• 0–1: uses the specified percentage of available threads. PAL uses all available threads if the number is 1.0.<br>• -1: uses all of available threads. | |
| N_ESTIMATORS | INTEGER | 100 | Specifies the number of trees to grow. | |
| MAX_FEATURES | INTEGER | sqrt(p), where p is the number of valid independent variables | Specifies the number of randomly selected variables for splitting. | Should not larger than the valid number of features. |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| ALLOW_MISSING_LABEL | INTEGER | 1 | Specifies if missing target value is allowed.<br>• 0: Not allowed. An error occurs if missing dependent presents.<br>• 1: Allowed. The datum with missing dependent is removed. | |
| SEED | INTEGER | 0 | Specifies the seed for random number generator.<br>• 0: Uses the current time (in second) as seed<br>• Others: Uses the specified value as seed | |
| MIN_SAMPLES_LEAF | INTEGER | 1 for classification | Specifies the minimum number of records in a leaf. | |
| SPLIT_THRESHOLD | DOUBLE | 1.00E-05 | Specifies the stop condition: if the improvement value of the best split is less than this value, the tree stops growing. | |
| MAX_DEPTH | INTEGER | Unlimited | The maximum depth of a tree. | Cannot be larger than 56. |
| SAMPLE_FRACTION | DOUBLE | 1 | The fraction of data used for training.<br>Assume there are n pieces of data, SAMPLE_FRACTION is r, then n*r data is selected for training. | In the range of [0,1]. |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| STRATA | (INTEGER, DOUBLE), or (VARCHAR/NVARCHAR, DOUBLE) | No default value | Specifies the stratified sampling for classification.<br><br>This is a pair parameter, which means a parameter name corresponds to two values. The first value is the class label (INTEGER or VARCHAR/NVARCHAR), and the second one is the proportion of this class occupies in the sampling data. The class with a proportion less than 0 or larger than 1 will be ignored.<br><br>If some classes are absent, they share the remaining proportion equally.<br><br>If the summation of strata is less than 1, the remaining is shared by all classes equally. | Valid only for classification.<br><br>The summation of the proportion of strata should not be larger than 1.<br><br>If absent, bagging is used. |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| PRIORS | (INTEGER, DOUBLE), (VARCHAR/NVARCHAR, DOUBLE) | Determined by the proportion of every class in the training data | Specifies the prior probabilities for classification.<br>This is also a pair parameter. The first value is the class label (INTEGER or VARCHAR/NVARCHAR), and the second one is the prior probability of this class. The class with a prior probability less than 0 or larger than 1 will be ignored.<br>If some classes are absent, they share the remaining probability equally.<br>If the summation of priors is less than 1, the remaining is shared by all classes equally. | Valid only for classification.<br>The summation of the priors should not be larger than 1. |
| COMPRESSION | INTEGER | 0 | Specifies if the model is stored in compressed format<br>• 0: No<br>• 1: Yes | |
| MAX_BITS | INTEGER | 12 | The maximum number of bits to quantize continuous features. Equivalent to use 2^MAX_BITS bins. Must be less than 31. | Valid only when COMPRESSION equals to 1 |
| QUANTIZE_RATE | DOUBLE | 0.005 | Quantize a feature if the largest class frequency of which is less than QUANTIZE_RATE. | Valid only when COMPRESSION equals to 1 |
| FITTINGS_QUANTIZATION | INTEGER | 0 | Indicate whether to quantize fitting values.<br>• 0: No<br>• 1: Yes | Valid only when COMPRESSION equals to 1 |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| CATEGORICAL_VARIABLE | VARCHAR or NVARCHAR | Detected from input data | Indicates which variables are treated as categorical. The default behavior is:<br>• STRING: categorical<br>• INTEGER and DOUBLE: continuous | Valid only for INTEGER variables; omitted otherwise. |
| LABEL | VARCHAR or NVARCHAR | The last column name of the DATA table | Specifies the variable used as dependent. | Cannot be the first column name if KEY is 1. |
| MODEL_FORMAT | INTEGER | 1 | Specifies the tree model format for storage:<br>• 1: JSON<br>• 2: PMML | If both MODEL_FORMAT and COMPRESSION are specified, COMPRESSION takes precedence. |

### SUPPORT VECTOR MACHINE

Optional:

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| KERNEL_TYPE | INTEGER | 2 | Kernel type:<br>• 0: LINEAR KERNEL<br>• 1: POLY KERNEL<br>• 2: RBF KERNEL<br>• 3: SIGMOID KERNEL | |
| THREAD_RATIO | DOUBLE | -1 | The ratio of available threads.<br>• 0: single thread<br>• 0–1: uses the specified percentage of available threads. PAL uses all available threads if the number is 1.0.<br>• -1: uses all of available threads. | Only valid for parameter selection task. |
| POLY_DEGREE | INTEGER | 3 | Coefficient for the PLOY KERNEL type.<br>Value range: ≥ 1 | Only valid when KERNEL_TYPE is 1. |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| RBF_GAMMA | DOUBLE | 1.0/number of features in the dataset | Coefficient for the RBF KERNEL type.<br>Value range: > 0 | Only valid when KERNEL_TYPE is 2. |
| COEF_LIN | DOUBLE | 0 | Coefficient for the POLY/SIGMOID KERNEL type. | Only valid when KERNEL_TYPE is 1 or 3. |
| COEF_CONST | DOUBLE | 0 | Coefficient for the POLY/SIGMOID KERNEL type. | Only valid when KERNEL_TYPE is 1 or 3. |
| SVM_C | DOUBLE | 100 | Trade-off between training error and margin.<br>Value range: > 0 | |
| SCALE_INFO | INTEGER | 1 | No scale<br>1: Standardization<br>The algorithm transforms the data to have zero mean and unit variance. The general formula is given as:<br>$x_{new}=\frac{x-\mu}{\sigma}$,<br>where x is the origin feature vector, μ is the mean of that feature vector, and σ is its standard deviation.<br>2: Rescaling<br>The algorithm rescales the range of the features to scale the range in [-1,1]. The general formula is given as:<br>$x_{new}=-1+\frac{x-\min(x)}{\max(x)-\min(x)}*2$,<br>where x is the origin feature vector. | |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| SCALE_LABEL | INTEGER | 1 | Controls whether to standardize the label for SVR.<br>• 0: No<br>• 1: Yes | Only valid when TYPE is 2 and SCALE_INFO is 1. |
| SHRINK | INTEGER | 1 | Decides whether to use shrink strategy or not:<br>• 0: Does not use shrink strategy<br>• 1: Uses shrink strategy | Using shrink strategy may accelerate the training process. |
| HANDLE_MISSING | INTEGER | 1 | Whether to handle missing values:<br>• 0: No<br>• 1: Yes | |
| CATEGORY_WEIGHT | DOUBLE | 0.707 | Represents the weight of category attributes (γ). The value must be greater than 0. | |
| TOL | DOUBLE | 0.001 | Specifies the error tolerance in the training process. The value must be greater than 0. | |
| EVALUATION_SEED | INTEGER | 0 | The random seed in parameter selection. The value must be greater than 0. | |
| CATEGORICAL_VARIABLE | VARCHAR or NVARCHAR | No default value | Indicates which variables are treated as categorical. The default behavior is:<br>• STRING: categorical<br>• INTEGER and DOUBLE: continuous | Valid only for INTEGER variables; omitted otherwise. |
| LABEL | VARCHAR or NVARCHAR | The last column name of the DATA table | Specifies the variable used as dependent. | Cannot be the first column name if KEY is 1. |
| COMPRESSION | INTEGER | 1 | Specifies if the model is stored in compressed format.<br>• 0 : no<br>• 1: yes | |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| MAX_BITS | INTEGER | 12 | The maximum number of bits to quantize continuous features. Equivalent to use 2^MAX_BITS bins. Must be less than 31. | Valid only when the value of COMPRESSION is 1. |
| MAX_QUANTIZATION_ITER | INTEGER | 1000 | The maximum iteration steps for quantization. | Valid only when the value of COMPRESSION is 1. |

Parameter search and selection

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| RESAMPLING_METHOD | NVARCHAR | No default value | Specifies the resampling method for model evaluation or parameter selection.<br>• cv<br>• cv_sha<br>• cv_hyperband<br>• stratified_cv<br>• stratified_cv_sha<br>• stratified_cv_hyperband<br>• bootstrap<br>• bootstrap_sha<br>• bootstrap_hyperband<br>• stratified_bootstrap<br>• stratified_bootstrap_sha<br>• stratified_bootstrap_hyperband<br>stratified_* can only apply to classification algorithms. If no value is specified for this parameter, neither model evaluation nor parameter selection is activated. | |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| EVALUATION_METRIC | NVARCHAR | No default value | Specifies the evaluation metric for model evaluation or parameter selection.<br>• ACCURACY<br>• F1_SCORE<br>• AUC | |
| FOLD_NUM | INTEGER | No default value | Specifies the fold number for the cross validation method. | Mandatory and valid only when RESAMPLING_METHOD is set to cv or stratified_cv. |
| REPEAT_TIMES | INTEGER | 1 | Specifies the number of repeat times for resampling. | |
| PARAM_SEARCH_STRATEGY | NVARCHAR | No default value | Specifies the parameter search method:<br>• grid<br>• random | When *_sha is specified in RESAMPLING_METHOD, PARAM_SEARCH_STRATEGY is mandatory. When *_hyperband is specified in RESAMPLING_METHOD, PARAM_SEARCH_STRATEGY is set to random whatever any other value is provided. |
| RANDOM_SEARCH_TIMES | INTEGER | No default value | Specifies the number of times to randomly select candidate parameters for selection. | Mandatory and valid when PARAM_SEARCH_STRATEGY is set to random. |
| REDUCTION_RATE | DOUBLE | 3.0 | Specifies reduction rate in SHA or Hyperband method. | Only valid when RESAMPLING_METHOD is specified as *_sha or *_hyberband. |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| AGGRESSIVE_ELIMINATION | INTEGER | | Specifies whether to apply aggressive elimination while using SHA method. Aggressive elimination happens when the data size and parameters size to be searched does not match and there are still bunch of parameters to be searched while data size reaches its upper limits. If aggressive elimination is applied, lower bound of limit of data size will be used multiple times first to reduce number of parameters.<br>• 0: does not apply aggressive elimination<br>• 1: applies aggressive elimination | Only valid when RESAMPLING_METHOD is specified as *_sha. |
| SEED | INTEGER | 0 | Specifies the seed for random generation. Use system time when 0 is specified. | |
| TIMEOUT | INTEGER | 0 | Specifies maximum running time for model evaluation or parameter selection, in seconds. No timeout when 0 is specified. | |
| PROGRESS_INDICATOR_ID | NVARCHAR | No default value | Sets an ID of progress indicator for model evaluation or parameter selection. No progress indicator is active if no value is provided. | |
| SVM_C_VALUES | NVARCHAR | No default value | Specifies values of SVM_C to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| SVM_C_RANGE | NVARCHAR | No default value | Specifies the range of SVM_C to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| RBF_GAMMA_VALUES | NVARCHAR | No default value | Specifies values of RBF_GAMMA to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| RBF_GAMMA_RANGE | NVARCHAR | No default value | Specifies the range of RBF_GAMMA to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| POLY_DEGREE_VALUES | NVARCHAR | No default value | Specifies values of POLY_DEGREE to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| POLY_DEGREE_RANGE | NVARCHAR | No default value | Specifies the range of POLY_DEGREE to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| COEF_LIN_VALUES | NVARCHAR | No default value | Specifies values of COEF_LIN to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| COEF_LIN_RANGE | NVARCHAR | No default value | Specifies the range of COEF_LIN to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| COEF_CONST_VALUES | NVARCHAR | No default value | Specifies values of COEF_CONST to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| COEF_CONST_RANGE | NVARCHAR | No default value | Specifies the range of COEF_CONST to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |

## _SYS_AFL. PAL_UNIFIED_CLASSIFICATION_PREDICT

### UNIFIED CLASSIFICATION

Mandatory Parameters

| Name | Data Type | Default Value | Description |
| --- | --- | --- | --- |
| FUNCTION | VARCHAR or NVARCHAR | No default value | The parameter must be same with the FUCTION used to train the model.<br>• DT: decision trees<br>• HGBT: hybrid gradient boosting tree<br>• LOGR: logistic regression<br>• M_LOGR: multi-class logistic regression<br>• MLP: multilayer perceptron<br>• NB: naïve bayes<br>• RDT: random decision trees<br>• SVM: support vector machine |

Optional Parameters

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
| --- | --- | --- | --- | --- |
| FEATURE_ATTRIBUTION_METHOD | INTEGER | 2 | Controls which method to use in model reasoning.<br>• 0: no reasoning<br>• 1: SAABAS<br>• 2: SHAP | Valid only for the functions: Decision Trees, Random Decision Trees and Hybrid Gradient Boosting Tree to generating the reason code. |
| TOP_K_ATTRIBUTIONS | INTEGER | 10 | Outputs the attributions of top k features which contribute the most. | |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| SAMPLESIZE | INTEGER | 0 | Number of sampled combinations of features. If it equals to 0, it is determined by algorithm heuristically. | It is better to user a number which is larger than the number of the columns of the data table.<br>Available for the following functions:<br>• Naive Bayes<br>• Support Vector Machine<br>• Multilayer Perceptron<br>• Multi-class Logistic Regression |
| SEED | INTEGER | 0 | Specifies the seed for random number generator.<br>• 0: Uses the current time (in second) as seed<br>• Others: Uses the specified value as seed | Available for the following functions:<br>• Naive Bayes<br>• Support Vector Machine<br>• Multilayer Perceptron<br>• Multi-class Logistic Regression |
| HANDLE_MISSING_VALUE | INTEGER | 0 | Whether to handle missing values:<br>• 0: No<br>• 1: Yes | |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| IMPUTATION_TYPE | INTEGER | 1 | Chooses the overall default imputation type for all columns:<br>• 0: NONE<br>• 1: MODE_MEAN<br>• 2: MODE_MEDIAN<br>• 3: MODE_ALL-ZERO<br>• 4: MODE_ALS<br>• 5: DELETE<br>NONE and DELETE: set all columns to imputation type NONE/DELETE.<br>MODE_MEAN, MODE_MEDIAN, MODE_ALLZERO, and MODE_ALS: set categorical columns to the first imputation type (the type before “”); set numerical columns to the second imputation type (the type after “”). ALLZERO means to fill all missing values in numerical columns with 0. | Only valid when HANDLE_MISSING_VALUE is 1. |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| <column name>_IMPUTATION_TYPE | INTEGER | No default value | Sets imputation type of a specified column. This value overwrites the default imputation type deduced by the IMPUTATION_TYPE parameter:<br>• 0: NONE<br>• 1: DELETE<br>• 100: MODE<br>• 101: SPECIFIED_CATEGORICAL_VALUE<br>• 200: MEAN<br>• 201: MEDIAN<br>• 203:SPECIFIED_NUMERICAL_VALUE<br>• 204: ALS<br>For type 101 (SPECIFIED_CATEGORICAL_VALUE), use the following syntax to specify a categorical value to be filled in a column V0:<br>INSERT INTO #PAL_PARAMETER_TBL VALUES('V0_IMPUTATION_TYPE', 101, NULL, '<CATEGORICAL_VALUE>');<br>For type 203 (SPECIFIED_NUMERICAL_VALUE), use the following syntax to specify a numerical value to be filled in a column V1:<br>INSERT INTO #PAL_PARAMETER_TBL VALUES('V1_IMPUTA- | Only valid when HANDLE_MISSING_VALUE is 1. |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| | | | TION_TYPE', 203, `<NUMERICAL_VALUE>`, NULL);<br><br>**Note**<br>A column of integer type can be set to type 101 (SPECIFIED_CATEGORICAL_VALUE) or type 203 (SPECIFIED_NUMERICAL_VALUE):<br>• when setting to type 101 (SPECIFIED_CATEGORICAL_VALUE), the string must contain a valid integer value, otherwise an error will be thrown;<br>• when setting to type 203 (SPECIFIED_NUMERICAL_VALUE), the double value will be converted to integer. | |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| ALS_FACTOR_NUMBER | INTEGER | 3 | Length of factor vectors, that is, the f in ALS model.<br>**Note**<br>ALS_FACTOR_NUMBER should be less than the number of numerical columns to get a meaningful result. | |
| ALS_REGULARIZATION | DOUBLE | 1e-2 | L2 regularization of the factors, that is, the λ in ALS model. | |
| ALS_MAX_ITERATION | INTEGER | 20 | Maximum number of iterations when training ALS model. | |
| ALS_SEED | INTEGER | 0 | Specifies the seed of the random number generator used in the training of ALS model:<br>• 0: Uses the current time as the seed<br>• Others: Uses the specified value as the seed. | |
| ALS_EXIT_THRESHOLD | DOUBLE | 0.0 | The training of ALS model exits if the value of cost function is not decreased more than the value of ALS_EXIT_THRESHOLD since the last check.<br>If ALS_EXIT_THRESHOLD is set to 0, there is no check and the training only exits on reaching the maximum number of iterations. Evaluations of cost function require additional calculations. You can set this parameter to 0 to avoid it. | |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| ALS_EXIT_INTERVAL | INTEGER | 5 | During the training of ALS model, it will calculate cost function and check the exit criteria every `<ALS_EXIT_INTERVAL>` iterations. Evaluations of cost function require additional calculations. | |
| ALS_LINEAR_SYSTEM_SOLVER | INTEGER | 0 | • 0: cholesky solver<br>• 1: cg solver (recommended when ALS_FACTOR_NUMBER is large) | |
| ALS_CG_MAX_ITERATION | INTEGER | 3 | Maximum number of iteration of cg solver. | |
| ALS_CENTERING | INTEGER | 1 | Whether to center the data by column before training ALS model.<br>• 0: no centering<br>• 1: centering. | |
| ALS_SCALING | INTEGER | 1 | Whether to scale the data by column before training ALS model.<br>• 0: no scaling<br>• 1: scaling | |

## DECISION TREES

Optional

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION |
|---|---|---|---|
| THREAD_RATIO | DOUBLE | -1 | The ratio of available threads.<br>• 0: single thread<br>• 0–1: uses the specified percentage of available threads. PAL uses all available threads if the number is 1.0.<br>• -1: uses all of available threads. |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION |
|---|---|---|---|
| VERBOSE | INTEGER | 0 | Specifies whether to output all classes and the corresponding confidences for each data.<br>• 0: no<br>• 1: yes |

### HYBRID GRADIENT BOOSTING TREE

Optional

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| THREAD_RATIO | DOUBLE | -1 | The ratio of available threads.<br>• 0: single thread<br>• 0–1: uses the specified percentage of available threads. PAL uses all available threads if the number is 1.0.<br>• -1: uses all of available threads. | |
| VERBOSE | INTEGER | 0 | Specifies whether to output all classes and the corresponding confidences for each data.<br>• 0: no<br>• 1: yes | |
| MISSING_REPLACEMENT | INTEGER | 1 | Specifies the missing replacement strategy:<br>• 1: marginalises each missing feature out independently<br>• 2: marginalises all missing features in an instance as a whole corresponding to each category | Strategy 2 is a much coarser approximation. Only apply strategy 2 when there are a great number of missing features and a lot of classed, which might cause memory insufficiency. |

### LOGISTIC REGRESSION

Mandatory

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION |
|---|---|---|---|
| CLASS_MAP0 | VARCHAR | No default value | The same value as the parameter of PAL_LOGISTIC_REGRESSION. |
| CLASS_MAP1 | VARCHAR | No default value | The same value as the parameter of PAL_LOGISTIC_REGRESSION. |
| CATEGORICAL_VARIABLE | VARCHAR | No default value | Indicates whether a column data is actually corresponding to a category variable even the data type of this column is INTEGER.<br>By default, VARCHAR or NVARCHAR is category variable, and INTEGER or DOUBLE is continuous variable. |

Optional

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION |
|---|---|---|---|
| THREAD_RATIO | DOUBLE | -1 | The ratio of available threads.<br>• 0: single thread<br>• 0–1: uses the specified percentage of available threads. PAL uses all available threads if the number is 1.0.<br>• -1: uses all of available threads. |

## MULTI-CLASS LOGISTIC REGRESSION

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION |
|---|---|---|---|
| THREAD_RATIO | DOUBLE | -1 | The ratio of available threads.<br>• 0: single thread<br>• 0–1: uses the specified percentage of available threads. PAL uses all available threads if the number is 1.0.<br>• -1: uses all of available threads. |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION |
|---|---|---|---|
| IGNORE_UNKNOWN_CATEGORY | INTEGER | 1 | • 0: Procedures will report error if unknown category value is found.<br>• 1: Ignores unknown category value |
| VERBOSE | INTEGER | 0 | • 1: Outputs the probability of all label categories<br>• 0: Outputs the category of the highest probability only |

### MULTILAYER PERCEPTRON

Optional

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION |
|---|---|---|---|
| THREAD_RATIO | DOUBLE | -1 | The ratio of available threads.<br>• 0: single thread<br>• 0–1: uses the specified percentage of available threads. PAL uses all available threads if the number is 1.0.<br>• -1: uses all of available threads. |
| VERBOSE | INTEGER | 0 | Output scoring probabilities for each class. |

### NAÏVE BAYES

Optional

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION |
|---|---|---|---|
| LAPLACE | DOUBLE | 0 | Enables or disables Laplace smoothing.<br>• 0: Disables Laplace smoothing.<br>• Positive value: Enables Laplace smoothing for discrete values.<br>**Note**<br>Non-zero LAPLACE value is required if there exist discrete category values that only occur in the test set. The LAPLACE value you set here takes precedence over the values read from JSON models. |
| VERBOSE | INTEGER | 0 | • 1: Outputs the probability of all label categories.<br>• 0: Outputs the category of the highest probability only. |
| THREAD_RATIO | DOUBLE | -1 | The ratio of available threads.<br>• 0: single thread<br>• 0–1: uses the specified percentage of available threads. PAL uses all available threads if the number is 1.0.<br>• -1: uses all of available threads. |

### RANDOM DECISION TREES

Optional

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| THREAD_RATIO | DOUBLE | -1 | The ratio of available threads.<br>• 0: single thread<br>• 0–1: uses the specified percentage of available threads. PAL uses all available threads if the number is 1.0.<br>• -1: uses all of available threads. | |
| VERBOSE | INTEGER | 0 | Specifies whether to output all classes and the corresponding confidences for each data.<br>• 0: no<br>• 1: yes | |
| BLOCK_SIZE | INTEGER | 0 | Specifies the number of data loaded per time during scoring.<br>• 0: load all data once<br>• Others: the specified number | This parameter is for reducing memory consumption, especially as the predict data is huge, or it consists of a large number of missing independent variables. However, you might lose some efficiency. |
| MISSING_REPLACEMENT | INTEGER | 1 | Specifies the missing replacement strategy:<br>• 1: marginalises each missing feature out independently<br>• 2: marginalises all missing features in an instance as a whole corresponding to each category | Strategy 2 is a much coarser approximation. Only apply strategy 2 when there are a great number of missing features and a lot of classed, which might cause memory insufficiency. |

## SUPPORT VECTOR MACHINE

Optional

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION |
|---|---|---|---|
| THREAD_RATIO | DOUBLE | -1 | The ratio of available threads.<br>• 0: single thread<br>• 0–1: uses the specified percentage of available threads. PAL uses all available threads if the number is 1.0.<br>• -1: uses all of available threads. |
| VERBOSE | INTEGER | 0 | Output scoring probabilities for each class. |

## _SYS_AFL. PAL_UNIFIED_CLASSIFICATION_SCORE

### UNIFIED CLASSICATION

Mandatory

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| FUNCTION | VARCHAR or NVARCHAR | No default value | The parameter must be same with the FUCTION used to train the model.<br>• DT: decision trees<br>• HGBT: hybrid gradient boosting tree<br>• LOGR: logistic regression<br>• M_LOGR: multi-class logistic regression<br>• MLP: multilayer perceptron<br>• NB: naïve bayes<br>• RDT: random decision trees<br>• SVM: support vector machine |

Optional Parameters

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| FEATURE_ATTRIBUTION_METHOD | INTEGER | 2 | Controls which method to use in model reasoning.<br>• 0: no reasoning<br>• 1: SAABAS<br>• 2: SHAP | Valid only for the functions: Decision Trees, Random Decision Trees and Hybrid Gradient Boosting Tree to generating the reason code. |
| TOP_K_ATTRIBUTIONS | INTEGER | 10 | Outputs the attributions of top k features which contribute the most. | |
| MAX_RESULT_NUM | INTEGER | No default value | Specifies the output number of prediction results | |
| LABEL | VARCHAR or NVARCHAR | The last column name of the DATA table | | The setting of the parameter should be same with the one in train. |
| NTILES | INTEGER | 1 or 20 | Used to control the population tiles in metrics output.<br>The validation value should be at least 1 and no larger than the row size of the validation data.<br>**Note**<br>For AUC, this parameter means the maximum tiles. | If the row size of data for metrics evaluation is less than 20, the default value is the 1; otherwise it is 20. |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| SAMPLESIZE | INTEGER | 0 | Number of sampled combinations of features. If it equals to 0, it is determined by algorithm heuristically. | It is better to user a number which is larger than the number of the columns of the data table.<br>Available for the following functions:<br>• Naive Bayes<br>• Support Vector Machine<br>• Multilayer Perceptron<br>• Multi-class Logistic Regression |
| SEED | INTEGER | 0 | Specifies the seed for random number generator.<br>• 0: Uses the current time (in second) as seed<br>• Others: Uses the specified value as seed | Available for the following functions:<br>• Naive Bayes<br>• Support Vector Machine<br>• Multilayer Perceptron<br>• Multi-class Logistic Regression |
| HANDLE_MISSING_VALUE | INTEGER | 0 | Whether to handle missing values:<br>• 0: No<br>• 1: Yes | |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| IMPUTATION_TYPE | INTEGER | 1 | Chooses the overall default imputation type for all columns:<br>• 0: NONE<br>• 1: MODE_MEAN<br>• 2: MODE_MEDIAN<br>• 3: MODE_ALL-ZERO<br>• 4: MODE_ALS<br>• 5: DELETE<br>NONE and DELETE: set all columns to imputation type NONE/DELETE.<br>MODE_MEAN, MODE_MEDIAN, MODE_ALLZERO, and MODE_ALS: set categorical columns to the first imputation type (the type before “”); set numerical columns to the second imputation type (the type after “”). ALLZERO means to fill all missing values in numerical columns with 0. | Only valid when HANDLE_MISSING_VALUE is 1. |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| <column name>_IMPUTATION_TYPE | INTEGER | No default value | Sets imputation type of a specified column. This value overwrites the default imputation type deduced by the IMPUTATION_TYPE parameter:<br>• 0: NONE<br>• 1: DELETE<br>• 100: MODE<br>• 101: SPECIFIED_CATEGORICAL_VALUE<br>• 200: MEAN<br>• 201: MEDIAN<br>• 203:SPECIFIED_NUMERICAL_VALUE<br>• 204: ALS<br>For type 101 (SPECIFIED_CATEGORICAL_VALUE), use the following syntax to specify a categorical value to be filled in a column V0:<br>INSERT INTO #PAL_PARAMETER_TBL VALUES('V0_IMPUTATION_TYPE', 101, NULL, '<CATEGORICAL_VALUE>');<br>For type 203 (SPECIFIED_NUMERICAL_VALUE), use the following syntax to specify a numerical value to be filled in a column V1:<br>INSERT INTO #PAL_PARAMETER_TBL VALUES('V1_IMPUTA- | Only valid when HANDLE_MISSING_VALUE is 1. |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| | | | TION_TYPE', 203, `<NUMERICAL_VALUE>`, NULL);<br><br>**Note**<br>A column of integer type can be set to type 101 (SPECIFIED_CATEGORICAL_VALUE) or type 203 (SPECIFIED_NUMERICAL_VALUE):<br>• when setting to type 101 (SPECIFIED_CATEGORICAL_VALUE), the string must contain a valid integer value, otherwise an error will be thrown;<br>• when setting to type 203 (SPECIFIED_NUMERICAL_VALUE), the double value will be converted to integer. | |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| ALS_FACTOR_NUMBER | INTEGER | 3 | Length of factor vectors, that is, the f in ALS model.<br>**Note**<br>ALS_FACTOR_NUMBER should be less than the number of numerical columns to get a meaningful result. | |
| ALS_REGULARIZATION | DOUBLE | 1e-2 | L2 regularization of the factors, that is, the λ in ALS model. | |
| ALS_MAX_ITERATION | INTEGER | 20 | Maximum number of iterations when training ALS model. | |
| ALS_SEED | INTEGER | 0 | Specifies the seed of the random number generator used in the training of ALS model:<br>• 0: Uses the current time as the seed<br>• Others: Uses the specified value as the seed. | |
| ALS_EXIT_THRESHOLD | DOUBLE | 0.0 | The training of ALS model exits if the value of cost function is not decreased more than the value of ALS_EXIT_THRESHOLD since the last check.<br>If ALS_EXIT_THRESHOLD is set to 0, there is no check and the training only exits on reaching the maximum number of iterations. Evaluations of cost function require additional calculations. You can set this parameter to 0 to avoid it. | |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| ALS_EXIT_INTERVAL | INTEGER | 5 | During the training of ALS model, it will calculate cost function and check the exit criteria every `<ALS_EXIT_INTERVAL>` iterations. Evaluations of cost function require additional calculations. | |
| ALS_LINEAR_SYSTEM_SOLVER | INTEGER | 0 | • 0: cholesky solver<br>• 1: cg solver (recommended when ALS_FACTOR_NUMBER is large) | |
| ALS_CG_MAX_ITERATION | INTEGER | 3 | Maximum number of iteration of cg solver. | |
| ALS_CENTERING | INTEGER | 1 | Whether to center the data by column before training ALS model.<br>• 0: no centering<br>• 1: centering. | |
| ALS_SCALING | INTEGER | 1 | Whether to scale the data by column before training ALS model.<br>• 0: no scaling<br>• 1: scaling | |

### DECISION TREES

Optional

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION |
|---|---|---|---|
| THREAD_RATIO | DOUBLE | -1 | The ratio of available threads.<br>• 0: single thread<br>• 0–1: uses the specified percentage of available threads. PAL uses all available threads if the number is 1.0.<br>• -1: uses all of available threads. |

### HYBRID GRADIENT BOOSTING TREE

Optional

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| THREAD_RATIO | DOUBLE | -1 | The ratio of available threads.<br>• 0: single thread<br>• 0–1: uses the specified percentage of available threads. PAL uses all available threads if the number is 1.0.<br>• -1: uses all of available threads. | |
| MISSING_REPLACEMENT | INTEGER | 1 | Specifies the missing replacement strategy:<br>• 1: marginalises each missing feature out independently<br>• 2: marginalises all missing features in an instance as a whole corresponding to each category | Strategy 2 is a much coarser approximation. Only apply strategy 2 when there are a great number of missing features and a lot of classed, which might cause memory insufficiency. |

### LOGISTIC REGRESSION

Mandatory

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION |
|---|---|---|---|
| CLASS_MAP0 | VARCHAR | No default value | The same value as the parameter of PAL_LOGISTIC_REGRESSION. |
| CLASS_MAP1 | VARCHAR | No default value | The same value as the parameter of PAL_LOGISTIC_REGRESSION. |
| CATEGORICAL_VARIABLE | VARCHAR | No default value | Indicates whether a column data is actually corresponding to a category variable even the data type of this column is INTEGER.<br>By default, VARCHAR or NVARCHAR is category variable, and INTEGER or DOUBLE is continuous variable. |

Optional

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION |
|---|---|---|---|
| THREAD_RATIO | DOUBLE | -1 | The ratio of available threads.<br>• 0: single thread<br>• 0–1: uses the specified percentage of available threads. PAL uses all available threads if the number is 1.0.<br>• -1: uses all of available threads. |

### MULTI-CLASS LOGISTIC REGRESSION

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION |
|---|---|---|---|
| THREAD_RATIO | DOUBLE | -1 | The ratio of available threads.<br>• 0: single thread<br>• 0–1: uses the specified percentage of available threads. PAL uses all available threads if the number is 1.0.<br>• -1: uses all of available threads. |
| IGNORE_UNKNOWN_CATEGORY | INTEGER | 1 | • 0: Procedures will report error if unknown category value is found.<br>• 1: Ignores unknown category value |

### MULTILAYER PERCEPTRON

Optional

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION |
|---|---|---|---|
| THREAD_RATIO | DOUBLE | -1 | The ratio of available threads.<br>• 0: single thread<br>• 0–1: uses the specified percentage of available threads. PAL uses all available threads if the number is 1.0.<br>• -1: uses all of available threads. |

### NAÏVE BAYES

Optional

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION |
|---|---|---|---|
| LAPLACE | DOUBLE | 0 | Enables or disables Laplace smoothing.<br>• 0: Disables Laplace smoothing.<br>• Positive value: Enables Laplace smoothing for discrete values.<br>**Note**<br>Non-zero LAPLACE value is required if there exist discrete category values that only occur in the test set. The LAPLACE value you set here takes precedence over the values read from JSON models. |
| THREAD_RATIO | DOUBLE | -1 | The ratio of available threads.<br>• 0: single thread<br>• 0–1: uses the specified percentage of available threads. PAL uses all available threads if the number is 1.0.<br>• -1: uses all of available threads. |

### RANDOM DECISION TREES

Optional

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| THREAD_RATIO | DOUBLE | -1 | The ratio of available threads.<br>• 0: single thread<br>• 0–1: uses the specified percentage of available threads. PAL uses all available threads if the number is 1.0.<br>• -1: uses all of available threads. | |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| BLOCK_SIZE | INTEGER | 0 | Specifies the number of data loaded per time during scoring.<br>• 0: load all data once<br>• Others: the specified number | This parameter is for reducing memory consumption, especially as the predict data is huge, or it consists of a large number of missing independent variables. However, you might lose some efficiency. |
| MISSING_REPLACEMENT | INTEGER | 1 | Specifies the missing replacement strategy:<br>• 1: marginalises each missing feature out independently<br>• 2: marginalises all missing features in an instance as a whole corresponding to each category | Strategy 2 is a much coarser approximation. Only apply strategy 2 when there are a great number of missing features and a lot of classed, which might cause memory insufficiency. |

**SUPPORT VECTOR MACHINE**

Optional

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION |
|---|---|---|---|
| THREAD_RATIO | DOUBLE | -1 | The ratio of available threads.<br>• 0: single thread<br>• 0–1: uses the specified percentage of available threads. PAL uses all available threads if the number is 1.0.<br>• -1: uses all of available threads. |

## 5.13.2 Unified Classification with Pivoted Input Data

The pivoted unified classification interface refers to a new unified interface for classification algorithms, which accepts pivoted input data.

The classification algorithms include:

- Decision Trees
- Hybrid Gradient Boosting Tree

- Logistic Regression
- Multi-class Logistic Regression
- Naïve Bayes
- Random Decision Trees
- Support Vector Machine

## Prerequisite

All prerequisites for the original corresponding classification functions apply in unified classification interface.

## _SYS_AFL.PAL_UNIFIED_CLASSIFICATION_PIVOT

This function reads the input data, generates the training model and provides the evaluation statistics and metrics based on the training model and validation data.

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<META_DATA Table>` | IN |
| 2 | `<DATA Table>` | IN |
| 3 | `<PARAMETER Table>` | IN |
| 4 | `<MODEL Table>` | OUT |
| 5 | `<VARIABLE_IMPORTANCE Table>` | OUT |
| 6 | `<STATISTIC Table>` | OUT |
| 7 | `<OPTIMAL_PARAM Table>` | OUT |
| 8 | `<CONFUSION_MATRIX Table>` | OUT |
| 9 | `<METRICS Table>` | OUT |
| 10 | `<PARTITION_TYPE Table>` | OUT |
| 11 | `<PLACE_HOLDER1 Table>` | OUT |

### Signature

### Input Table

| Table | Column | Data Type | Description |
|---|---|---|---|
| META_DATA | 1st column | VRACHAR, NVARCHAR | The name of the variable. |
| | 2nd column | VRACHAR, NVARCHAR | The type of the variable:<br>• CONTINUOUS<br>• CATEGORICAL<br>• TARGET |
| DATA | 1st column | INTEGER, VRACHAR, or NVARCHAR | Record ID. |
| | 2nd column | VARCHAR, NVARCHAR | Variable name. |
| | 3rd column | VARCHAR, NVARCHAR | Variable value. |
| | 4th column | INTEGER | Self-defined data partition:<br>• 1:training<br>• 2:validation |

### Parameter Table

See details in Parameter Tables [page 1681].

### Output Table

| Table | Column | Column Name | Column Data Type |
|---|---|---|---|
| MODEL | 1st column | ROW_INDEX | INTEGER |
| | 2nd column | PART_INDEX | INTEGER |
| | 3rd column | MODEL_CONTENT | NCLOB |
| VARIABLE_IMPORTANCE | 1st column | VARIABLE_NAME | NVARCHAR(256) |
| | 2nd column | IMPORTANCE | DOUBLE |
| STATISTICS | 1st column | STAT_NAME | NVARCHAR (256) |
| | 2nd column | STAT_VALUE | NVARCHAR (1000) |
| | 3rd column | CLASS_NAME | NVARCHAR (256) |
| OPTIMAL_PARAM | 1st column | PARAM_NAME | NVARCHAR (256) |
| | 2nd column | INT_VALUE | INTEGER |
| | 3rd column | DOUBLE_VALUE | DOUBLE |
| | 4th column | STRING_VALUE | NVARCHAR (1000) |
| CONFUSION_MATRIX | 1st column | ACTUAL_CLASS | NVARCHAR (1000) |
| | 2nd column | PREDICTED_CLASS | NVARCHAR (1000) |
| | 3rd column | COUNT | INTEGER |
| METRICS | 1st column | NAME | NVARCHAR (256) |

| Table | Column | Column Name | Column Data Type |
|---|---|---|---|
| | 2nd column | X | DOUBLE |
| | 3rd column | Y | DOUBLE |
| PARTITION_TYPE | 1st column | ID column name | ID column data type |
| | 2nd column | TYPE | INTEGER |
| PLACE_HOLDER1 | 1st column | OBJECT | NVARCHAR(10) |
| | 2nd column | KEY | NVARCHAR(10) |
| | 3rd column | VALUE | NVARCHAR(10) |

### Examples

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

**Example 1: Random Decision Trees with stratified data partition**

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_META_DATA_TBL;
CREATE COLUMN TABLE PAL_META_DATA_TBL(
    VARIABLE_NAME NVARCHAR(100),
    VARIABLE_TYPE NVARCHAR(100)
);
INSERT INTO PAL_META_DATA_TBL VALUES ('ATTR1', 'CONTINUOUS');
INSERT INTO PAL_META_DATA_TBL VALUES ('ATTR2', 'CONTINUOUS');
INSERT INTO PAL_META_DATA_TBL VALUES ('ATTR3', 'CATEGORICAL');
INSERT INTO PAL_META_DATA_TBL VALUES ('ATTR4', 'CATEGORICAL');
INSERT INTO PAL_META_DATA_TBL VALUES ('LABEL', 'TARGET');
DROP TABLE PAL_DATA_TBL;
CREATE COLUMN TABLE PAL_DATA_TBL (
    DATA_ID INTEGER,
    VARIABLE_NAME NVARCHAR(100),
    VAR_VALUE NVARCHAR(100),
    PURPOSE  INTEGER
);
INSERT INTO PAL_DATA_TBL VALUES(0,'ATTR1','1',2);
INSERT INTO PAL_DATA_TBL VALUES(0,'ATTR2','10',2);
INSERT INTO PAL_DATA_TBL VALUES(0,'ATTR3','100',2);
INSERT INTO PAL_DATA_TBL VALUES(0,'ATTR4','A',2);
INSERT INTO PAL_DATA_TBL VALUES(0,'LABEL','1',2);
INSERT INTO PAL_DATA_TBL VALUES(1,'ATTR1',1.1,1);
INSERT INTO PAL_DATA_TBL VALUES(1,'ATTR2',10.1,1);
INSERT INTO PAL_DATA_TBL VALUES(1,'ATTR3',100.1,1);
INSERT INTO PAL_DATA_TBL VALUES(1,'ATTR4','A',1);
INSERT INTO PAL_DATA_TBL VALUES(1,'LABEL',1,1);
INSERT INTO PAL_DATA_TBL VALUES(2,'ATTR1',1.2,1);
INSERT INTO PAL_DATA_TBL VALUES(2,'ATTR2',10.2,1);
INSERT INTO PAL_DATA_TBL VALUES(2,'ATTR3',100,1);
INSERT INTO PAL_DATA_TBL VALUES(2,'ATTR4','A',1);
INSERT INTO PAL_DATA_TBL VALUES(2,'LABEL',1,1);
INSERT INTO PAL_DATA_TBL VALUES(3,'ATTR1',1.3,1);
INSERT INTO PAL_DATA_TBL VALUES(3,'ATTR2',10.4,1);
INSERT INTO PAL_DATA_TBL VALUES(3,'ATTR3',100,1);
INSERT INTO PAL_DATA_TBL VALUES(3,'ATTR4','A',1);
INSERT INTO PAL_DATA_TBL VALUES(3,'LABEL',1,1);
INSERT INTO PAL_DATA_TBL VALUES(4,'ATTR1',1.2,1);
INSERT INTO PAL_DATA_TBL VALUES(4,'ATTR2',10.3,1);
```

```
INSERT INTO PAL_DATA_TBL VALUES(4,'ATTR3',100,1);
INSERT INTO PAL_DATA_TBL VALUES(4,'ATTR4','AB',1);
INSERT INTO PAL_DATA_TBL VALUES(4,'LABEL',1,1);
INSERT INTO PAL_DATA_TBL VALUES(5,'ATTR1',4,1);
INSERT INTO PAL_DATA_TBL VALUES(5,'ATTR2',40,1);
INSERT INTO PAL_DATA_TBL VALUES(5,'ATTR3',400,1);
INSERT INTO PAL_DATA_TBL VALUES(5,'ATTR4','AB',1);
INSERT INTO PAL_DATA_TBL VALUES(5,'LABEL',2,1);
INSERT INTO PAL_DATA_TBL VALUES(6,'ATTR1',4.1,2);
INSERT INTO PAL_DATA_TBL VALUES(6,'ATTR2',40,2);
INSERT INTO PAL_DATA_TBL VALUES(6,'ATTR3',400,2);
INSERT INTO PAL_DATA_TBL VALUES(6,'ATTR4','AB',2);
INSERT INTO PAL_DATA_TBL VALUES(6,'LABEL',2,2);
INSERT INTO PAL_DATA_TBL VALUES(7,'ATTR1',4.2,1);
INSERT INTO PAL_DATA_TBL VALUES(7,'ATTR2',40.2,1);
INSERT INTO PAL_DATA_TBL VALUES(7,'ATTR3',400,1);
INSERT INTO PAL_DATA_TBL VALUES(7,'ATTR4','AB',1);
INSERT INTO PAL_DATA_TBL VALUES(7,'LABEL',2,1);
INSERT INTO PAL_DATA_TBL VALUES(8,'ATTR1',4.3,1);
INSERT INTO PAL_DATA_TBL VALUES(8,'ATTR2',40.4,1);
INSERT INTO PAL_DATA_TBL VALUES(8,'ATTR3',400,1);
INSERT INTO PAL_DATA_TBL VALUES(8,'ATTR4','AB',1);
INSERT INTO PAL_DATA_TBL VALUES(8,'LABEL',2,1);
INSERT INTO PAL_DATA_TBL VALUES(9,'ATTR1',4.2,1);
INSERT INTO PAL_DATA_TBL VALUES(9,'ATTR2',40.3,1);
INSERT INTO PAL_DATA_TBL VALUES(9,'ATTR3',400,1);
INSERT INTO PAL_DATA_TBL VALUES(9,'ATTR4','AB',1);
INSERT INTO PAL_DATA_TBL VALUES(9,'LABEL',2,1);
INSERT INTO PAL_DATA_TBL VALUES(10,'ATTR1',9,1);
INSERT INTO PAL_DATA_TBL VALUES(10,'ATTR2',90,1);
INSERT INTO PAL_DATA_TBL VALUES(10,'ATTR3',900,1);
INSERT INTO PAL_DATA_TBL VALUES(10,'ATTR4','B',1);
INSERT INTO PAL_DATA_TBL VALUES(10,'LABEL',3,1);
INSERT INTO PAL_DATA_TBL VALUES(11,'ATTR1',9.1,1);
INSERT INTO PAL_DATA_TBL VALUES(11,'ATTR2',90.1,1);
INSERT INTO PAL_DATA_TBL VALUES(11,'ATTR3',900,1);
INSERT INTO PAL_DATA_TBL VALUES(11,'ATTR4','A',1);
INSERT INTO PAL_DATA_TBL VALUES(11,'LABEL',3,1);
INSERT INTO PAL_DATA_TBL VALUES(12,'ATTR1',9.2,2);
INSERT INTO PAL_DATA_TBL VALUES(12,'ATTR2',90.2,2);
INSERT INTO PAL_DATA_TBL VALUES(12,'ATTR3',900,2);
INSERT INTO PAL_DATA_TBL VALUES(12,'ATTR4','B',2);
INSERT INTO PAL_DATA_TBL VALUES(12,'LABEL',3,2);
INSERT INTO PAL_DATA_TBL VALUES(13,'ATTR1',9.3,1);
INSERT INTO PAL_DATA_TBL VALUES(13,'ATTR2',90.4,1);
INSERT INTO PAL_DATA_TBL VALUES(13,'ATTR3',900,1);
INSERT INTO PAL_DATA_TBL VALUES(13,'ATTR4','A',1);
INSERT INTO PAL_DATA_TBL VALUES(13,'LABEL',3,1);
INSERT INTO PAL_DATA_TBL VALUES(14,'ATTR1',9.2,1);
INSERT INTO PAL_DATA_TBL VALUES(14,'ATTR2',90.3,1);
INSERT INTO PAL_DATA_TBL VALUES(14,'ATTR3',900,1);
INSERT INTO PAL_DATA_TBL VALUES(14,'ATTR4','A',1);
INSERT INTO PAL_DATA_TBL VALUES(14,'LABEL',3,1);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (100)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FUNCTION', NULL, NULL, 'RDT');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SPLIT_THRESHOLD', NULL, 0.0000001, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MIN_SAMPLES_LEAF', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('N_ESTIMATORS', 10, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MAX_DEPTH', 55, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SEED', 2, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('NTILES', 2, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PARTITION_METHOD', 2, NULL, NULL);
```

```
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PARTITION_RANDOM_SEED', 2, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PARTITION_STRATIFIED_VARIABLE', NULL,
NULL, 'LABEL');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PARTITION_TRAINING_PERCENT', NULL, 0.7,
NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('OUTPUT_PARTITION_RESULT', 1, NULL, NULL);
DROP TABLE PAL_MODEL_RDT_TBL;
CREATE COLUMN TABLE PAL_MODEL_RDT_TBL (
    "ROW_INDEX" INTEGER,
    "PART_INDEX" INTEGER,
    "MODEL_CONTENT" NCLOB
);
CALL _SYS_AFL.PAL_UNIFIED_CLASSIFICATION_PIVOT (PAL_META_DATA_TBL, PAL_DATA_TBL,
#PAL_PARAMETER_TBL, PAL_MODEL_RDT_TBL, ?, ?, ?, ?, ?, ?, ?) WITH OVERVIEW;
```

### Expected Result

Model:

MODEL_CONTENT_
<PMML version="4.0" xmlns="http://www.dmg.o

MODEL_OWNER_INDEX_
rg/PMML-4_0xmlns:xs...

MODEL_OUTPUT_PMML_version="4.0" xmlns="http://www.dmg.org

MODEL_OWNER_INDEX_/PMML-4_0xmlns:xs...

MODEL_OUTPUT_PMML_version="4.0" xmlns="http://www.dmg.org

MODEL_CONTENT_/PMML-4_0xmlns:xs...

MODEL_OUTPUT_PMML_version="4.0" xmlns="http://www.dmg.org

MODEL_FORMAT_CONTENT_/PMML-4_0xmlns:xs...

MODEL_OUTPUT_PMML_version="4.0" xmlns="http://www.dmg.org

MODEL_OWNER_INDEX_/PMML-4_0xmlns:xs...

MODEL_FORMAT_PMML_version="4.0" xmlns="http://www.dmg.org/

MODEL_OWNER_INDEX_/PMML-4_0xmlns:xs...

MODEL_OUTPUT_TS_PMML version="4.0" xmlns="http://www.dmg.org

MODEL_OUTPUT_/PMML-4_0xmlns:xs...

MODEL_OUTPUT_SPMMLversion="4.0" xmlns=http://www.dmg.org

MODEL_OWNER_INDEX_/PMML-4_0xmlns:xs...

MODEL_OUTPUT_SPMML version="4.0" xmlns="http://www.dmg.org"

MODELPARQUET_INDEX_/PMML-4_0xmlns:xs...

MODEL_OUTPUT_PMML version="4.0" xmlns="http://www.dmg.org

MODEL_OWNER_INDEX_/PMML-4_0xmlns:xs...

MODEL_ROW_INDEX_PMML version="4.0" xmlns="http://www.dmg.org

MODEL_FORMAT_CONTENT_/PMML-4_0xmlns:xsi...

MODEL_ROW_INDEX_1,
"feature":["ATTR1","ATTR2","ATTR3","ATTR4

MODEL_OWNER_INDEX_",","LABEL"],"featureNum":5,"i...

IMP:

| | VARIABLE_NAME | IMPORTANCE |
|---|---|---|
| 1 | ATTR1 | 0.2850877192982456 |
| 2 | ATTR2 | 0.29166666666666663 |
| 3 | ATTR3 | 0.30482456140350883 |
| 4 | ATTR4 | 0.11842105263157898 |

STATISTIC:

| | STAT_NAME | STAT_VALUE | CLASS_NAME |
|---|---|---|---|
| 1 | AUC | 1 | ? |
| 2 | RECALL | 1 | 1 |
| 3 | PRECISION | 1 | 1 |
| 4 | F1_SCORE | 1 | 1 |
| 5 | SUPPORT | 2 | 1 |
| 6 | RECALL | 1 | 2 |
| 7 | PRECISION | 1 | 2 |
| 8 | F1_SCORE | 1 | 2 |
| 9 | SUPPORT | 2 | 2 |
| 10 | RECALL | 1 | 3 |
| 11 | PRECISION | 1 | 3 |
| 12 | 12 F1_SCORE | 1 | 3 |
| 13 | SUPPORT | 1 | 3 |
| 14 | ACCURACY | 1 | ? |
| 15 | KAPPA | 1 | ? |
| 16 | MCC | 1 | ? |

CONFUSION MATRIX:

| PREDICTED_CLASS_1 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|

PREDICTED_CLASS_70_30_3

METRICS:

| | NAME | X | Y |
|---|---|---|---|
| 1 | RANDOM_CUMGAINS | 0 | 0 |
| 2 | RANDOM_CUMGAINS | 1 | 1 |
| 3 | RANDOM_LIFT | 0 | 1 |
| 4 | RANDOM_LIFT | 1 | 1 |
| 5 | RANDOM_CUMLIFT | 0 | 1 |
| 6 | RANDOM_CUMLIFT | 1 | 1 |
| 7 | PERF_CUMGAINS | 0 | 0 |
| 8 | PERF_CUMGAINS | 0.4 | 1 |
| 9 | PERF_CUMGAINS | 1 | 1 |
| 10 | PERF_LIFT | 0 | 3 |
| 11 | PERF_LIFT | 0.4 | 2 |
| 12 | PERF_LIFT | 1 | 0 |
| 13 | PERF_CUMLIFT | 0 | 3 |
| 14 | PERF_CUMLIFT | 0.4 | 2.50... |
| 15 | PERF_CUMLIFT | 1 | 1 |
| 16 | ROC_FPR | 0 | 0 |
| 17 | ROC_TPR | 0 | 0 |
| 18 | ROC_FPR | 1 | 0 |
| 19 | ROC_TPR | 1 | 1 |
| 20 | ROC_FPR | 2 | 1 |
| 21 | ROC_TPR | 2 | 1 |
| 22 | CUMGAINS | 0 | 0 |
| 23 | CUMGAINS | 0.4 | 1 |
| 24 | CUMGAINS | 1 | 1 |
| 25 | LIFT | 0 | 3 |
| 26 | LIFT | 0.4 | 2 |
| 27 | LIFT | 1 | 0 |
| 28 | CUMLIFT | 0 | 3 |
| 29 | CUMLIFT | 0.4 | 2.50... |
| 30 | CUMLIFT | 1 | 1 |

PARTITION_TYPE:

| | |
|---|---|
| DATA_TYPE | |
| 1 | |
| 2 | |
| 3 | |
| 4 | |
| 5 | |
| 6 | |
| 7 | |
| 10 | |
| 8 | |
| 12 | |
| 9 | |
| 13 | |
| 11 | |
| 14 | |
| 12 | |
| 21 | |
| 12 | |
| 22 | |
| 12 | |
| 23 | |
| 12 | |
| 24 | |
| 12 | |
| 25 | |

**Example 2: Support Vector Machine with user-defined data partition**

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_META_DATA_TBL;
CREATE COLUMN TABLE PAL_META_DATA_TBL(
    VARIABLE_NAME NVARCHAR(100),
    VARIABLE_TYPE NVARCHAR(100)
);
INSERT INTO PAL_META_DATA_TBL VALUES ('ATTR1', 'CONTINUOUS');
INSERT INTO PAL_META_DATA_TBL VALUES ('ATTR2', 'CONTINUOUS');
INSERT INTO PAL_META_DATA_TBL VALUES ('ATTR3', 'CATEGORICAL');
INSERT INTO PAL_META_DATA_TBL VALUES ('ATTR4', 'CATEGORICAL');
INSERT INTO PAL_META_DATA_TBL VALUES ('LABEL', 'TARGET');
DROP TABLE PAL_DATA_TBL;
CREATE COLUMN TABLE PAL_DATA_TBL (
    DATA_ID INTEGER,
```

```
    VARIABLE_NAME NVARCHAR(100),
    VAR_VALUE NVARCHAR(100),
    PURPOSE  INTEGER
);
INSERT INTO PAL_DATA_TBL VALUES(0,'ATTR1','1',2);
INSERT INTO PAL_DATA_TBL VALUES(0,'ATTR2','10',2);
INSERT INTO PAL_DATA_TBL VALUES(0,'ATTR3','100',2);
INSERT INTO PAL_DATA_TBL VALUES(0,'ATTR4','A',2);
INSERT INTO PAL_DATA_TBL VALUES(0,'LABEL','1',2);
INSERT INTO PAL_DATA_TBL VALUES(1,'ATTR1',1.1,1);
INSERT INTO PAL_DATA_TBL VALUES(1,'ATTR2',10.1,1);
INSERT INTO PAL_DATA_TBL VALUES(1,'ATTR3',100.1,1);
INSERT INTO PAL_DATA_TBL VALUES(1,'ATTR4','A',1);
INSERT INTO PAL_DATA_TBL VALUES(1,'LABEL',1,1);
INSERT INTO PAL_DATA_TBL VALUES(2,'ATTR1',1.2,1);
INSERT INTO PAL_DATA_TBL VALUES(2,'ATTR2',10.2,1);
INSERT INTO PAL_DATA_TBL VALUES(2,'ATTR3',100,1);
INSERT INTO PAL_DATA_TBL VALUES(2,'ATTR4','A',1);
INSERT INTO PAL_DATA_TBL VALUES(2,'LABEL',1,1);
INSERT INTO PAL_DATA_TBL VALUES(3,'ATTR1',1.3,1);
INSERT INTO PAL_DATA_TBL VALUES(3,'ATTR2',10.4,1);
INSERT INTO PAL_DATA_TBL VALUES(3,'ATTR3',100,1);
INSERT INTO PAL_DATA_TBL VALUES(3,'ATTR4','A',1);
INSERT INTO PAL_DATA_TBL VALUES(3,'LABEL',1,1);
INSERT INTO PAL_DATA_TBL VALUES(4,'ATTR1',1.2,1);
INSERT INTO PAL_DATA_TBL VALUES(4,'ATTR2',10.3,1);
INSERT INTO PAL_DATA_TBL VALUES(4,'ATTR3',100,1);
INSERT INTO PAL_DATA_TBL VALUES(4,'ATTR4','AB',1);
INSERT INTO PAL_DATA_TBL VALUES(4,'LABEL',1,1);
INSERT INTO PAL_DATA_TBL VALUES(5,'ATTR1',4,1);
INSERT INTO PAL_DATA_TBL VALUES(5,'ATTR2',40,1);
INSERT INTO PAL_DATA_TBL VALUES(5,'ATTR3',400,1);
INSERT INTO PAL_DATA_TBL VALUES(5,'ATTR4','AB',1);
INSERT INTO PAL_DATA_TBL VALUES(5,'LABEL',2,1);
INSERT INTO PAL_DATA_TBL VALUES(6,'ATTR1',4.1,2);
INSERT INTO PAL_DATA_TBL VALUES(6,'ATTR2',40,2);
INSERT INTO PAL_DATA_TBL VALUES(6,'ATTR3',400,2);
INSERT INTO PAL_DATA_TBL VALUES(6,'ATTR4','AB',2);
INSERT INTO PAL_DATA_TBL VALUES(6,'LABEL',2,2);
INSERT INTO PAL_DATA_TBL VALUES(7,'ATTR1',4.2,1);
INSERT INTO PAL_DATA_TBL VALUES(7,'ATTR2',40.2,1);
INSERT INTO PAL_DATA_TBL VALUES(7,'ATTR3',400,1);
INSERT INTO PAL_DATA_TBL VALUES(7,'ATTR4','AB',1);
INSERT INTO PAL_DATA_TBL VALUES(7,'LABEL',2,1);
INSERT INTO PAL_DATA_TBL VALUES(8,'ATTR1',4.3,1);
INSERT INTO PAL_DATA_TBL VALUES(8,'ATTR2',40.4,1);
INSERT INTO PAL_DATA_TBL VALUES(8,'ATTR3',400,1);
INSERT INTO PAL_DATA_TBL VALUES(8,'ATTR4','AB',1);
INSERT INTO PAL_DATA_TBL VALUES(8,'LABEL',2,1);
INSERT INTO PAL_DATA_TBL VALUES(9,'ATTR1',4.2,1);
INSERT INTO PAL_DATA_TBL VALUES(9,'ATTR2',40.3,1);
INSERT INTO PAL_DATA_TBL VALUES(9,'ATTR3',400,1);
INSERT INTO PAL_DATA_TBL VALUES(9,'ATTR4','AB',1);
INSERT INTO PAL_DATA_TBL VALUES(9,'LABEL',2,1);
INSERT INTO PAL_DATA_TBL VALUES(10,'ATTR1',9,1);
INSERT INTO PAL_DATA_TBL VALUES(10,'ATTR2',90,1);
INSERT INTO PAL_DATA_TBL VALUES(10,'ATTR3',900,1);
INSERT INTO PAL_DATA_TBL VALUES(10,'ATTR4','B',1);
INSERT INTO PAL_DATA_TBL VALUES(10,'LABEL',3,1);
INSERT INTO PAL_DATA_TBL VALUES(11,'ATTR1',9.1,1);
INSERT INTO PAL_DATA_TBL VALUES(11,'ATTR2',90.1,1);
INSERT INTO PAL_DATA_TBL VALUES(11,'ATTR3',900,1);
INSERT INTO PAL_DATA_TBL VALUES(11,'ATTR4','A',1);
INSERT INTO PAL_DATA_TBL VALUES(11,'LABEL',3,1);
INSERT INTO PAL_DATA_TBL VALUES(12,'ATTR1',9.2,2);
INSERT INTO PAL_DATA_TBL VALUES(12,'ATTR2',90.2,2);
INSERT INTO PAL_DATA_TBL VALUES(12,'ATTR3',900,2);
INSERT INTO PAL_DATA_TBL VALUES(12,'ATTR4','B',2);
```

```
INSERT INTO PAL_DATA_TBL VALUES(12,'LABEL',3,2);
INSERT INTO PAL_DATA_TBL VALUES(13,'ATTR1',9.3,1);
INSERT INTO PAL_DATA_TBL VALUES(13,'ATTR2',90.4,1);
INSERT INTO PAL_DATA_TBL VALUES(13,'ATTR3',900,1);
INSERT INTO PAL_DATA_TBL VALUES(13,'ATTR4','A',1);
INSERT INTO PAL_DATA_TBL VALUES(13,'LABEL',3,1);
INSERT INTO PAL_DATA_TBL VALUES(14,'ATTR1',9.2,1);
INSERT INTO PAL_DATA_TBL VALUES(14,'ATTR2',90.3,1);
INSERT INTO PAL_DATA_TBL VALUES(14,'ATTR3',900,1);
INSERT INTO PAL_DATA_TBL VALUES(14,'ATTR4','A',1);
INSERT INTO PAL_DATA_TBL VALUES(14,'LABEL',3,1);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (100)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FUNCTION', NULL, NULL, 'SVM');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PARTITION_METHOD', 1, NULL, NULL);
--parameters to be selected
INSERT INTO #PAL_PARAMETER_TBL VALUES ('RBF_GAMMA_VALUES',NULL,NULL,
'{0.01,0.05,0.07}');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SVM_C_VALUES',NULL,NULL, '{10,50,100}');
--parameter selection parameters
INSERT INTO #PAL_PARAMETER_TBL VALUES ('RESAMPLING_METHOD', NULL, NULL, 'cv');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('EVALUATION_METRIC', NULL, NULL,
'F1_SCORE');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FOLD_NUM', 2, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('REPEAT_TIMES', 2, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SEED', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PROGRESS_INDICATOR_ID', NULL, NULL,
'TEST');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PARAM_SEARCH_STRATEGY', NULL, NULL,
'grid');
DROP TABLE PAL_MODEL_SVM_TBL;
CREATE COLUMN TABLE PAL_MODEL_SVM_TBL (
    "ROW_INDEX" INTEGER,
    "PART_INDEX" INTEGER,
    "MODEL_CONTENT" NCLOB
);
CALL _SYS_AFL.PAL_UNIFIED_CLASSIFICATION_PIVOT (PAL_META_DATA_TBL, PAL_DATA_TBL,
#PAL_PARAMETER_TBL, PAL_MODEL_SVM_TBL, ?,?, ?,?,?,?,?) with overview;
```

### Expected Result

MODEL:


```
MODEL_PAGE_WORKING_INDEX_10{"C":10.0,"CONTENT":"{\"impute_model\":{
```

MODEL_OUTPUT_
\"column_statistics\":[{\"COLU...
2S1VM

MODEL_OWNER_INDEX_32 feature": ["ATTR1", "ATTR2", "ATTR3", "ATTR4

MODEL_OWNER_INDEX_"," LABEL"],"featureNum":5,"i...

STATISTICS:

| | STAT_NAME | STAT_VALUE | CLASS_NAME |
|---|---|---|---|
| 1 | RECALL | 1 | 1 |
| 2 | PRECISION | 1 | 1 |
| 3 | F1_SCORE | 1 | 1 |
| 4 | SUPPORT | 1 | 1 |
| 5 | RECALL | 1 | 2 |

| | STAT_NAME | STAT_VALUE | CLASS_NAME |
|---|---|---|---|
| 6 | PRECISION | 1 | 2 |
| 7 | F1_SCORE | 1 | 2 |
| 8 | SUPPORT | 1 | 2 |
| 9 | RECALL | 1 | 3 |
| 10 | PRECISION | 1 | 3 |
| 11 | F1_SCORE | 1 | 3 |
| 12 | SUPPORT | 1 | 3 |
| 13 | ACCURACY | 1 | ? |
| 14 | KAPPA | 1 | ? |
| 15 | MCC | 1 | ? |

OPTIMAL PARAMETER:

| | PARAM_NAME | INT_VALUE | DOUBLE_VALUE | STRING_VALUE |
|---|---|---|---|---|
| 1 | RBF_GAMMA | ? | 0.05 | ? |
| 2 | SVM_C | ? | 10 | ? |

CONFUSION MATRIX:

| | ACTUAL_CLASS | PREDICTED_CLASS | COUNT |
|---|---|---|---|
| 1 | 1 | 1 | 1 |
| 2 | 1 | 2 | 0 |
| 3 | 1 | 3 | 0 |
| 4 | 2 | 1 | 0 |
| 5 | 2 | 2 | 1 |
| 6 | 2 | 3 | 0 |
| 7 | 3 | 1 | 0 |
| 8 | 3 | 2 | 0 |
| 9 | 3 | 3 | 1 |

## _SYS_AFL. PAL_UNIFIED_CLASSIFICATION_PREDICT_PIVOT

This function uses the training model to make predictive analysis.

**Overview of All Tables**

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA Table>` | IN |
| 2 | `<MODEL Table>` | IN |
| 3 | `<PARAMETER Table>` | IN |
| 4 | `<RESULT Table>` | OUT |
| 5 | `<PLACE_HOLDER1 Table>` | OUT |

### Input Table

| Table | Column | Data Type | Description |
|---|---|---|---|
| DATA | 1st column | INTEGER, VRACHAR, or NVARCHAR | Record ID. |
| | 2nd column | VARCHAR, NVARCHAR | Variable name. |
| | 3rd column | VARCHAR, NVARCHAR | Variable value. |

### Model Table

| Table | Column | Data Type |
|---|---|---|
| MODEL | 1st column | INTEGER |
| | 2nd column | INTEGER |
| | 3rd column | NCLOB |

### Parameter Table

See more details in Parameter Tables [page 1681].

### Output Table

| Table | Column | Column Name | Column Data Type | Description |
|---|---|---|---|---|
| RESULT | 1st column | ID column name | ID column data type | |
| | 2nd column | SCORE | NVARCHAR (256) | |
| | 3rd column | CONFIDENCE | DOUBLE | |
| | 4th column | REASON CODE | NCLOB | Only valid for random decision trees, decision trees, and hybrid gradient boosting trees. |
| PLACE_HOLDER | 1st column | OBJECT | NVARCHAR(10) | |
| | 2nd column | KEY | NVARCHAR(10) | |

| Table | Column | Column Name | Column Data Type | Description |
|---|---|---|---|---|
| | 3rd column | VALUE | NVARCHAR(10) | |

### Example: Random Decision Trees Prediction Procedure

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_PREDICT_DATA_TBL;
CREATE COLUMN TABLE PAL_PREDICT_DATA_TBL (
    DATA_ID INTEGER,
    VARIABLE_NAME NVARCHAR(100),
    VAR_VALUE NVARCHAR(100)
);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(0,'ATTR1','1');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(0,'ATTR2','10');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(0,'ATTR3','100');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(0,'ATTR4','A');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(1,'ATTR1',1.1);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(1,'ATTR2',10.1);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(1,'ATTR3',100.1);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(1,'ATTR4','A');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(2,'ATTR1',1.2);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(2,'ATTR2',10.2);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(2,'ATTR3',100);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(2,'ATTR4','A');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(3,'ATTR1',1.3);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(3,'ATTR2',10.4);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(3,'ATTR3',100);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(3,'ATTR4','A');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(4,'ATTR1',4);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(4,'ATTR2',40);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(4,'ATTR3',400);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(4,'ATTR4','AB');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(5,'ATTR1',9.2);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(5,'ATTR2',90.2);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(5,'ATTR3',900);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(5,'ATTR4','B');
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (100)
);
CALL _SYS_AFL.PAL_UNIFIED_CLASSIFICATION_PREDICT_PIVOT(PAL_PREDICT_DATA_TBL,
PAL_MODEL_RDT_TBL, #PAL_PARAMETER_TBL, ?, ?);
```

### Expected Result

RESULT:

| | DATA_ID | SCORE | CONFIDENCE | REASON_CODE |
|---|---|---|---|---|
| 1 | 0 | 1 | 1 | [{"attr":"ATTR1","pct":49.0,"val":0.26666666666666661},{"attr":"ATTR2","p... |
| 2 | 1 | 1 | 0.8 | [{"attr":"ATTR1","pct":36.0,"val":0.26666666666666661},{"attr":"ATTR3","p... |
| 3 | 2 | 1 | 1 | [{"attr":"ATTR1","pct":49.0,"val":0.26666666666666661},{"attr":"ATTR2","p... |
| 4 | 3 | 1 | 1 | [{"attr":"ATTR1","pct":49.0,"val":0.26666666666666661},{"attr":"ATTR2","p... |
| 5 | 4 | 2 | 1 | [{"attr":"ATTR2","pct":33.0,"val":0.19333333333333333},{"attr":"ATTR3","p... |
| 6 | 5 | 3 | 0.8 | [{"attr":"ATTR1","pct":33.0,"val":0.21666666666666665},{"attr":"ATTR2","p... |

## _SYS_AFL. PAL_UNIFIED_CLASSIFICATION_SCORE_PIVOT

This function makes calculated predictions, and provides the evaluation statistics and metrics for the trained model.

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA Table>` | IN |
| 2 | `<MODEL Table>` | IN |
| 3 | `<PARAMETER Table>` | IN |
| 4 | `<RESULT Table>` | OUT |
| 5 | `<STATISTIC Table>` | OUT |
| 6 | `<CONFUSION_MATRIX Table>` | OUT |
| 7 | `<METRICS Table>` | OUT |

### Input Table

| Table | Column | Data Type | Description |
|---|---|---|---|
| DATA | 1st column | INTEGER, VRACHAR, or NVARCHAR | Record ID. |
| | 2nd column | VARCHAR, NVARCHAR | Variable name. |
| | 3rd column | VARCHAR, NVARCHAR | Variable value. |

### Model Table

| Table | Column | Data Type |
|---|---|---|
| MODEL | 1st column | INTEGER |

| Table | Column | Data Type |
|---|---|---|
| | 2nd column | INTEGER |
| | 3rd column | NCLOB |

### Parameter Table

See more details in Parameter Tables [page 1681].

### Output Table

| Table | Column | Column Name | Column Data Type | Description |
|---|---|---|---|---|
| RESULT | 1st column | ID column name | ID column data type | |
| | 2nd column | SCORE | NVARCHAR (256) | |
| | 3rd column | CONFIDENCE | DOUBLE | |
| | 4th column | REASON CODE | NCLOB | Only valid for random decision trees, decision trees, and hybrid gradient boosting trees. |
| STATISTICS | 1st column | STAT_NAME | NVARCHAR (256) | |
| | 2nd column | STAT_VALUE | NVARCHAR (1000) | |
| | 3rd column | CLASS_NAME | NVARCHAR (256) | |
| CONFUSION_MATRIX | 1st column | ACTUAL_CLASS | NVARCHAR (1000) | |
| | 2nd column | PREDICTED_CLASS | NVARCHAR (1000) | |
| | 3rd column | COUNT | INTEGER | |
| METRICS | 1st column | NAME | NVARCHAR (256) | |
| | 2nd column | X | DOUBLE | |
| | 3rd column | Y | DOUBLE | |

### Example: Random Decision Trees scoring procedure

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_SCORING_DATA_TBL;
CREATE COLUMN TABLE PAL_SCORING_DATA_TBL (
    DATA_ID INTEGER,
    VARIABLE_NAME NVARCHAR(100),
    VAR_VALUE NVARCHAR(100)
);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(0,'ATTR1','1');
INSERT INTO PAL_SCORING_DATA_TBL VALUES(0,'ATTR2','10');
INSERT INTO PAL_SCORING_DATA_TBL VALUES(0,'ATTR3','100');
INSERT INTO PAL_SCORING_DATA_TBL VALUES(0,'ATTR4','A');
INSERT INTO PAL_SCORING_DATA_TBL VALUES(0,'LABEL','1');
```

```
INSERT INTO PAL_SCORING_DATA_TBL VALUES(1,'ATTR1',1.1);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(1,'ATTR2',10.1);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(1,'ATTR3',100.1);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(1,'ATTR4','A');
INSERT INTO PAL_SCORING_DATA_TBL VALUES(1,'LABEL','1');
INSERT INTO PAL_SCORING_DATA_TBL VALUES(2,'ATTR1',1.2);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(2,'ATTR2',10.2);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(2,'ATTR3',100);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(2,'ATTR4','A');
INSERT INTO PAL_SCORING_DATA_TBL VALUES(2,'LABEL','1');
INSERT INTO PAL_SCORING_DATA_TBL VALUES(3,'ATTR1',1.3);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(3,'ATTR2',10.4);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(3,'ATTR3',100);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(3,'ATTR4','A');
INSERT INTO PAL_SCORING_DATA_TBL VALUES(3,'LABEL','1');
INSERT INTO PAL_SCORING_DATA_TBL VALUES(4,'ATTR1',4);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(4,'ATTR2',40);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(4,'ATTR3',400);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(4,'ATTR4','AB');
INSERT INTO PAL_SCORING_DATA_TBL VALUES(4,'LABEL','2');
INSERT INTO PAL_SCORING_DATA_TBL VALUES(5,'ATTR1',9.2);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(5,'ATTR2',90.2);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(5,'ATTR3',900);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(5,'ATTR4','B');
INSERT INTO PAL_SCORING_DATA_TBL VALUES(5,'LABEL','3');
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (100)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MAX_RESULT_NUM', 2, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('NTILES', 2, NULL, NULL);
CALL _SYS_AFL.PAL_UNIFIED_CLASSIFICATION_SCORE_PIVOT(PAL_SCORING_DATA_TBL,
PAL_MODEL_RDT_TBL, #PAL_PARAMETER_TBL, ?, ?,?,?);
```

### Expected Result

RESULT:

| | DATA_ID | SCORE | CONFIDENCE | REASON_CODE |
|---|---|---|---|---|
| 1 | 0 | 1 | 1 | [{"attr":"ATTR1","pct":49.0,"val":0.26666666666666661},{"attr":"ATTR2","p... |
| 2 | 1 | 1 | 0.8 | [{"attr":"ATTR1","pct":36.0,"val":0.26666666666666661},{"attr":"ATTR3","p... |

STATISTICS:

| | |
|---|---|
| STAT_NAME | STAT_VALUE |
| PARAM_NAME | PARAM_VALUE |

SYSTEM_RECALL_PRECISION_41_SCORE_SUPPORT_RECALL_

CONFUSION MATRIX:

| PREDICTED_CLASS | 14 | 20 | 30 | 40 | 52 | 60 | 70 | 80 | 93 |
|---|---|---|---|---|---|---|---|---|---|

METRICS:

| | NAME | X | Y |
|---|---|---|---|
| 1 | RANDOM_CUMGAINS | 0 | 0 |
| 2 | RANDOM_CUMGAINS | 1 | 1 |
| 3 | RANDOM_LIFT | 0 | 1 |
| 4 | RANDOM_LIFT | 1 | 1 |
| 5 | RANDOM_CUMLIFT | 0 | 1 |
| 6 | RANDOM_CUMLIFT | 1 | 1 |
| 7 | PERF_CUMGAINS | 0 | 0 |
| 8 | PERF_CUMGAINS | 0.5 | 1 |
| 9 | PERF_CUMGAINS | 1 | 1 |
| 10 | PERF_LIFT | 0 | 3 |
| 11 | PERF_LIFT | 0.5 | 0 |
| 12 | PERF_LIFT | 1 | 0 |
| 13 | PERF_CUMLIFT | 0 | 3 |
| 14 | PERF_CUMLIFT | 0.5 | 2 |
| 15 | PERF_CUMLIFT | 1 | 1 |
| 16 | ROC_FPR | 0 | 0 |
| 17 | ROC_TPR | 0 | 0 |
| 18 | ROC_FPR | 1 | 0 |
| 19 | ROC_TPR | 1 | 0.66... |
| 20 | ROC_FPR | 2 | 0 |
| 21 | ROC_TPR | 2 | 1 |
| 22 | ROC_FPR | 3 | 0.16... |
| 23 | ROC_TPR | 3 | 1 |
| 24 | ROC_FPR | 4 | 1 |
| 25 | ROC_TPR | 4 | 1 |
| 26 | CUMGAINS | 0 | 0 |
| 27 | CUMGAINS | 0.5 | 1 |
| 28 | CUMGAINS | 1 | 1 |
| 29 | LIFT | 0 | 3 |
| 30 | LIFT | 0.5 | 0 |
| 31 | LIFT | 1 | 0 |
| 32 | CUMLIFT | 0 | 3 |
| 33 | CUMLIFT | 0.5 | 2 |
| 34 | CUMLIFT | 1 | 1 |

## 5.13.2.1 Parameters Specific to Pivoted Unified Classification

The pivoted unified classification function supports almost all the parameters in the original classification function only except some of them disabled and some of them renamed. Besides, the unified classification function has some unique parameters that work in it only.

This topic lists all the parameters renamed or disabled from the original classification parameters and parameters unique to the unified classification function only. Refer to Parameter Tables [page 1681] for a complete list of parameters.

## _SYS_AFL.PAL_UNIFIED_CLASSIFICATION_PIVOT

Mandatory parameters

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| FUNCTION | VARCHAR or NVARCHAR | No default value | The parameter must be the same with the FUNCTION used to train the model.<br>• DT: decision trees<br>• HGBT: hybrid gradient boosting tree<br>• LOGR: logistic regression<br>• M_LOGR: multi-class logistic regression<br>• MLP: multilayer perception<br>• NB: naïve bayes<br>• RDT: random decision trees<br>• SVM: support vector machine |

Optional parameters

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| PARTITION_METHOD | INTEGER | 0 | Defines the way to divide the dataset:<br>• 0: no partition<br>• 1: user defined partition<br>• 2: stratified partition | |
| PARTITION_RANDOM_SEED | INTEGER | 0 | Indicates the seed used to initialize the random number generator:<br>• 0: uses the system time<br>• Not 0: uses the specified seed | Valid only when PARTITION_METHOD is set to 2. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| PARTITION_STRATIFIED_VARIABLE | VARCHAR | No default value | Indicates the column used for stratification. | Valid only when PARTITION_METHOD is set to 2. |
| PARTITION_TRAINING_PERCENT | DOUBLE | 0.8 | The percentage of data used for training.<br>Value: numbers between 0 and 1. | Valid only when PARTITION_METHOD is set to 2. |
| PARTITION_TRAINING_SIZE | INTEGER | No default value | Row size of data used for training.<br>Value: numbers larger or equal to 0. | If both TRAINING_PERCENT and TRAINING_SIZE are specified, TRAINING_PERCENT takes precedence. |
| OUTPUT_PARTITION_RESULT | INTEGER | 0 | Controls whether to output the partition result.<br>• 0: No<br>• 1: Yes | Valid only when PARTITION_METHOD is 1 or 2 and KEY is 1. |
| NTILES | INTEGER | 1 or 20 | Used to control the population tiles in metrics output.<br>The validation value should be at least 1 and no larger than the row size of the validation data.<br>**Note**<br>For AUC, this parameter means the maximum tiles. | If the row size of data for metrics evaluation is less than 20, the default value is 1; otherwise it is 20. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| BACKGROUND_SIZE | INTEGER | 0 | The row size of background data. | It should not be larger than the row size of train data.<br>Valid for the functions:<br>• Naive Bayes<br>• Support Vector Machine<br>• Multilayer Perceptron<br>• Multi-class Logistic Regression |
| BACKGROUND_SAMPLING_SEED | INTEGER | 0 | Specifies the seed for random number generator in the background samping.<br>• 0: Uses the current time (in second) as seed<br>• Others: Uses the specified value as seed | Valid for the functions:<br>• Naive Bayes<br>• Support Vector Machine<br>• Multilayer Perceptron<br>• Multi-class Logistic Regression |
| HANDLE_MISSING_VALUE | INTEGER | 0 | Whether to handle missing values:<br>• 0: No<br>• 1: Yes | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| IMPUTATION_TYPE | INTEGER | 1 | Chooses the overall default imputation type for all columns:<br>• 0: NONE<br>• 1: MODE_MEAN<br>• 2: MODE_MEDIAN<br>• 3: MODE_ALL-ZERO<br>• 4: MODE_ALS<br>• 5: DELETE<br>NONE and DELETE: set all columns to imputation type NONE/DELETE.<br>MODE_MEAN, MODE_MEDIAN, MODE_ALLZERO, and MODE_ALS: set categorical columns to the first imputation type (the type before “”); set numerical columns to the second imputation type (the type after “”). ALLZERO means to fill all missing values in numerical columns with 0. | Only valid when HANDLE_MISSING_VALUE is 1. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| `<column name>`_IMPUTATION_TYPE | INTEGER | No default value | Sets imputation type of a specified column. This value overwrites the default imputation type deduced by the IMPUTATION_TYPE parameter:<br>• 0: NONE<br>• 1: DELETE<br>• 100: MODE<br>• 101: SPECIFIED_CATEGORICAL_VALUE<br>• 200: MEAN<br>• 201: MEDIAN<br>• 203:SPECIFIED_NUMERICAL_VALUE<br>• 204: ALS<br>For type 101 (SPECIFIED_CATEGORICAL_VALUE), use the following syntax to specify a categorical value to be filled in a column V0:<br>INSERT INTO #PAL_PARAMETER_TBL VALUES('V0_IMPUTATION_TYPE', 101, NULL, '`<CATEGORICAL_VALUE>`');<br>For type 203 (SPECIFIED_NUMERICAL_VALUE), use the following syntax to specify a numerical value to be filled in a column V1:<br>INSERT INTO #PAL_PARAMETER_TBL VALUES('V1_IMPUTA- | Only valid when HANDLE_MISSING_VALUE is 1. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| | | | TION_TYPE', 203, `<NUMERICAL_VALUE>`, NULL);<br><br>**Note**<br>A column of integer type can be set to type 101 (SPECIFIED_CATEGORICAL_VALUE) or type 203 (SPECIFIED_NUMERICAL_VALUE):<br>• when setting to type 101 (SPECIFIED_CATEGORICAL_VALUE), the string must contain a valid integer value, otherwise an error will be thrown;<br>• when setting to type 203 (SPECIFIED_NUMERICAL_VALUE), the double value will be converted to integer. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| ALS_FACTOR_NUMBER | INTEGER | 3 | Length of factor vectors, that is, the f in ALS model.<br>**Note**<br>ALS_FACTOR_NUMBER should be less than the number of numerical columns to get a meaningful result. | |
| ALS_REGULARIZATION | DOUBLE | 1e-2 | L2 regularization of the factors, that is, the λ in ALS model. | |
| ALS_MAX_ITERATION | INTEGER | 20 | Maximum number of iterations when training ALS model. | |
| ALS_SEED | INTEGER | 0 | Specifies the seed of the random number generator used in the training of ALS model:<br>• 0: Uses the current time as the seed<br>• Others: Uses the specified value as the seed. | |
| ALS_EXIT_THRESHOLD | DOUBLE | 0.0 | The training of ALS model exits if the value of cost function is not decreased more than the value of ALS_EXIT_THRESHOLD since the last check.<br>If ALS_EXIT_THRESHOLD is set to 0, there is no check and the training only exits on reaching the maximum number of iterations. Evaluations of cost function require additional calculations. You can set this parameter to 0 to avoid it. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| ALS_EXIT_INTERVAL | INTEGER | 5 | During the training of ALS model, it will calculate cost function and check the exit criteria every `<ALS_EXIT_INTERVAL>` iterations. Evaluations of cost function require additional calculations. | |
| ALS_LINEAR_SYSTEM_SOLVER | INTEGER | 0 | • 0: cholesky solver<br>• 1: cg solver (recommended when ALS_FACTOR_NUMBER is large) | |
| ALS_CG_MAX_ITERATION | INTEGER | 3 | Maximum number of iteration of cg solver. | |
| ALS_CENTERING | INTEGER | 1 | Whether to center the data by column before training ALS model.<br>• 0: no centering<br>• 1: centering. | |
| ALS_SCALING | INTEGER | 1 | Whether to scale the data by column before training ALS model.<br>• 0: no scaling<br>• 1: scaling | |

### Decision Trees

| ORIGINAL PARAMETERS | RENAMED/DISABLED PARAMETERS |
|---|---|
| ALLOW_MISSING_DEPENDENT | ALLOW_MISSING_LABEL |
| MIN_RECORDS_OF_PARENT | MIN_RECORDS_PARENT |
| MIN_RECORDS_OF_PARENT_VALUES | MIN_RECORDS_PARENT_VALUES |
| MIN_RECORDS_OF_PARENT_RANGE | MIN_RECORDS_PARENT_RANGE |
| MIN_RECORDS_OF_LEAF | MIN_RECORDS_LEAF |
| MIN_RECORDS_OF_LEAF_VALUES | MIN_RECORDS_LEAF_VALUES |
| MIN_RECORDS_OF_LEAF_RANGE | MIN_RECORDS_LEAF_RANGE |
| IS_OUTPUT_RULES | Disabled |
| IS_OUTPUT_CONFUSION_MATRIX | Disabled |

## HYBRID GRADIENT BOOSTING TREE

| ORIGINAL PARAMETERS | RENAMED/DISABLED PARAMETERS |
|---|---|
| COL_SAMPLE_RATE_BYSPLIT | COL_SUBSAMPLE_SPLIT |
| COL_SAMPLE_RATE_BYSPLIT_RANGE | COL_SUBSAMPLE_SPLIT_RANGE |
| COL_SUBSAMPLE_SPLIT_VALUES | COL_SUBSAMPLE_SPLIT_VALUES |
| COL_SAMPLE_RATE_BYTREE | COL_SUBSAMPLE_TREE |
| COL_SAMPLE_RATE_BYTREE_RANGE | COL_SUBSAMPLE_TREE_RANGE |
| COL_SAMPLE_RATE_BYTREE_VALUES | COL_SUBSAMPLE_TREE_VALUES |
| ETA | LEARNING_RATE |
| ETA_RANGE | LEARNING_RATE_RANGE |
| ETA_VALUES | LEARNING_RATE_VALUES |
| GAMMA | SPLIT_THRESHOLD |
| GAMMA_RANGE | SPLIT_THRESHOLD_RANGE |
| GAMMA_VALUES | SPLIT_THRESHOLD_VALUES |
| ITER_NUM | N_ESTIMATORS |
| ITER_NUM_RANGE | N_ESTIMATORS_RANGE |
| ITER_NUM_VALUES | N_ESTIMATORS_VALUES |
| MIN_CHILD_HESSIAN | MIN_SAMPLES_WEIGHT_LEAF |
| MIN_CHILD_HESSIAN_RANGE | MIN_SAMPLES_WEIGHT_LEAF_RANGE |
| MIN_CHILD_HESSIAN_VALUES | MIN_SAMPLES_WEIGHT_LEAF_VALUES |
| NODE_SIZE | MIN_SAMPLES_LEAF |
| NODE_SIZE_RANGE | MIN_SAMPLES_LEAF_RANGE |
| NODE_SIZE_VALUES | MIN_SAMPLES_LEAF_VALUES |
| NODE_WEIGHT_CONSTRAINT | MAX_W_IN_SPLIT |
| NODE_WEIGHT_CONSTRAINT_RANGE | MAX_W_IN_SPLIT_RANGE |
| NODE_WEIGHT_CONSTRAINT_VALUES | MAX_W_IN_SPLIT_VALUES |
| ROW_SAMPLE_RATE | SUBSAMPLE |
| ROW_SAMPLE_RATE_RANGE | SUBSAMPLE_RANGE |
| ROW_SAMPLE_RATE_VALUES | SUBSAMPLE_VALUES |
| SKETCH_EPS | SKETCH_ESP |
| CALCULATE_IMPORTANCE | Disabled |
| CALCULATE_CONFUSION_MATRIX | Disabled |

### MULTI-CLASS LOGISTIC REGRESSION

| ORIGINAL PARAMETERS | RENAMED/DISABLED PARAMETERS |
|---|---|
| ENET_ALPHA | ALPHA |
| ENET_ALPHA_RANGE | ALPHA_RANGE |

| ORIGINAL PARAMETERS | RENAMED/DISABLED PARAMETERS |
|---|---|
| ENET_ALPHA_VALUES | ALPHA_VALUES |
| ENET_LAMBDA | LAMB |
| ENET_LAMBDA_RANGE | LAMB_RANGE |
| ENET_LAMBDA_VALUES | LAMB_VALUES |
| EXIT_THRESHOLD | TOL |
| MAX_ITERATION | MAX_ITER |
| PMML_EXPORT | Disabled |

## LOGISTIC REGRESSION

| ORIGINAL PARAMETERS | RENAMED/DISABLED PARAMETERS |
|---|---|
| ENET_ALPHA | ALPHA |
| ENET_ALPHA_RANGE | ALPHA_RANGE |
| ENET_ALPHA_VALUES | ALPHA_VALUES |
| ENET_LAMBDA | LAMB |
| ENET_LAMBDA_RANGE | LAMB_RANGE |
| ENET_LAMBDA_VALUES | LAMB_VALUES |
| EXIT_THRESHOLD | TOL |
| MAX_ITERATION | MAX_ITER |
| PMML_EXPORT | Disabled |

## MULTILAYER PERCEPTRON

| ORIGINAL PARAMETERS | RENAMED/DISABLED PARAMETERS |
|---|---|
| HIDDEN_LAYER_ACTIVE_FUNC | ACTIVATION |
| HIDDEN_LAYER_ACTIVE_FUNC_VALUES | ACTIVATION_OPTIONS |
| HIDDEN_LAYER_SIZE_VALUES | HIDDEN_LAYER_SIZE_OPTIONS |
| MAX_ITERATION | MAX_ITER |
| MINI_BATCH_SIZE | BATCH_SIZE |
| MINI_BATCH_SIZE_VALUES | BATCH_SIZE_VALUES |
| MINI_BATCH_SIZE_RANGE | BATCH_SIZE_RANGE |
| MOMENTUM_FACTOR | MOMENTUM |
| MOMENTUM_FACTOR_VALUES | MOMENTUM_VALUES |
| MOMENTUM_FACTOR_RANGE | MOMENTUM_RANGE |
| OUTPUT_LAYER_ACTIVE_FUNC | OUTPUT_ACTIVATION |
| OUTPUT_LAYER_ACTIVE_FUNC_VALUES | OUTPUT_ACTIVATION_OPTIONS |

| ORIGINAL PARAMETERS | RENAMED/DISABLED PARAMETERS |
| --- | --- |
| FUNCTIONALITY | Disabled |

### NAÏVE BAYES

| ORIGINAL PARAMETERS | RENAMED PARAMETERS |
| --- | --- |
| LAPLACE | ALPHA |
| LAPLACE_RANGE | ALPHA_RANGE |
| LAPLACE_VALUES | ALPHA_VALUES |

### RANDOM DECISION TREES

| ORIGINAL PARAMETERS | RENAMED/DISABLED PARAMETERS |
| --- | --- |
| ALLOW_MISSING_DEPENDENT | ALLOW_MISSING_LABEL |
| NODE_SIZE | MIN_SAMPLES_LEAF |
| TREES_NUM | N_ESTIMATORS |
| TRY_NUM | MAX_FEATURES |
| CALCULATE_OOB | Disabled |

### SUPPORT VECTOR MACHINE

| ORIGINAL PARAMETERS | RENAMED/DISABLED PARAMETERS |
| --- | --- |
| ERROR_TOL | TOL |
| TYPE | Disabled |

## _SYS_AFL. PAL_UNIFIED_CLASSIFICATION_PREDICT_PIVOT

The pivoted unified classification prediction procedure supports all the parameters in the prediction procedure of the original PAL functions except the VERBOSE_OUTPUT parameter in some functions is renamed to VERBOSE.

Mandatory parameters

None

Optional Parameters

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| FEATURE_ATTRIBUTION_METHOD | INTEGER | 2 | Controls which method to use in model reasoning:<br>• 0: no reasoning<br>• 1: SAABAS<br>• 2: SHAP | Valid only for these functions to generate the reason code:<br>• Decision Trees<br>• Random Decision Trees<br>• Hybrid Gradient Boosting Tree |
| TOP_K_ATTRIBUTIONS | INTEGER | 10 | Outputs the attributions of top k features which contribute the most. | |
| SAMPLESIZE | INTEGER | 0 | Number of sampled combinations of features. If it equals to 0, it is determined by algorithm heuristically. | It is better to user a number which is larger than the number of the columns of the data table.<br>Available for the following functions:<br>• Naive Bayes<br>• Support Vector Machine<br>• Multilayer Perceptron<br>• Multi-class Logistic Regression |
| SEED | INTEGER | 0 | Specifies the seed for random number generator.<br>• 0: Uses the current time (in second) as seed<br>• Others: Uses the specified value as seed | Available for the following functions:<br>• Naive Bayes<br>• Support Vector Machine<br>• Multilayer Perceptron<br>• Multi-class Logistic Regression |
| HANDLE_MISSING_VALUE | INTEGER | 0 | Whether to handle missing values:<br>• 0: No<br>• 1: Yes | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| IMPUTATION_TYPE | INTEGER | 1 | Chooses the overall default imputation type for all columns:<br>• 0: NONE<br>• 1: MODE_MEAN<br>• 2: MODE_MEDIAN<br>• 3: MODE_ALL-ZERO<br>• 4: MODE_ALS<br>• 5: DELETE<br>NONE and DELETE: set all columns to imputation type NONE/DELETE.<br>MODE_MEAN, MODE_MEDIAN, MODE_ALLZERO, and MODE_ALS: set categorical columns to the first imputation type (the type before “”); set numerical columns to the second imputation type (the type after “”). ALLZERO means to fill all missing values in numerical columns with 0. | Only valid when HANDLE_MISSING_VALUE is 1. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| `<column name>`_IMPUTATION_TYPE | INTEGER | No default value | Sets imputation type of a specified column. This value overwrites the default imputation type deduced by the IMPUTATION_TYPE parameter:<br>• 0: NONE<br>• 1: DELETE<br>• 100: MODE<br>• 101: SPECIFIED_CATEGORICAL_VALUE<br>• 200: MEAN<br>• 201: MEDIAN<br>• 203:SPECIFIED_NUMERICAL_VALUE<br>• 204: ALS<br>For type 101 (SPECIFIED_CATEGORICAL_VALUE), use the following syntax to specify a categorical value to be filled in a column V0:<br>INSERT INTO #PAL_PARAMETER_TBL VALUES('V0_IMPUTATION_TYPE', 101, NULL, '`<CATEGORICAL_VALUE>`');<br>For type 203 (SPECIFIED_NUMERICAL_VALUE), use the following syntax to specify a numerical value to be filled in a column V1:<br>INSERT INTO #PAL_PARAMETER_TBL VALUES('V1_IMPUTA- | Only valid when HANDLE_MISSING_VALUE is 1. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| | | | TION_TYPE', 203, `<NUMERICAL_VALUE>`, NULL);<br><br>**Note**<br>A column of integer type can be set to type 101 (SPECIFIED_CATEGORICAL_VALUE) or type 203 (SPECIFIED_NUMERICAL_VALUE):<br>• when setting to type 101 (SPECIFIED_CATEGORICAL_VALUE), the string must contain a valid integer value, otherwise an error will be thrown;<br>• when setting to type 203 (SPECIFIED_NUMERICAL_VALUE), the double value will be converted to integer. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| ALS_FACTOR_NUMBER | INTEGER | 3 | Length of factor vectors, that is, the f in ALS model.<br>**Note**<br>ALS_FACTOR_NUMBER should be less than the number of numerical columns to get a meaningful result. | |
| ALS_REGULARIZATION | DOUBLE | 1e-2 | L2 regularization of the factors, that is, the λ in ALS model. | |
| ALS_MAX_ITERATION | INTEGER | 20 | Maximum number of iterations when training ALS model. | |
| ALS_SEED | INTEGER | 0 | Specifies the seed of the random number generator used in the training of ALS model:<br>• 0: Uses the current time as the seed<br>• Others: Uses the specified value as the seed. | |
| ALS_EXIT_THRESHOLD | DOUBLE | 0.0 | The training of ALS model exits if the value of cost function is not decreased more than the value of ALS_EXIT_THRESHOLD since the last check.<br>If ALS_EXIT_THRESHOLD is set to 0, there is no check and the training only exits on reaching the maximum number of iterations. Evaluations of cost function require additional calculations. You can set this parameter to 0 to avoid it. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| ALS_EXIT_INTERVAL | INTEGER | 5 | During the training of ALS model, it will calculate cost function and check the exit criteria every `<ALS_EXIT_INTERVAL>` iterations. Evaluations of cost function require additional calculations. | |
| ALS_LINEAR_SYSTEM_SOLVER | INTEGER | 0 | • 0: cholesky solver<br>• 1: cg solver (recommended when ALS_FACTOR_NUMBER is large) | |
| ALS_CG_MAX_ITERATION | INTEGER | 3 | Maximum number of iteration of cg solver. | |
| ALS_CENTERING | INTEGER | 1 | Whether to center the data by column before training ALS model.<br>• 0: no centering<br>• 1: centering. | |
| ALS_SCALING | INTEGER | 1 | Whether to scale the data by column before training ALS model.<br>• 0: no scaling<br>• 1: scaling | |

## _SYS_AFL. PAL_UNIFIED_CLASSIFICATION_SCORE_PIVOT

The pivoted unified classification score procedure supports all the parameters in the prediction procedures of the original PAL functions except the VERBOSE and VERBOSE_OUTPUT parameters.

Mandatory parameters

None

Optional Parameters

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| MAX_RESULT_NUM | INTEGER | No default value | Specifies the output number of prediction results. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| NTILES | INTEGER | 1 or 20 | Used to control the population tiles in the metrics output.<br>The value should be at least 1 and no larger than the row size of the input data.<br>**Note**<br>For AUC, this parameter means the maximum tiles. | If the row size of data for metrics evaluation is less than 20, the default value is the 1; otherwise it is 20. |
| FEATURE_ATTRIBUTION_METHOD | INTEGER | 2 | Controls which method to use in model reasoning:<br>• 0: no reasoning<br>• 1: SAABAS<br>• 2: SHAP | Valid only for these functions to generate the reason code:<br>• Decision Trees<br>• Random Decision Trees<br>• Hybrid Gradient Boosting Tree |
| TOP_K_ATTRIBUTIONS | INTEGER | 10 | Outputs the attributions of top k features which contribute the most. | |
| SAMPLESIZE | INTEGER | 0 | Number of sampled combinations of features. If it equals to 0, it is determined by algorithm heuristically. | It is better to user a number which is larger than the number of the columns of the data table.<br>Available for the following functions:<br>• Naive Bayes<br>• Support Vector Machine<br>• Multilayer Perceptron<br>• Multi-class Logistic Regression |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| SEED | INTEGER | 0 | Specifies the seed for random number generator.<br>• 0: Uses the current time (in second) as seed<br>• Others: Uses the specified value as seed | Available for the following functions:<br>• Naive Bayes<br>• Support Vector Machine<br>• Multilayer Perceptron<br>• Multi-class Logistic Regression |
| HANDLE_MISSING_VALUE | INTEGER | 0 | Whether to handle missing values:<br>• 0: No<br>• 1: Yes | |
| IMPUTATION_TYPE | INTEGER | 1 | Chooses the overall default imputation type for all columns:<br>• 0: NONE<br>• 1: MODE_MEAN<br>• 2: MODE_MEDIAN<br>• 3: MODE_ALLZERO<br>• 4: MODE_ALS<br>• 5: DELETE<br>NONE and DELETE: set all columns to imputation type NONE/DELETE.<br>MODE_MEAN, MODE_MEDIAN, MODE_ALLZERO, and MODE_ALS: set categorical columns to the first imputation type (the type before “”); set numerical columns to the second imputation type (the type after “”). ALLZERO means to fill all missing values in numerical columns with 0. | Only valid when HANDLE_MISSING_VALUE is 1. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| `<column name>`_IMPUTATION_TYPE | INTEGER | No default value | Sets imputation type of a specified column. This value overwrites the default imputation type deduced by the IMPUTATION_TYPE parameter:<br>• 0: NONE<br>• 1: DELETE<br>• 100: MODE<br>• 101: SPECIFIED_CATEGORICAL_VALUE<br>• 200: MEAN<br>• 201: MEDIAN<br>• 203:SPECIFIED_NUMERICAL_VALUE<br>• 204: ALS<br>For type 101 (SPECIFIED_CATEGORICAL_VALUE), use the following syntax to specify a categorical value to be filled in a column V0:<br>INSERT INTO #PAL_PARAMETER_TBL VALUES('V0_IMPUTATION_TYPE', 101, NULL, '`<CATEGORICAL_VALUE>`');<br>For type 203 (SPECIFIED_NUMERICAL_VALUE), use the following syntax to specify a numerical value to be filled in a column V1:<br>INSERT INTO #PAL_PARAMETER_TBL VALUES('V1_IMPUTA- | Only valid when HANDLE_MISSING_VALUE is 1. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| | | | TION_TYPE', 203, `<NUMERICAL_VALUE>`, NULL);<br><br>**Note**<br>A column of integer type can be set to type 101 (SPECIFIED_CATEGORICAL_VALUE) or type 203 (SPECIFIED_NUMERICAL_VALUE):<br>• when setting to type 101 (SPECIFIED_CATEGORICAL_VALUE), the string must contain a valid integer value, otherwise an error will be thrown;<br>• when setting to type 203 (SPECIFIED_NUMERICAL_VALUE), the double value will be converted to integer. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| ALS_FACTOR_NUMBER | INTEGER | 3 | Length of factor vectors, that is, the f in ALS model.<br>**Note**<br>ALS_FACTOR_NUMBER should be less than the number of numerical columns to get a meaningful result. | |
| ALS_REGULARIZATION | DOUBLE | 1e-2 | L2 regularization of the factors, that is, the λ in ALS model. | |
| ALS_MAX_ITERATION | INTEGER | 20 | Maximum number of iterations when training ALS model. | |
| ALS_SEED | INTEGER | 0 | Specifies the seed of the random number generator used in the training of ALS model:<br>• 0: Uses the current time as the seed<br>• Others: Uses the specified value as the seed. | |
| ALS_EXIT_THRESHOLD | DOUBLE | 0.0 | The training of ALS model exits if the value of cost function is not decreased more than the value of ALS_EXIT_THRESHOLD since the last check.<br>If ALS_EXIT_THRESHOLD is set to 0, there is no check and the training only exits on reaching the maximum number of iterations. Evaluations of cost function require additional calculations. You can set this parameter to 0 to avoid it. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| ALS_EXIT_INTERVAL | INTEGER | 5 | During the training of ALS model, it will calculate cost function and check the exit criteria every `<ALS_EXIT_INTERVAL>` iterations. Evaluations of cost function require additional calculations. | |
| ALS_LINEAR_SYSTEM_SOLVER | INTEGER | 0 | • 0: cholesky solver<br>• 1: cg solver (recommended when ALS_FACTOR_NUMBER is large) | |
| ALS_CG_MAX_ITERATION | INTEGER | 3 | Maximum number of iteration of cg solver. | |
| ALS_CENTERING | INTEGER | 1 | Whether to center the data by column before training ALS model.<br>• 0: no centering<br>• 1: centering. | |
| ALS_SCALING | INTEGER | 1 | Whether to scale the data by column before training ALS model.<br>• 0: no scaling<br>• 1: scaling | |

## 5.13.2.2 Data Partition

A data partition process is provided in the train procedure implementation. The input data can be divided into the train set and the validation set. The train set is used for generating the training model and the validation set is used to evaluate the quality of the model.

You can execute data partition by setting the following parameters.

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| PARTITION_METHOD | INTEGER | 0 | Defines the way to divide the dataset:<br>• 0: no partition<br>• 1: user defined partition<br>• 2: stratified partition | If the parameter is set to be 1, an additional purpose column in data table is needed. |
| PARTITION_RANDOM_SEED | INTEGER | 0 | Indicates the seed used to initialize the random number generator:<br>• 0: Uses the system time<br>• Not 0: Uses the specified seed | Valid only when PARTITION_METHOD is set to 2. |
| PARTITION_STRATIFIED_VARIABLE | VARCHAR | No default value | Indicates which column is used for stratification. | Valid only when PARTITION_METHOD is set to 2. |
| PARTITION_TRAINING_PERCENT | DOUBLE | 0.8 | The percentage of data used for training.<br>Value: 0 - 1 | Valid only when PARTITION_METHOD is set to 2. |
| PARTITION_TRAINING_SIZE | INTEGER | No default value | Row size of data used for training.<br>Value: ≥ 0 | If both TRAINING_PERCENT and TRAINING_SIZE are specified, TRAINING_PERCENT takes precedence. |
| OUTPUT_PARTITION_RESULT | INTEGER | 0 | Controls whether to output the partition result.<br>• 0: No<br>• 1: Yes | Valid only when PARTITION_METHOD is 1 or 2 and KEY is 1. |

## 5.13.2.3 Model Evaluation

Unified classification with pivoted input data provides three output tables as the evaluating results of the training model: statistics table, confusion matrix table, and the metrics table. They are generated in both the train and scoring procedures. In the training procedure, model evaluation works only if the PARTITION_METHOD is 1 or 2.

### Statistics Table

In the statistics table, the following statistics are supported:

- AUC: See Area Under Curve (AUC) [page 174] for more details.
- RECALL, PRECISION, F1-SCORE, SUPPORT: See Confusion Matrix [page 183] for more details.
- ACCURACY: The percentage of samples that are correctly classified.
- KAPPA (Cohen's kappa): Compares an observed accuracy with an expected accuracy given by a random system. KAPPA is calculated by the following equation:

  $$\text{KAPPA} = \frac{p_o - p_e}{1 - p_e}$$

  Where $p_o$ is the observed accuracy and $p_e$ is the expected accuracy. In the binary case, $p_o$ and $p_e$ can be calculated by the following equations:
  N = TP+TN+FP+FN

  $$p_o = \frac{TP+TN}{N}$$

  $$p_e = \frac{(TP+FN)*(TP+FP)-(FP+TN)*(FN+TN)}{N^2}$$

- MCC (Matthews Correlation Coefficient): In a binary case, the statistic of MCC is calculated by the following equation:

  $$\text{MCC} = \frac{TP*TN-FP*FN}{\sqrt{(TP+FP)*(TP+FN)*(TN+FP)*(TN+FN)}}$$

  Where TP, TN, FP and FN represent True Positive, True Negative, False Positive and False Negative respectively.

**Confusion Matrix Table**

The confusion matrix table provides the result of confusion matrix calculation based on the validation set. See Confusion Matrix [page 183] for more details.

**Metrics Table**

> **Note**
>
> The metrics table of Support Vector Machine is empty due to the limitations of the SVM algorithm.

In the metrics table, the following metrics are output:

- ROC: See Area Under Curve (AUC) [page 174] for more details.
- LIFT: The result of LIFT is a curve. The curve can be obtained by plotting the interval lift at several threshold of the population. The calculation formulas for interval lift are listed below:

  $$\text{Lift} = \frac{PPV}{Positive\ Rate}$$

  $$\text{PPV} = \frac{TP}{TP+FP}$$

  $$\text{Positive Rate} = \frac{TP+FN}{TP+FP+TN+FN}$$

- CUMULATIVE LIFT: The result of CUMULATIVE LIFT is a curve. The curve can be obtained by plotting the lift at several threshold of the population. The calculation formulas for lift are listed below:

$$\text{CUM_Lift} = \frac{CUM_PPV}{Positive\ Rate}$$

$$\text{CUM_PPV} = \frac{CUM_TP}{CUM_TP+CUM_FP}$$

$$\text{Positive Rate} = \frac{TP+FN}{TP+FP+TN+FN}$$

- CUMULATIVE GAINS: The result of CUMULATIVE GAINS is a curve. The curve can be obtained by plotting the true positive rate (TPR) at several threshold of the population. The calculation formula for TPR is listed below:

$$\text{TPR} = \frac{TP}{TP+FN}$$

## Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

**Example 1: Random Decision Trees Train Procedure**

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_META_DATA_TBL;
CREATE COLUMN TABLE PAL_META_DATA_TBL(
    VARIABLE_NAME NVARCHAR(100),
    VARIABLE_TYPE NVARCHAR(100)
);
INSERT INTO PAL_META_DATA_TBL VALUES ('ATTR1', 'CONTINUOUS');
INSERT INTO PAL_META_DATA_TBL VALUES ('ATTR2', 'CONTINUOUS');
INSERT INTO PAL_META_DATA_TBL VALUES ('ATTR3', 'CATEGORICAL');
INSERT INTO PAL_META_DATA_TBL VALUES ('ATTR4', 'CATEGORICAL');
INSERT INTO PAL_META_DATA_TBL VALUES ('LABEL', 'TARGET');
DROP TABLE PAL_DATA_TBL;
CREATE COLUMN TABLE PAL_DATA_TBL (
    DATA_ID INTEGER,
    VARIABLE_NAME NVARCHAR(100),
    VAR_VALUE NVARCHAR(100),
    PURPOSE  INTEGER
);
INSERT INTO PAL_DATA_TBL VALUES(0,'ATTR1','1',2);
INSERT INTO PAL_DATA_TBL VALUES(0,'ATTR2','10',2);
INSERT INTO PAL_DATA_TBL VALUES(0,'ATTR3','100',2);
INSERT INTO PAL_DATA_TBL VALUES(0,'ATTR4','A',2);
INSERT INTO PAL_DATA_TBL VALUES(0,'LABEL','1',2);
INSERT INTO PAL_DATA_TBL VALUES(1,'ATTR1',1.1,1);
INSERT INTO PAL_DATA_TBL VALUES(1,'ATTR2',10.1,1);
INSERT INTO PAL_DATA_TBL VALUES(1,'ATTR3',100.1,1);
```

```
INSERT INTO PAL_DATA_TBL VALUES(1,'ATTR4','A',1);
INSERT INTO PAL_DATA_TBL VALUES(1,'LABEL',1,1);
INSERT INTO PAL_DATA_TBL VALUES(2,'ATTR1',1.2,1);
INSERT INTO PAL_DATA_TBL VALUES(2,'ATTR2',10.2,1);
INSERT INTO PAL_DATA_TBL VALUES(2,'ATTR3',100,1);
INSERT INTO PAL_DATA_TBL VALUES(2,'ATTR4','A',1);
INSERT INTO PAL_DATA_TBL VALUES(2,'LABEL',1,1);
INSERT INTO PAL_DATA_TBL VALUES(3,'ATTR1',1.3,1);
INSERT INTO PAL_DATA_TBL VALUES(3,'ATTR2',10.4,1);
INSERT INTO PAL_DATA_TBL VALUES(3,'ATTR3',100,1);
INSERT INTO PAL_DATA_TBL VALUES(3,'ATTR4','A',1);
INSERT INTO PAL_DATA_TBL VALUES(3,'LABEL',1,1);
INSERT INTO PAL_DATA_TBL VALUES(4,'ATTR1',1.2,1);
INSERT INTO PAL_DATA_TBL VALUES(4,'ATTR2',10.3,1);
INSERT INTO PAL_DATA_TBL VALUES(4,'ATTR3',100,1);
INSERT INTO PAL_DATA_TBL VALUES(4,'ATTR4','AB',1);
INSERT INTO PAL_DATA_TBL VALUES(4,'LABEL',1,1);
INSERT INTO PAL_DATA_TBL VALUES(5,'ATTR1',4,1);
INSERT INTO PAL_DATA_TBL VALUES(5,'ATTR2',40,1);
INSERT INTO PAL_DATA_TBL VALUES(5,'ATTR3',400,1);
INSERT INTO PAL_DATA_TBL VALUES(5,'ATTR4','AB',1);
INSERT INTO PAL_DATA_TBL VALUES(5,'LABEL',2,1);
INSERT INTO PAL_DATA_TBL VALUES(6,'ATTR1',4.1,2);
INSERT INTO PAL_DATA_TBL VALUES(6,'ATTR2',40,2);
INSERT INTO PAL_DATA_TBL VALUES(6,'ATTR3',400,2);
INSERT INTO PAL_DATA_TBL VALUES(6,'ATTR4','AB',2);
INSERT INTO PAL_DATA_TBL VALUES(6,'LABEL',2,2);
INSERT INTO PAL_DATA_TBL VALUES(7,'ATTR1',4.2,1);
INSERT INTO PAL_DATA_TBL VALUES(7,'ATTR2',40.2,1);
INSERT INTO PAL_DATA_TBL VALUES(7,'ATTR3',400,1);
INSERT INTO PAL_DATA_TBL VALUES(7,'ATTR4','AB',1);
INSERT INTO PAL_DATA_TBL VALUES(7,'LABEL',2,1);
INSERT INTO PAL_DATA_TBL VALUES(8,'ATTR1',4.3,1);
INSERT INTO PAL_DATA_TBL VALUES(8,'ATTR2',40.4,1);
INSERT INTO PAL_DATA_TBL VALUES(8,'ATTR3',400,1);
INSERT INTO PAL_DATA_TBL VALUES(8,'ATTR4','AB',1);
INSERT INTO PAL_DATA_TBL VALUES(8,'LABEL',2,1);
INSERT INTO PAL_DATA_TBL VALUES(9,'ATTR1',4.2,1);
INSERT INTO PAL_DATA_TBL VALUES(9,'ATTR2',40.3,1);
INSERT INTO PAL_DATA_TBL VALUES(9,'ATTR3',400,1);
INSERT INTO PAL_DATA_TBL VALUES(9,'ATTR4','AB',1);
INSERT INTO PAL_DATA_TBL VALUES(9,'LABEL',2,1);
INSERT INTO PAL_DATA_TBL VALUES(10,'ATTR1',9,1);
INSERT INTO PAL_DATA_TBL VALUES(10,'ATTR2',90,1);
INSERT INTO PAL_DATA_TBL VALUES(10,'ATTR3',900,1);
INSERT INTO PAL_DATA_TBL VALUES(10,'ATTR4','B',1);
INSERT INTO PAL_DATA_TBL VALUES(10,'LABEL',3,1);
INSERT INTO PAL_DATA_TBL VALUES(11,'ATTR1',9.1,1);
INSERT INTO PAL_DATA_TBL VALUES(11,'ATTR2',90.1,1);
INSERT INTO PAL_DATA_TBL VALUES(11,'ATTR3',900,1);
INSERT INTO PAL_DATA_TBL VALUES(11,'ATTR4','A',1);
INSERT INTO PAL_DATA_TBL VALUES(11,'LABEL',3,1);
INSERT INTO PAL_DATA_TBL VALUES(12,'ATTR1',9.2,2);
INSERT INTO PAL_DATA_TBL VALUES(12,'ATTR2',90.2,2);
INSERT INTO PAL_DATA_TBL VALUES(12,'ATTR3',900,2);
INSERT INTO PAL_DATA_TBL VALUES(12,'ATTR4','B',2);
INSERT INTO PAL_DATA_TBL VALUES(12,'LABEL',3,2);
INSERT INTO PAL_DATA_TBL VALUES(13,'ATTR1',9.3,1);
INSERT INTO PAL_DATA_TBL VALUES(13,'ATTR2',90.4,1);
INSERT INTO PAL_DATA_TBL VALUES(13,'ATTR3',900,1);
INSERT INTO PAL_DATA_TBL VALUES(13,'ATTR4','A',1);
INSERT INTO PAL_DATA_TBL VALUES(13,'LABEL',3,1);
INSERT INTO PAL_DATA_TBL VALUES(14,'ATTR1',9.2,1);
INSERT INTO PAL_DATA_TBL VALUES(14,'ATTR2',90.3,1);
INSERT INTO PAL_DATA_TBL VALUES(14,'ATTR3',900,1);
INSERT INTO PAL_DATA_TBL VALUES(14,'ATTR4','A',1);
INSERT INTO PAL_DATA_TBL VALUES(14,'LABEL',3,1);
DROP TABLE PAL_PARAMETER_TBL;
```

```
CREATE  COLUMN TABLE PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (100)
);
INSERT INTO PAL_PARAMETER_TBL VALUES ('FUNCTION', NULL, NULL, 'RDT');
INSERT INTO PAL_PARAMETER_TBL VALUES ('SPLIT_THRESHOLD', NULL, 0.0000001, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('MIN_SAMPLES_LEAF', 1, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('N_ESTIMATORS', 10, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('MAX_DEPTH', 55, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('SEED', 2, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('NTILES', 2, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('PARTITION_METHOD', 2, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('PARTITION_RANDOM_SEED', 2, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('PARTITION_STRATIFIED_VARIABLE', NULL,
NULL, 'LABEL');
INSERT INTO PAL_PARAMETER_TBL VALUES ('PARTITION_TRAINING_PERCENT', NULL, 0.7,
NULL);
DROP TABLE PAL_MODEL_RDT_TBL;
CREATE COLUMN TABLE PAL_MODEL_RDT_TBL (
    "ROW_INDEX" INTEGER,
    "PART_INDEX" INTEGER,
    "MODEL_CONTENT" NCLOB
);
DROP TABLE PAL_STAT_TBL;
CREATE COLUMN TABLE PAL_STAT_TBL (
    "STAT_NAME" NVARCHAR(256),
    "STAT_VALUE" NVARCHAR(1000),
    "CLASS_NAME" NVARCHAR(256)
);
DROP TABLE PAL_CMATRIX_TBL;
CREATE COLUMN TABLE PAL_CMATRIX_TBL (
    "ACTUAL_CLASS" NVARCHAR(256),
    "PREDICTED_CLASS" NVARCHAR(256),
    "COUNT" INTEGER
);
DROP TABLE PAL_METRICS_TBL;
CREATE COLUMN TABLE PAL_METRICS_TBL (
    "NAME" NVARCHAR(256),
    "X" DOUBLE,
    "Y" DOUBLE
);
DO
BEGIN
    lt_meta_data = SELECT * FROM PAL_META_DATA_TBL;
    lt_data = SELECT * FROM PAL_DATA_TBL;
    lt_ctrl = SELECT * FROM PAL_PARAMETER_TBL;
    CALL _SYS_AFL.PAL_UNIFIED_CLASSIFICATION_PIVOT
(:lt_meta_data, :lt_data, :lt_ctrl, lt_model, lt_imp,lt_stat,
lt_opt,lt_cm,lt_metrics,lt_partition,lt_ph1);
    INSERT INTO PAL_MODEL_RDT_TBL SELECT * FROM :lt_model;
    INSERT INTO PAL_STAT_TBL SELECT * FROM :lt_stat;
    INSERT INTO PAL_CMATRIX_TBL SELECT * FROM :lt_cm;
    INSERT INTO PAL_METRICS_TBL SELECT * FROM :lt_metrics;
END;
SELECT * FROM PAL_STAT_TBL;
SELECT * FROM PAL_CMATRIX_TBL;
SELECT * FROM PAL_METRICS_TBL;
```

Expected Result

STATISTICS:

| | STAT_NAME | STAT_VALUE | CLASS_NAME |
|---|---|---|---|
| 1 | AUC | 1 | ? |
| 2 | RECALL | 1 | 1 |
| 3 | PRECISION | 1 | 1 |
| 4 | F1_SCORE | 1 | 1 |
| 5 | SUPPORT | 2 | 1 |
| 6 | RECALL | 1 | 2 |
| 7 | PRECISION | 1 | 2 |
| 8 | F1_SCORE | 1 | 2 |
| 9 | SUPPORT | 2 | 2 |
| 10 | RECALL | 1 | 3 |
| 11 | PRECISION | 1 | 3 |
| 12 | F1_SCORE | 1 | 3 |
| 13 | SUPPORT | 1 | 3 |
| 14 | ACCURACY | 1 | ? |
| 15 | KAPPA | 1 | ? |
| 16 | MCC | 1 | ? |

CONFUSION MATRIX:

| | ACTUAL_CLASS | PREDICTED_CLASS | COUNT |
|---|---|---|---|
| 1 | 1 | 1 | 2 |
| 2 | 1 | 2 | 0 |
| 3 | 1 | 3 | 0 |
| 4 | 2 | 1 | 0 |
| 5 | 2 | 2 | 2 |
| 6 | 2 | 3 | 0 |
| 7 | 3 | 1 | 0 |
| 8 | 3 | 2 | 0 |
| 9 | 3 | 3 | 1 |

METRICS:

| | NAME | X | Y |
|---|---|---|---|
| 1 | RANDOM_CUMGAINS | 0 | 0 |
| 2 | RANDOM_CUMGAINS | 1 | 1 |
| 3 | RANDOM_LIFT | 0 | 1 |
| 4 | RANDOM_LIFT | 1 | 1 |
| 5 | RANDOM_CUMLIFT | 0 | 1 |
| 6 | RANDOM_CUMLIFT | 1 | 1 |
| 7 | PERF_CUMGAINS | 0 | 0 |
| 8 | PERF_CUMGAINS | 0.4 | 1 |
| 9 | PERF_CUMGAINS | 1 | 1 |
| 10 | PERF_LIFT | 0 | 3 |
| 11 | PERF_LIFT | 0.4 | 2 |
| 12 | PERF_LIFT | 1 | 0 |
| 13 | PERF_CUMLIFT | 0 | 3 |
| 14 | PERF_CUMLIFT | 0.4 | 2.50... |
| 15 | PERF_CUMLIFT | 1 | 1 |
| 16 | ROC_FPR | 0 | 0 |
| 17 | ROC_TPR | 0 | 0 |
| 18 | ROC_FPR | 1 | 0 |
| 19 | ROC_TPR | 1 | 1 |
| 20 | ROC_FPR | 2 | 1 |
| 21 | ROC_TPR | 2 | 1 |
| 22 | CUMGAINS | 0 | 0 |
| 23 | CUMGAINS | 0.4 | 1 |
| 24 | CUMGAINS | 1 | 1 |
| 25 | LIFT | 0 | 3 |
| 26 | LIFT | 0.4 | 2 |
| 27 | LIFT | 1 | 0 |
| 28 | CUMLIFT | 0 | 3 |
| 29 | CUMLIFT | 0.4 | 2.50... |
| 30 | CUMLIFT | 1 | 1 |

### Example 2: Random Decision Trees Scoring Procedure

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_SCORING_DATA_TBL;
CREATE COLUMN TABLE PAL_SCORING_DATA_TBL (
    DATA_ID INTEGER,
    VARIABLE_NAME NVARCHAR(100),
    VAR_VALUE NVARCHAR(100)
);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(0,'ATTR1','1');
INSERT INTO PAL_SCORING_DATA_TBL VALUES(0,'ATTR2','10');
INSERT INTO PAL_SCORING_DATA_TBL VALUES(0,'ATTR3','100');
INSERT INTO PAL_SCORING_DATA_TBL VALUES(0,'ATTR4','A');
INSERT INTO PAL_SCORING_DATA_TBL VALUES(0,'LABEL','1');
INSERT INTO PAL_SCORING_DATA_TBL VALUES(1,'ATTR1',1.1);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(1,'ATTR2',10.1);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(1,'ATTR3',100.1);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(1,'ATTR4','A');
INSERT INTO PAL_SCORING_DATA_TBL VALUES(1,'LABEL','1');
INSERT INTO PAL_SCORING_DATA_TBL VALUES(2,'ATTR1',1.2);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(2,'ATTR2',10.2);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(2,'ATTR3',100);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(2,'ATTR4','A');
INSERT INTO PAL_SCORING_DATA_TBL VALUES(2,'LABEL','1');
INSERT INTO PAL_SCORING_DATA_TBL VALUES(3,'ATTR1',1.3);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(3,'ATTR2',10.4);
```

```
INSERT INTO PAL_SCORING_DATA_TBL VALUES(3,'ATTR3',100);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(3,'ATTR4','A');
INSERT INTO PAL_SCORING_DATA_TBL VALUES(3,'LABEL','1');
INSERT INTO PAL_SCORING_DATA_TBL VALUES(4,'ATTR1',4);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(4,'ATTR2',40);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(4,'ATTR3',400);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(4,'ATTR4','AB');
INSERT INTO PAL_SCORING_DATA_TBL VALUES(4,'LABEL','2');
INSERT INTO PAL_SCORING_DATA_TBL VALUES(5,'ATTR1',9.2);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(5,'ATTR2',90.2);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(5,'ATTR3',900);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(5,'ATTR4','B');
INSERT INTO PAL_SCORING_DATA_TBL VALUES(5,'LABEL','3');
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (100)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MAX_RESULT_NUM', 2, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('NTILES', 2, NULL, NULL);
CALL _SYS_AFL.PAL_UNIFIED_CLASSIFICATION_SCORE_PIVOT(PAL_SCORING_DATA_TBL,
PAL_MODEL_RDT_TBL, #PAL_PARAMETER_TBL, ?, ?,?,?);
```

Expected Result

STATISTICS:

| | STAT_NAME | STAT_VALUE | CLASS_NAME |
|---|---|---|---|
| 1 | AUC | 1 | ? |
| 2 | RECALL | 1 | 1 |
| 3 | PRECISION | 1 | 1 |
| 4 | F1_SCORE | 1 | 1 |
| 5 | SUPPORT | 4 | 1 |
| 6 | RECALL | 1 | 2 |
| 7 | PRECISION | 1 | 2 |
| 8 | F1_SCORE | 1 | 2 |
| 9 | SUPPORT | 1 | 2 |
| 10 | RECALL | 1 | 3 |
| 11 | PRECISION | 1 | 3 |
| 12 | F1_SCORE | 1 | 3 |
| 13 | SUPPORT | 1 | 3 |
| 14 | ACCURACY | 1 | ? |
| 15 | KAPPA | 1 | ? |
| 16 | MCC | 1 | ? |

CONFUSION MATRIX:

| | ACTUAL_CLASS | PREDICTED_CLASS | COUNT |
|---|---|---|---|
| 1 | 1 | 1 | 4 |
| 2 | 1 | 2 | 0 |
| 3 | 1 | 3 | 0 |
| 4 | 2 | 1 | 0 |
| 5 | 2 | 2 | 1 |
| 6 | 2 | 3 | 0 |
| 7 | 3 | 1 | 0 |
| 8 | 3 | 2 | 0 |
| 9 | 3 | 3 | 1 |

METRICS:

| | NAME | X | Y |
|---|---|---|---|
| 1 | RANDOM_CUMGAINS | 0 | 0 |
| 2 | RANDOM_CUMGAINS | 1 | 1 |
| 3 | RANDOM_LIFT | 0 | 1 |
| 4 | RANDOM_LIFT | 1 | 1 |
| 5 | RANDOM_CUMLIFT | 0 | 1 |
| 6 | RANDOM_CUMLIFT | 1 | 1 |
| 7 | PERF_CUMGAINS | 0 | 0 |
| 8 | PERF_CUMGAINS | 0.5 | 1 |
| 9 | PERF_CUMGAINS | 1 | 1 |
| 10 | PERF_LIFT | 0 | 3 |
| 11 | PERF_LIFT | 0.5 | 0 |
| 12 | PERF_LIFT | 1 | 0 |
| 13 | PERF_CUMLIFT | 0 | 3 |
| 14 | PERF_CUMLIFT | 0.5 | 2 |
| 15 | PERF_CUMLIFT | 1 | 1 |
| 16 | ROC_FPR | 0 | 0 |
| 17 | ROC_TPR | 0 | 0 |
| 18 | ROC_FPR | 1 | 0 |
| 19 | ROC_TPR | 1 | 0.66... |
| 20 | ROC_FPR | 2 | 0 |
| 21 | ROC_TPR | 2 | 1 |
| 22 | ROC_FPR | 3 | 0.16... |
| 23 | ROC_TPR | 3 | 1 |
| 24 | ROC_FPR | 4 | 1 |
| 25 | ROC_TPR | 4 | 1 |
| 26 | CUMGAINS | 0 | 0 |
| 27 | CUMGAINS | 0.5 | 1 |
| 28 | CUMGAINS | 1 | 1 |
| 29 | LIFT | 0 | 3 |
| 30 | LIFT | 0.5 | 0 |
| 31 | LIFT | 1 | 0 |
| 32 | CUMLIFT | 0 | 3 |
| 33 | CUMLIFT | 0.5 | 2 |
| 34 | CUMLIFT | 1 | 1 |

## 5.13.2.4 Local Interpretability of Models

In the process of machine learning predictive modeling, you can also view why the predicative results are made.

In the process of machine learning predictive modeling, you can also view why the predicative results are made.

Based on the Shapley Values in game theory, SHAP (Shapley Additive Explanations) was proposed to explain individual predictions.

When it comes to specified machine learning model, the implementations of SHAP values have variant versions.

- For tree-based machine learning models, a tree SHAP explainer (tree SHAP) is provided. This method can exactly compute the Shapley values in polynomial time and satisfy the interpretation consistency. Additionally, an approximating method, Saabas is also implemented to calculate the attribution values, which takes only the single decision path into account and could run in a logarithmic time (over the number of nodes).
- For nonlinear and non-tree based models, for example, Support Vector Mahcine, a kernel SHAP explainer (kernel SHAP) is provided. Kernel SHAP uses samplings with a kernel to estimate the SHAP values. The samplings called background data are sampled from the train dataset in the trainining process and stored in the model table.

The following table gives an overview of the supported SHAP versions for different functions.

| FUNCTION NAME | METHOD |
|---|---|
| Decision Trees | • tree SHAP<br>• Saabas |
| Random Decision Trees | • tree SHAP<br>• Saabas |
| Hybrid Gradient Boosting Tree | • tree SHAP<br>• Saabas |
| Naive Bayes | kernel SHAP |
| Support Vector Machine | kernel SHAP |
| Multilayer Perceptron | kernel SHAP |

## _SYS_AFL.PAL_UNIFIED_CLASSIFICATION

### Parameter Table

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| BACKGROUND_SIZE | INTEGER | 0 | The row size of background data. | It should not be larger than the row size of train data. Valid for the functions:<br>• Naive Bayes<br>• Support Vector Machine<br>• Multilayer Perceptron |
| BACKGROUND_SAMPLING_SEED | INTEGER | 0 | Specifies the seed for random number generator in the background samping.<br>• 0: Uses the current time (in second) as seed<br>• Others: Uses the specified value as seed | Valid for the functions:<br>• Naive Bayes<br>• Support Vector Machine<br>• Multilayer Perceptron |

### Examples

Assume that:

- DM_PAL is a schema belonging to USER1;
- USER1 has been assigned the AFL__SYS_AFL_AFLPAL_EXECUTE or AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION role.

**Example 1: tree SHAP ( Random Decision Trees )**

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_META_DATA_TBL;
CREATE COLUMN TABLE PAL_META_DATA_TBL(
    VARIABLE_NAME NVARCHAR(100),
    VARIABLE_TYPE NVARCHAR(100)
);
INSERT INTO PAL_META_DATA_TBL VALUES ('ATTR1', 'CONTINUOUS');
INSERT INTO PAL_META_DATA_TBL VALUES ('ATTR2', 'CONTINUOUS');
INSERT INTO PAL_META_DATA_TBL VALUES ('ATTR3', 'CATEGORICAL');
INSERT INTO PAL_META_DATA_TBL VALUES ('ATTR4', 'CATEGORICAL');
INSERT INTO PAL_META_DATA_TBL VALUES ('LABEL', 'TARGET');
DROP TABLE PAL_DATA_TBL;
CREATE COLUMN TABLE PAL_DATA_TBL (
    DATA_ID INTEGER,
    VARIABLE_NAME NVARCHAR(100),
    VAR_VALUE NVARCHAR(100),
    PURPOSE  INTEGER
);
INSERT INTO PAL_DATA_TBL VALUES(0,'ATTR1','1',2);
INSERT INTO PAL_DATA_TBL VALUES(0,'ATTR2','10',2);
INSERT INTO PAL_DATA_TBL VALUES(0,'ATTR3','100',2);
INSERT INTO PAL_DATA_TBL VALUES(0,'ATTR4','A',2);
INSERT INTO PAL_DATA_TBL VALUES(0,'LABEL','1',2);
INSERT INTO PAL_DATA_TBL VALUES(1,'ATTR1',1.1,1);
INSERT INTO PAL_DATA_TBL VALUES(1,'ATTR2',10.1,1);
INSERT INTO PAL_DATA_TBL VALUES(1,'ATTR3',100.1,1);
INSERT INTO PAL_DATA_TBL VALUES(1,'ATTR4','A',1);
```

```
INSERT INTO PAL_DATA_TBL VALUES(1,'LABEL',1,1);
INSERT INTO PAL_DATA_TBL VALUES(2,'ATTR1',1.2,1);
INSERT INTO PAL_DATA_TBL VALUES(2,'ATTR2',10.2,1);
INSERT INTO PAL_DATA_TBL VALUES(2,'ATTR3',100,1);
INSERT INTO PAL_DATA_TBL VALUES(2,'ATTR4','A',1);
INSERT INTO PAL_DATA_TBL VALUES(2,'LABEL',1,1);
INSERT INTO PAL_DATA_TBL VALUES(3,'ATTR1',1.3,1);
INSERT INTO PAL_DATA_TBL VALUES(3,'ATTR2',10.4,1);
INSERT INTO PAL_DATA_TBL VALUES(3,'ATTR3',100,1);
INSERT INTO PAL_DATA_TBL VALUES(3,'ATTR4','A',1);
INSERT INTO PAL_DATA_TBL VALUES(3,'LABEL',1,1);
INSERT INTO PAL_DATA_TBL VALUES(4,'ATTR1',1.2,1);
INSERT INTO PAL_DATA_TBL VALUES(4,'ATTR2',10.3,1);
INSERT INTO PAL_DATA_TBL VALUES(4,'ATTR3',100,1);
INSERT INTO PAL_DATA_TBL VALUES(4,'ATTR4','AB',1);
INSERT INTO PAL_DATA_TBL VALUES(4,'LABEL',1,1);
INSERT INTO PAL_DATA_TBL VALUES(5,'ATTR1',4,1);
INSERT INTO PAL_DATA_TBL VALUES(5,'ATTR2',40,1);
INSERT INTO PAL_DATA_TBL VALUES(5,'ATTR3',400,1);
INSERT INTO PAL_DATA_TBL VALUES(5,'ATTR4','AB',1);
INSERT INTO PAL_DATA_TBL VALUES(5,'LABEL',2,1);
INSERT INTO PAL_DATA_TBL VALUES(6,'ATTR1',4.1,2);
INSERT INTO PAL_DATA_TBL VALUES(6,'ATTR2',40,2);
INSERT INTO PAL_DATA_TBL VALUES(6,'ATTR3',400,2);
INSERT INTO PAL_DATA_TBL VALUES(6,'ATTR4','AB',2);
INSERT INTO PAL_DATA_TBL VALUES(6,'LABEL',2,2);
INSERT INTO PAL_DATA_TBL VALUES(7,'ATTR1',4.2,1);
INSERT INTO PAL_DATA_TBL VALUES(7,'ATTR2',40.2,1);
INSERT INTO PAL_DATA_TBL VALUES(7,'ATTR3',400,1);
INSERT INTO PAL_DATA_TBL VALUES(7,'ATTR4','AB',1);
INSERT INTO PAL_DATA_TBL VALUES(7,'LABEL',2,1);
INSERT INTO PAL_DATA_TBL VALUES(8,'ATTR1',4.3,1);
INSERT INTO PAL_DATA_TBL VALUES(8,'ATTR2',40.4,1);
INSERT INTO PAL_DATA_TBL VALUES(8,'ATTR3',400,1);
INSERT INTO PAL_DATA_TBL VALUES(8,'ATTR4','AB',1);
INSERT INTO PAL_DATA_TBL VALUES(8,'LABEL',2,1);
INSERT INTO PAL_DATA_TBL VALUES(9,'ATTR1',4.2,1);
INSERT INTO PAL_DATA_TBL VALUES(9,'ATTR2',40.3,1);
INSERT INTO PAL_DATA_TBL VALUES(9,'ATTR3',400,1);
INSERT INTO PAL_DATA_TBL VALUES(9,'ATTR4','AB',1);
INSERT INTO PAL_DATA_TBL VALUES(9,'LABEL',2,1);
INSERT INTO PAL_DATA_TBL VALUES(10,'ATTR1',9,1);
INSERT INTO PAL_DATA_TBL VALUES(10,'ATTR2',90,1);
INSERT INTO PAL_DATA_TBL VALUES(10,'ATTR3',900,1);
INSERT INTO PAL_DATA_TBL VALUES(10,'ATTR4','B',1);
INSERT INTO PAL_DATA_TBL VALUES(10,'LABEL',3,1);
INSERT INTO PAL_DATA_TBL VALUES(11,'ATTR1',9.1,1);
INSERT INTO PAL_DATA_TBL VALUES(11,'ATTR2',90.1,1);
INSERT INTO PAL_DATA_TBL VALUES(11,'ATTR3',900,1);
INSERT INTO PAL_DATA_TBL VALUES(11,'ATTR4','A',1);
INSERT INTO PAL_DATA_TBL VALUES(11,'LABEL',3,1);
INSERT INTO PAL_DATA_TBL VALUES(12,'ATTR1',9.2,2);
INSERT INTO PAL_DATA_TBL VALUES(12,'ATTR2',90.2,2);
INSERT INTO PAL_DATA_TBL VALUES(12,'ATTR3',900,2);
INSERT INTO PAL_DATA_TBL VALUES(12,'ATTR4','B',2);
INSERT INTO PAL_DATA_TBL VALUES(12,'LABEL',3,2);
INSERT INTO PAL_DATA_TBL VALUES(13,'ATTR1',9.3,1);
INSERT INTO PAL_DATA_TBL VALUES(13,'ATTR2',90.4,1);
INSERT INTO PAL_DATA_TBL VALUES(13,'ATTR3',900,1);
INSERT INTO PAL_DATA_TBL VALUES(13,'ATTR4','A',1);
INSERT INTO PAL_DATA_TBL VALUES(13,'LABEL',3,1);
INSERT INTO PAL_DATA_TBL VALUES(14,'ATTR1',9.2,1);
INSERT INTO PAL_DATA_TBL VALUES(14,'ATTR2',90.3,1);
INSERT INTO PAL_DATA_TBL VALUES(14,'ATTR3',900,1);
INSERT INTO PAL_DATA_TBL VALUES(14,'ATTR4','A',1);
INSERT INTO PAL_DATA_TBL VALUES(14,'LABEL',3,1);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
```

```
    "PARAM_NAME" VARCHAR (100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (100)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FUNCTION', NULL, NULL, 'RDT');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SPLIT_THRESHOLD', NULL, 0.0000001, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MIN_SAMPLES_LEAF', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('N_ESTIMATORS', 10, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MAX_DEPTH', 55, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SEED', 2, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('NTILES', 2, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PARTITION_METHOD', 2, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PARTITION_RANDOM_SEED', 2, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PARTITION_STRATIFIED_VARIABLE', NULL,
NULL, 'LABEL');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PARTITION_TRAINING_PERCENT', NULL, 0.7,
NULL);
DROP TABLE PAL_MODEL_RDT_TBL;
CREATE COLUMN TABLE PAL_MODEL_RDT_TBL (
    "ROW_INDEX" INTEGER,
    "PART_INDEX" INTEGER,
    "MODEL_CONTENT" NCLOB
);
CALL _SYS_AFL.PAL_UNIFIED_CLASSIFICATION_PIVOT (PAL_META_DATA_TBL, PAL_DATA_TBL,
#PAL_PARAMETER_TBL, PAL_MODEL_RDT_TBL, ?, ?, ?, ?, ?, ?, ?) WITH OVERVIEW;
```

**Example 2: Background data based kernel SHAP ( Support Vector Machine )**

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_META_DATA_TBL;
CREATE COLUMN TABLE PAL_META_DATA_TBL(
    VARIABLE_NAME NVARCHAR(100),
    VARIABLE_TYPE NVARCHAR(100)
);
INSERT INTO PAL_META_DATA_TBL VALUES ('ATTR1', 'CONTINUOUS');
INSERT INTO PAL_META_DATA_TBL VALUES ('ATTR2', 'CONTINUOUS');
INSERT INTO PAL_META_DATA_TBL VALUES ('ATTR3', 'CATEGORICAL');
INSERT INTO PAL_META_DATA_TBL VALUES ('ATTR4', 'CATEGORICAL');
INSERT INTO PAL_META_DATA_TBL VALUES ('LABEL', 'TARGET');
DROP TABLE PAL_DATA_TBL;
CREATE COLUMN TABLE PAL_DATA_TBL (
    DATA_ID INTEGER,
    VARIABLE_NAME NVARCHAR(100),
    VAR_VALUE NVARCHAR(100),
    PURPOSE  INTEGER
);
INSERT INTO PAL_DATA_TBL VALUES(0,'ATTR1','1',2);
INSERT INTO PAL_DATA_TBL VALUES(0,'ATTR2','10',2);
INSERT INTO PAL_DATA_TBL VALUES(0,'ATTR3','100',2);
INSERT INTO PAL_DATA_TBL VALUES(0,'ATTR4','A',2);
INSERT INTO PAL_DATA_TBL VALUES(0,'LABEL','1',2);
INSERT INTO PAL_DATA_TBL VALUES(1,'ATTR1',1.1,1);
INSERT INTO PAL_DATA_TBL VALUES(1,'ATTR2',10.1,1);
INSERT INTO PAL_DATA_TBL VALUES(1,'ATTR3',100.1,1);
INSERT INTO PAL_DATA_TBL VALUES(1,'ATTR4','A',1);
INSERT INTO PAL_DATA_TBL VALUES(1,'LABEL',1,1);
INSERT INTO PAL_DATA_TBL VALUES(2,'ATTR1',1.2,1);
INSERT INTO PAL_DATA_TBL VALUES(2,'ATTR2',10.2,1);
INSERT INTO PAL_DATA_TBL VALUES(2,'ATTR3',100,1);
INSERT INTO PAL_DATA_TBL VALUES(2,'ATTR4','A',1);
INSERT INTO PAL_DATA_TBL VALUES(2,'LABEL',1,1);
INSERT INTO PAL_DATA_TBL VALUES(3,'ATTR1',1.3,1);
INSERT INTO PAL_DATA_TBL VALUES(3,'ATTR2',10.4,1);
INSERT INTO PAL_DATA_TBL VALUES(3,'ATTR3',100,1);
INSERT INTO PAL_DATA_TBL VALUES(3,'ATTR4','A',1);
INSERT INTO PAL_DATA_TBL VALUES(3,'LABEL',1,1);
```

```
INSERT INTO PAL_DATA_TBL VALUES(4,'ATTR1',1.2,1);
INSERT INTO PAL_DATA_TBL VALUES(4,'ATTR2',10.3,1);
INSERT INTO PAL_DATA_TBL VALUES(4,'ATTR3',100,1);
INSERT INTO PAL_DATA_TBL VALUES(4,'ATTR4','AB',1);
INSERT INTO PAL_DATA_TBL VALUES(4,'LABEL',1,1);
INSERT INTO PAL_DATA_TBL VALUES(5,'ATTR1',4,1);
INSERT INTO PAL_DATA_TBL VALUES(5,'ATTR2',40,1);
INSERT INTO PAL_DATA_TBL VALUES(5,'ATTR3',400,1);
INSERT INTO PAL_DATA_TBL VALUES(5,'ATTR4','AB',1);
INSERT INTO PAL_DATA_TBL VALUES(5,'LABEL',2,1);
INSERT INTO PAL_DATA_TBL VALUES(6,'ATTR1',4.1,2);
INSERT INTO PAL_DATA_TBL VALUES(6,'ATTR2',40,2);
INSERT INTO PAL_DATA_TBL VALUES(6,'ATTR3',400,2);
INSERT INTO PAL_DATA_TBL VALUES(6,'ATTR4','AB',2);
INSERT INTO PAL_DATA_TBL VALUES(6,'LABEL',2,2);
INSERT INTO PAL_DATA_TBL VALUES(7,'ATTR1',4.2,1);
INSERT INTO PAL_DATA_TBL VALUES(7,'ATTR2',40.2,1);
INSERT INTO PAL_DATA_TBL VALUES(7,'ATTR3',400,1);
INSERT INTO PAL_DATA_TBL VALUES(7,'ATTR4','AB',1);
INSERT INTO PAL_DATA_TBL VALUES(7,'LABEL',2,1);
INSERT INTO PAL_DATA_TBL VALUES(8,'ATTR1',4.3,1);
INSERT INTO PAL_DATA_TBL VALUES(8,'ATTR2',40.4,1);
INSERT INTO PAL_DATA_TBL VALUES(8,'ATTR3',400,1);
INSERT INTO PAL_DATA_TBL VALUES(8,'ATTR4','AB',1);
INSERT INTO PAL_DATA_TBL VALUES(8,'LABEL',2,1);
INSERT INTO PAL_DATA_TBL VALUES(9,'ATTR1',4.2,1);
INSERT INTO PAL_DATA_TBL VALUES(9,'ATTR2',40.3,1);
INSERT INTO PAL_DATA_TBL VALUES(9,'ATTR3',400,1);
INSERT INTO PAL_DATA_TBL VALUES(9,'ATTR4','AB',1);
INSERT INTO PAL_DATA_TBL VALUES(9,'LABEL',2,1);
INSERT INTO PAL_DATA_TBL VALUES(10,'ATTR1',9,1);
INSERT INTO PAL_DATA_TBL VALUES(10,'ATTR2',90,1);
INSERT INTO PAL_DATA_TBL VALUES(10,'ATTR3',900,1);
INSERT INTO PAL_DATA_TBL VALUES(10,'ATTR4','B',1);
INSERT INTO PAL_DATA_TBL VALUES(10,'LABEL',3,1);
INSERT INTO PAL_DATA_TBL VALUES(11,'ATTR1',9.1,1);
INSERT INTO PAL_DATA_TBL VALUES(11,'ATTR2',90.1,1);
INSERT INTO PAL_DATA_TBL VALUES(11,'ATTR3',900,1);
INSERT INTO PAL_DATA_TBL VALUES(11,'ATTR4','A',1);
INSERT INTO PAL_DATA_TBL VALUES(11,'LABEL',3,1);
INSERT INTO PAL_DATA_TBL VALUES(12,'ATTR1',9.2,2);
INSERT INTO PAL_DATA_TBL VALUES(12,'ATTR2',90.2,2);
INSERT INTO PAL_DATA_TBL VALUES(12,'ATTR3',900,2);
INSERT INTO PAL_DATA_TBL VALUES(12,'ATTR4','B',2);
INSERT INTO PAL_DATA_TBL VALUES(12,'LABEL',3,2);
INSERT INTO PAL_DATA_TBL VALUES(13,'ATTR1',9.3,1);
INSERT INTO PAL_DATA_TBL VALUES(13,'ATTR2',90.4,1);
INSERT INTO PAL_DATA_TBL VALUES(13,'ATTR3',900,1);
INSERT INTO PAL_DATA_TBL VALUES(13,'ATTR4','A',1);
INSERT INTO PAL_DATA_TBL VALUES(13,'LABEL',3,1);
INSERT INTO PAL_DATA_TBL VALUES(14,'ATTR1',9.2,1);
INSERT INTO PAL_DATA_TBL VALUES(14,'ATTR2',90.3,1);
INSERT INTO PAL_DATA_TBL VALUES(14,'ATTR3',900,1);
INSERT INTO PAL_DATA_TBL VALUES(14,'ATTR4','A',1);
INSERT INTO PAL_DATA_TBL VALUES(14,'LABEL',3,1);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE  LOCAL TEMPORARY  COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (100)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FUNCTION', NULL, NULL, 'SVM');
--generate background data
INSERT INTO #PAL_PARAMETER_TBL VALUES ('BACKGROUND_SIZE', 5, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('BACKGROUND_SAMPLING_SEED', 1234, NULL,
NULL);
--parameters to be selected
```

```
INSERT INTO #PAL_PARAMETER_TBL VALUES ('RBF_GAMMA_VALUES',NULL,NULL,
'{0.01,0.05,0.07}');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SVM_C_VALUES',NULL,NULL, '{10,50,100}');
--parameter selection parameters
INSERT INTO #PAL_PARAMETER_TBL VALUES ('RESAMPLING_METHOD', NULL, NULL, 'cv');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('EVALUATION_METRIC', NULL, NULL,
'F1_SCORE');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FOLD_NUM', 2, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('REPEAT_TIMES', 2, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SEED', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PROGRESS_INDICATOR_ID', NULL, NULL,
'TEST');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PARAM_SEARCH_STRATEGY', NULL, NULL,
'grid');
DROP TABLE PAL_MODEL_SVM_TBL;
CREATE COLUMN TABLE PAL_MODEL_SVM_TBL (
    "ROW_INDEX" INTEGER,
    "PART_INDEX" INTEGER,
    "MODEL_CONTENT" NCLOB
);
CALL _SYS_AFL.PAL_UNIFIED_CLASSIFICATION_PIVOT (PAL_META_DATA_TBL, PAL_DATA_TBL,
#PAL_PARAMETER_TBL, PAL_MODEL_SVM_TBL, ?, ?, ?, ?, ?, ?, ?) WITH OVERVIEW;
```

**Expected Result**

MODEL:

| | ROW_INDEX | PART_INDEX | MODEL_CONTENT |
|---|---|---|---|
| 1 | 0 | 0 | {"means":[4.826666666666667,46.86],"deviations":[3.4172810132925157,34. |
| 2 | -1 | 1 | SVM |
| 3 | -2 | 2 | {"feature":["ATTR1","ATTR2","ATTR3","ATTR4","LABEL"],"featureNum":5,"i... |
| 4 | -10 | 3 | {"cate_vals":[{"cateVal":["900","400","400","400"]},{"cateVal":["A","AB","A... |

## _SYS_AFL.PAL_UNIFIED_CLASSIFICATION_PREDICT

**Parameter Table**

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| TOP_K_ATTRIBU-TIONS | INTEGER | 10 | Outputs the attribu-tions of top k features which contribute the most. | |
| FEATURE_ATTRIBU-TION_METHOD | INTEGER | 2 | Controls which method to use in model reasoning.<br>• 0: no reasoning<br>• 1: Saabas<br>• 2: tree SHAP | Valid for the functions:<br>• Decision Trees<br>• Random Decision Trees<br>• Hybrid Gradient Boosting Tree |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| SAMPLESIZE | INTEGER | 0 | Number of sampled combinations of features. If it equals to 0, then it will be determined by algorithm heuristically. | It is better to user a number which is larger than the number of the columns of the data table. |
| SEED | INTEGER | 0 | Specifies the seed for random number generator.<br>• 0: Uses the current time (in second) as seed<br>• Others: Uses the specified value as seed | Valid for the functions:<br>• Naive Bayes<br>• Support Vector Machine<br>• Multilayer Perceptron |

### Examples

Assume that:

- DM_PAL is a schema belonging to USER1;
- USER1 has been assigned the AFL__SYS_AFL_AFLPAL_EXECUTE or AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION role.

**Example 1: tree SHAP ( Random Decision Trees )**

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_PREDICT_DATA_TBL;
CREATE COLUMN TABLE PAL_PREDICT_DATA_TBL (
    DATA_ID INTEGER,
    VARIABLE_NAME NVARCHAR(100),
    VAR_VALUE NVARCHAR(100)
);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(0,'ATTR1','1');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(0,'ATTR2','10');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(0,'ATTR3','100');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(0,'ATTR4','A');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(1,'ATTR1',1.1);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(1,'ATTR2',10.1);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(1,'ATTR3',100.1);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(1,'ATTR4','A');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(2,'ATTR1',1.2);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(2,'ATTR2',10.2);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(2,'ATTR3',100);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(2,'ATTR4','A');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(3,'ATTR1',1.3);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(3,'ATTR2',10.4);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(3,'ATTR3',100);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(3,'ATTR4','A');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(4,'ATTR1',4);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(4,'ATTR2',40);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(4,'ATTR3',400);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(4,'ATTR4','AB');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(5,'ATTR1',9.2);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(5,'ATTR2',90.2);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(5,'ATTR3',900);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(5,'ATTR4','B');
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
```

```
    "PARAM_NAME" VARCHAR (100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (100)
);
CALL _SYS_AFL.PAL_UNIFIED_CLASSIFICATION_PREDICT_PIVOT(PAL_PREDICT_DATA_TBL,
PAL_MODEL_RDT_TBL, #PAL_PARAMETER_TBL, ?, ?);
```

### Expected Result

RESULT:

| | DATA_ID | SCORE | CONFIDENCE | REASON_CODE |
|---|---|---|---|---|
| 1 | 0 | 1 | 1 | [{"attr":"ATTR1","pct":49.0,"val":0.26666666666666661},{"attr":"ATTR2","p... |
| 2 | 1 | 1 | 0.8 | [{"attr":"ATTR1","pct":36.0,"val":0.26666666666666661},{"attr":"ATTR3","p... |
| 3 | 2 | 1 | 1 | [{"attr":"ATTR1","pct":49.0,"val":0.26666666666666661},{"attr":"ATTR2","p... |
| 4 | 3 | 1 | 1 | [{"attr":"ATTR1","pct":49.0,"val":0.26666666666666661},{"attr":"ATTR2","p... |
| 5 | 4 | 2 | 1 | [{"attr":"ATTR2","pct":33.0,"val":0.19333333333333333},{"attr":"ATTR3","p... |
| 6 | 5 | 3 | 0.8 | [{"attr":"ATTR1","pct":33.0,"val":0.21666666666666665},{"attr":"ATTR2","p... |

### Example 2: Background data based kernel SHAP ( Support Vector Machine )

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_PREDICT_DATA_TBL;
CREATE COLUMN TABLE PAL_PREDICT_DATA_TBL (
    DATA_ID INTEGER,
    VARIABLE_NAME NVARCHAR(100),
    VAR_VALUE NVARCHAR(100)
);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(0,'ATTR1','1');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(0,'ATTR2','10');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(0,'ATTR3','100');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(0,'ATTR4','A');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(1,'ATTR1',1.1);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(1,'ATTR2',10.1);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(1,'ATTR3',100.1);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(1,'ATTR4','A');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(2,'ATTR1',1.2);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(2,'ATTR2',10.2);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(2,'ATTR3',100);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(2,'ATTR4','A');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(3,'ATTR1',1.3);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(3,'ATTR2',10.4);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(3,'ATTR3',100);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(3,'ATTR4','A');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(4,'ATTR1',4);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(4,'ATTR2',40);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(4,'ATTR3',400);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(4,'ATTR4','AB');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(5,'ATTR1',9.2);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(5,'ATTR2',90.2);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(5,'ATTR3',900);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(5,'ATTR4','B');
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (100)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SEED', 1234, NULL, NULL);
CALL _SYS_AFL.PAL_UNIFIED_CLASSIFICATION_PREDICT_PIVOT(PAL_PREDICT_DATA_TBL,
PAL_MODEL_SVM_TBL, #PAL_PARAMETER_TBL, ?, ?);
```

### Expected Result

RESULT:

| | DATA_ID | SCORE | CONFIDENCE | REASON_CODE |
|---|---|---|---|---|
| 1 | 0 | 1 | 0.8406030283283473 | [{"attr":"ATTR2","val":0.2523507110301279,"pct":35.0},{"attr":"ATTR1","val... |
| 2 | 1 | 1 | 0.8052875342104481 | [{"attr":"ATTR2","val":0.25398279327371767,"pct":38.1},{"attr":"ATTR1","v... |
| 3 | 2 | 1 | 0.8327598759295809 | [{"attr":"ATTR2","val":0.25289248990773607,"pct":35.7},{"attr":"ATTR1","v... |
| 4 | 3 | 1 | 0.8282433697106885 | [{"attr":"ATTR2","val":0.2524429017746152,"pct":36.0},{"attr":"ATTR1","val... |
| 5 | 4 | 2 | 0.742285034749803 | [{"attr":"ATTR2","val":0.037539606777961777,"pct":31.7},{"attr":"ATTR1","... |
| 6 | 5 | 3 | 0.7942015079983878 | [{"attr":"ATTR2","val":0.21363827066219089,"pct":39.800000000000007},{... |

## _SYS_AFL.PAL_UNIFIED_CLASSIFICATION_SCORE

### Parameter Table

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| TOP_K_ATTRIBUTIONS | INTEGER | 10 | Outputs the attributions of top k features which contribute the most. | |
| FEATURE_ATTRIBUTION_METHOD | INTEGER | 2 | Controls which method to use in model reasoning.<br>• 0: no reasoning<br>• 1: Saabas<br>• 2: tree SHAP | Valid for the functions:<br>• Decision Trees<br>• Random Decision Trees<br>• Hybrid Gradient Boosting Tree |
| SAMPLESIZE | INTEGER | 0 | Number of sampled combinations of features. If it equals to 0, then it will be determined by algorithm heuristically. | It is better to user a number which is larger than the number of the columns of the data table. |
| SEED | INTEGER | 0 | Specifies the seed for random number generator.<br>• 0: Uses the current time (in second) as seed<br>• Others: Uses the specified value as seed | Valid for the functions:<br>• Naive Bayes<br>• Support Vector Machine<br>• Multilayer Perceptron |

### Examples

Assume that:

- DM_PAL is a schema belonging to USER1;

- USER1 has been assigned the AFL__SYS_AFL_AFLPAL_EXECUTE or AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION role.

### Example 1: tree SHAP ( Random Decision Trees )

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_SCORING_DATA_TBL;
CREATE COLUMN TABLE PAL_SCORING_DATA_TBL (
    DATA_ID INTEGER,
    VARIABLE_NAME NVARCHAR(100),
    VAR_VALUE NVARCHAR(100)
);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(0,'ATTR1','1');
INSERT INTO PAL_SCORING_DATA_TBL VALUES(0,'ATTR2','10');
INSERT INTO PAL_SCORING_DATA_TBL VALUES(0,'ATTR3','100');
INSERT INTO PAL_SCORING_DATA_TBL VALUES(0,'ATTR4','A');
INSERT INTO PAL_SCORING_DATA_TBL VALUES(0,'LABEL','1');
INSERT INTO PAL_SCORING_DATA_TBL VALUES(1,'ATTR1',1.1);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(1,'ATTR2',10.1);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(1,'ATTR3',100.1);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(1,'ATTR4','A');
INSERT INTO PAL_SCORING_DATA_TBL VALUES(1,'LABEL','1');
INSERT INTO PAL_SCORING_DATA_TBL VALUES(2,'ATTR1',1.2);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(2,'ATTR2',10.2);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(2,'ATTR3',100);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(2,'ATTR4','A');
INSERT INTO PAL_SCORING_DATA_TBL VALUES(2,'LABEL','1');
INSERT INTO PAL_SCORING_DATA_TBL VALUES(3,'ATTR1',1.3);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(3,'ATTR2',10.4);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(3,'ATTR3',100);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(3,'ATTR4','A');
INSERT INTO PAL_SCORING_DATA_TBL VALUES(3,'LABEL','1');
INSERT INTO PAL_SCORING_DATA_TBL VALUES(4,'ATTR1',4);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(4,'ATTR2',40);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(4,'ATTR3',400);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(4,'ATTR4','AB');
INSERT INTO PAL_SCORING_DATA_TBL VALUES(4,'LABEL','2');
INSERT INTO PAL_SCORING_DATA_TBL VALUES(5,'ATTR1',9.2);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(5,'ATTR2',90.2);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(5,'ATTR3',900);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(5,'ATTR4','B');
INSERT INTO PAL_SCORING_DATA_TBL VALUES(5,'LABEL','3');
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (100)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MAX_RESULT_NUM', 2, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('NTILES', 2, NULL, NULL);
CALL _SYS_AFL.PAL_UNIFIED_CLASSIFICATION_SCORE_PIVOT(PAL_SCORING_DATA_TBL,
PAL_MODEL_RDT_TBL, #PAL_PARAMETER_TBL, ?, ?,?,?);
```

### Expected Result

RESULT:

| | DATA_ID | SCORE | CONFIDENCE | REASON_CODE |
|---|---|---|---|---|
| 1 | 0 | 1 | 1 | [{"attr":"ATTR1","pct":49.0,"val":0.26666666666666661},{"attr":"ATTR2","p... |
| 2 | 1 | 1 | 0.8 | [{"attr":"ATTR1","pct":36.0,"val":0.26666666666666661},{"attr":"ATTR3","p... |

### Example 2: Background data based kernel SHAP ( Support Vector Machine )

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_SCORING_DATA_TBL;
CREATE COLUMN TABLE PAL_SCORING_DATA_TBL (
    DATA_ID INTEGER,
    VARIABLE_NAME NVARCHAR(100),
    VAR_VALUE NVARCHAR(100)
);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(0,'ATTR1','1');
INSERT INTO PAL_SCORING_DATA_TBL VALUES(0,'ATTR2','10');
INSERT INTO PAL_SCORING_DATA_TBL VALUES(0,'ATTR3','100');
INSERT INTO PAL_SCORING_DATA_TBL VALUES(0,'ATTR4','A');
INSERT INTO PAL_SCORING_DATA_TBL VALUES(0,'LABEL','1');
INSERT INTO PAL_SCORING_DATA_TBL VALUES(1,'ATTR1',1.1);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(1,'ATTR2',10.1);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(1,'ATTR3',100.1);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(1,'ATTR4','A');
INSERT INTO PAL_SCORING_DATA_TBL VALUES(1,'LABEL','1');
INSERT INTO PAL_SCORING_DATA_TBL VALUES(2,'ATTR1',1.2);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(2,'ATTR2',10.2);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(2,'ATTR3',100);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(2,'ATTR4','A');
INSERT INTO PAL_SCORING_DATA_TBL VALUES(2,'LABEL','1');
INSERT INTO PAL_SCORING_DATA_TBL VALUES(3,'ATTR1',1.3);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(3,'ATTR2',10.4);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(3,'ATTR3',100);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(3,'ATTR4','A');
INSERT INTO PAL_SCORING_DATA_TBL VALUES(3,'LABEL','1');
INSERT INTO PAL_SCORING_DATA_TBL VALUES(4,'ATTR1',4);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(4,'ATTR2',40);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(4,'ATTR3',400);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(4,'ATTR4','AB');
INSERT INTO PAL_SCORING_DATA_TBL VALUES(4,'LABEL','2');
INSERT INTO PAL_SCORING_DATA_TBL VALUES(5,'ATTR1',9.2);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(5,'ATTR2',90.2);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(5,'ATTR3',900);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(5,'ATTR4','B');
INSERT INTO PAL_SCORING_DATA_TBL VALUES(5,'LABEL','3');
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (100)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MAX_RESULT_NUM', 2, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('NTILES', 2, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SEED', 1234, NULL, NULL);
CALL _SYS_AFL.PAL_UNIFIED_CLASSIFICATION_SCORE_PIVOT(PAL_SCORING_DATA_TBL,
PAL_MODEL_SVM_TBL, #PAL_PARAMETER_TBL, ?, ?,?,?);
```

### Expected Result

RESULT:

| | DATA_ID | SCORE | CONFIDENCE | REASON_CODE |
|---|---|---|---|---|
| 1 | 0 | 1 | 0.8406030283283473 | [{"attr":"ATTR2","val":0.2523507110301279,"pct":35.0},{"attr":"ATTR1","val... |
| 2 | 1 | 1 | 0.8052875342104481 | [{"attr":"ATTR2","val":0.25398279327371767,"pct":38.1},{"attr":"ATTR1","v... |

## 5.13.2.5 Model Compression

The training process may cost a large size of memory. Model compression aims to reduce the size of the model with minimum loss of the accuracy.

Model compression is provided for Support Vector Machine, Random Decision Trees, and Hybrid Gradient Boosting Tree. The following tables list the parameters for using model compression in Support Vector Machine, Random Decision Trees, and Hybrid Gradient Boosting Tree respectively.

Support Vector Machine

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| COMPRESSION | INTEGER | 1 | Specifies if the model is stored in compressed format.<br>• 0: no<br>• 1: yes | |
| MAX_BITS | INTEGER | 12 | The maximum number of bits to quantize continuous features. Equivalent to use $2^{MAX_BITS}$ bins. Must be less than 31. | Valid only when the value of COMPRESSION is 1. |
| MAX_QUANTIZATION_ITER | INTEGER | 1000 | The maximum iteration steps for quantization. | Valid only when the value of COMPRESSION is 1. |

Random Decision Trees

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| COMPRESSION | INTEGER | 0 | Specifies if the model is stored in compressed format.<br>• 0 : no<br>• 1: yes | |
| MAX_BITS | INTEGER | 12 | The maximum number of bits to quantize continuous features. Equivalent to use $2^{MAX_BITS}$ bins. Must be less than 31. | Valid only when the value of COMPRESSION is 1. |
| QUANTIZE_RATE | DOUBLE | 0.005 | Quantize a feature if the largest class frequency of which is less than QUANTIZE_RATE. | Valid only when the value of COMPRESSION is 1. |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| FITTINGS_QUANTIZATION | INTEGER | 0 | Indicate whether to quantize the fitting values.<br>• 0 : no<br>• 1: yes | Valid only when the value of COMPRESSION is 1. |

**Hybrid Gradient Boosting Tree**

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| COMPRESSION | INTEGER | 0 | Indicates whether result model should be compressed:<br>• 0. no;<br>• 1. yes. | |
| MAX_BITS | INTEGER | 12 | Maximum number of bits to quantize continuous features. Equivalent to use $2^{MAX_BITS}$ bins.<br>Must be less than 31. | Valid only when the value of COMPRESSION is 1. |

## Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

**Example: Random Decision Trees Model Compression**

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_META_DATA_TBL;
CREATE COLUMN TABLE PAL_META_DATA_TBL(
    VARIABLE_NAME NVARCHAR(100),
    VARIABLE_TYPE NVARCHAR(100)
);
INSERT INTO PAL_META_DATA_TBL VALUES ('ATTR1', 'CONTINUOUS');
INSERT INTO PAL_META_DATA_TBL VALUES ('ATTR2', 'CONTINUOUS');
INSERT INTO PAL_META_DATA_TBL VALUES ('ATTR3', 'CATEGORICAL');
INSERT INTO PAL_META_DATA_TBL VALUES ('ATTR4', 'CATEGORICAL');
INSERT INTO PAL_META_DATA_TBL VALUES ('LABEL', 'TARGET');
DROP TABLE PAL_DATA_TBL;
CREATE COLUMN TABLE PAL_DATA_TBL (
    DATA_ID INTEGER,
    VARIABLE_NAME NVARCHAR(100),
    VAR_VALUE NVARCHAR(100),
    PURPOSE  INTEGER
);
INSERT INTO PAL_DATA_TBL VALUES(0,'ATTR1','1',2);
```

```
INSERT INTO PAL_DATA_TBL VALUES(0,'ATTR2','10',2);
INSERT INTO PAL_DATA_TBL VALUES(0,'ATTR3','100',2);
INSERT INTO PAL_DATA_TBL VALUES(0,'ATTR4','A',2);
INSERT INTO PAL_DATA_TBL VALUES(0,'LABEL','1',2);
INSERT INTO PAL_DATA_TBL VALUES(1,'ATTR1',1.1,1);
INSERT INTO PAL_DATA_TBL VALUES(1,'ATTR2',10.1,1);
INSERT INTO PAL_DATA_TBL VALUES(1,'ATTR3',100.1,1);
INSERT INTO PAL_DATA_TBL VALUES(1,'ATTR4','A',1);
INSERT INTO PAL_DATA_TBL VALUES(1,'LABEL',1,1);
INSERT INTO PAL_DATA_TBL VALUES(2,'ATTR1',1.2,1);
INSERT INTO PAL_DATA_TBL VALUES(2,'ATTR2',10.2,1);
INSERT INTO PAL_DATA_TBL VALUES(2,'ATTR3',100,1);
INSERT INTO PAL_DATA_TBL VALUES(2,'ATTR4','A',1);
INSERT INTO PAL_DATA_TBL VALUES(2,'LABEL',1,1);
INSERT INTO PAL_DATA_TBL VALUES(3,'ATTR1',1.3,1);
INSERT INTO PAL_DATA_TBL VALUES(3,'ATTR2',10.4,1);
INSERT INTO PAL_DATA_TBL VALUES(3,'ATTR3',100,1);
INSERT INTO PAL_DATA_TBL VALUES(3,'ATTR4','A',1);
INSERT INTO PAL_DATA_TBL VALUES(3,'LABEL',1,1);
INSERT INTO PAL_DATA_TBL VALUES(4,'ATTR1',1.2,1);
INSERT INTO PAL_DATA_TBL VALUES(4,'ATTR2',10.3,1);
INSERT INTO PAL_DATA_TBL VALUES(4,'ATTR3',100,1);
INSERT INTO PAL_DATA_TBL VALUES(4,'ATTR4','AB',1);
INSERT INTO PAL_DATA_TBL VALUES(4,'LABEL',1,1);
INSERT INTO PAL_DATA_TBL VALUES(5,'ATTR1',4,1);
INSERT INTO PAL_DATA_TBL VALUES(5,'ATTR2',40,1);
INSERT INTO PAL_DATA_TBL VALUES(5,'ATTR3',400,1);
INSERT INTO PAL_DATA_TBL VALUES(5,'ATTR4','AB',1);
INSERT INTO PAL_DATA_TBL VALUES(5,'LABEL',2,1);
INSERT INTO PAL_DATA_TBL VALUES(6,'ATTR1',4.1,2);
INSERT INTO PAL_DATA_TBL VALUES(6,'ATTR2',40,2);
INSERT INTO PAL_DATA_TBL VALUES(6,'ATTR3',400,2);
INSERT INTO PAL_DATA_TBL VALUES(6,'ATTR4','AB',2);
INSERT INTO PAL_DATA_TBL VALUES(6,'LABEL',2,2);
INSERT INTO PAL_DATA_TBL VALUES(7,'ATTR1',4.2,1);
INSERT INTO PAL_DATA_TBL VALUES(7,'ATTR2',40.2,1);
INSERT INTO PAL_DATA_TBL VALUES(7,'ATTR3',400,1);
INSERT INTO PAL_DATA_TBL VALUES(7,'ATTR4','AB',1);
INSERT INTO PAL_DATA_TBL VALUES(7,'LABEL',2,1);
INSERT INTO PAL_DATA_TBL VALUES(8,'ATTR1',4.3,1);
INSERT INTO PAL_DATA_TBL VALUES(8,'ATTR2',40.4,1);
INSERT INTO PAL_DATA_TBL VALUES(8,'ATTR3',400,1);
INSERT INTO PAL_DATA_TBL VALUES(8,'ATTR4','AB',1);
INSERT INTO PAL_DATA_TBL VALUES(8,'LABEL',2,1);
INSERT INTO PAL_DATA_TBL VALUES(9,'ATTR1',4.2,1);
INSERT INTO PAL_DATA_TBL VALUES(9,'ATTR2',40.3,1);
INSERT INTO PAL_DATA_TBL VALUES(9,'ATTR3',400,1);
INSERT INTO PAL_DATA_TBL VALUES(9,'ATTR4','AB',1);
INSERT INTO PAL_DATA_TBL VALUES(9,'LABEL',2,1);
INSERT INTO PAL_DATA_TBL VALUES(10,'ATTR1',9,1);
INSERT INTO PAL_DATA_TBL VALUES(10,'ATTR2',90,1);
INSERT INTO PAL_DATA_TBL VALUES(10,'ATTR3',900,1);
INSERT INTO PAL_DATA_TBL VALUES(10,'ATTR4','B',1);
INSERT INTO PAL_DATA_TBL VALUES(10,'LABEL',3,1);
INSERT INTO PAL_DATA_TBL VALUES(11,'ATTR1',9.1,1);
INSERT INTO PAL_DATA_TBL VALUES(11,'ATTR2',90.1,1);
INSERT INTO PAL_DATA_TBL VALUES(11,'ATTR3',900,1);
INSERT INTO PAL_DATA_TBL VALUES(11,'ATTR4','A',1);
INSERT INTO PAL_DATA_TBL VALUES(11,'LABEL',3,1);
INSERT INTO PAL_DATA_TBL VALUES(12,'ATTR1',9.2,2);
INSERT INTO PAL_DATA_TBL VALUES(12,'ATTR2',90.2,2);
INSERT INTO PAL_DATA_TBL VALUES(12,'ATTR3',900,2);
INSERT INTO PAL_DATA_TBL VALUES(12,'ATTR4','B',2);
INSERT INTO PAL_DATA_TBL VALUES(12,'LABEL',3,2);
INSERT INTO PAL_DATA_TBL VALUES(13,'ATTR1',9.3,1);
INSERT INTO PAL_DATA_TBL VALUES(13,'ATTR2',90.4,1);
INSERT INTO PAL_DATA_TBL VALUES(13,'ATTR3',900,1);
INSERT INTO PAL_DATA_TBL VALUES(13,'ATTR4','A',1);
```

```
INSERT INTO PAL_DATA_TBL VALUES(13,'LABEL',3,1);
INSERT INTO PAL_DATA_TBL VALUES(14,'ATTR1',9.2,1);
INSERT INTO PAL_DATA_TBL VALUES(14,'ATTR2',90.3,1);
INSERT INTO PAL_DATA_TBL VALUES(14,'ATTR3',900,1);
INSERT INTO PAL_DATA_TBL VALUES(14,'ATTR4','A',1);
INSERT INTO PAL_DATA_TBL VALUES(14,'LABEL',3,1);
DROP TABLE PAL_PARAMETER_TBL;
CREATE  COLUMN TABLE PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (100)
);
INSERT INTO PAL_PARAMETER_TBL VALUES ('FUNCTION', NULL, NULL, 'RDT');
INSERT INTO PAL_PARAMETER_TBL VALUES ('SPLIT_THRESHOLD', NULL, 0.0000001, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('MIN_SAMPLES_LEAF', 1, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('N_ESTIMATORS', 10, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('MAX_DEPTH', 55, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('SEED', 2, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('NTILES', 2, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('COMPRESSION', 1, NULL,
NULL);                      ------parameter for model compression
INSERT INTO PAL_PARAMETER_TBL VALUES ('PARTITION_METHOD', 2, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('PARTITION_RANDOM_SEED', 2, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('PARTITION_STRATIFIED_VARIABLE', NULL,
NULL, 'LABEL');
INSERT INTO PAL_PARAMETER_TBL VALUES ('PARTITION_TRAINING_PERCENT', NULL, 0.7,
NULL);
DROP TABLE PAL_MODEL_RDT_TBL;
CREATE COLUMN TABLE PAL_MODEL_RDT_TBL (
    "ROW_INDEX" INTEGER,
    "PART_INDEX" INTEGER,
    "MODEL_CONTENT" NCLOB
);
DO
BEGIN
    lt_meta_data = SELECT * FROM PAL_META_DATA_TBL;
    lt_data = SELECT * FROM PAL_DATA_TBL;
    lt_ctrl = SELECT * FROM PAL_PARAMETER_TBL;
    CALL _SYS_AFL.PAL_UNIFIED_CLASSIFICATION_PIVOT
(:lt_meta_data, :lt_data, :lt_ctrl, lt_model, lt_imp,lt_stat,
lt_opt,lt_cm,lt_metrics,lt_partition,lt_ph1);
    INSERT INTO PAL_MODEL_RDT_TBL SELECT * FROM :lt_model;
END;
SELECT * FROM PAL_MODEL_RDT_TBL;
```

Expected Result

MODEL:

| | ROW_INDEX | PART_INDEX | MODEL_CONTENT |
|---|---|---|---|
| 1 | 0 | -1 | {"atFields":[{"dataType":"double","name":"ATTR1","opType":"cont","value... |
| 2 | 1 | 0 | {"cw":"RL1%\u0018c\f\u0001\u0000","np":"\u0005\u0005\u0005\u0005\... |
| 3 | 2 | 1 | {"cw":"\u001ct\u001ctOE\u0013q#/c/a\u0017 %h\u0005\u000e\u0005\u... |
| 4 | 3 | 2 | {"cw":"\u0006P\u0003(\u0012vl;#VAk!8\u001e\\\u000f3&\u0019S\u0003... |
| 5 | 4 | 3 | {"at":0,"cvs":["2.6","5.1","6.55"],"dicts":[[{"p":0.5,"v":0},{"p":0.5,"v":2}],[{"p":... |
| 6 | 5 | 4 | {"at":1,"cvs":["65","65.3","25.15"],"dicts":[[{"p":1.0,"v":1}],[{"p":0.5,"v":0},{"p... |
| 7 | 6 | 5 | {"at":2,"cvs":["\u0003\u0001","\u0002","\u0003","\u0000","\u0001\u0000... |
| 8 | 7 | 6 | {"at":3,"cvs":["\u0001","\u0000"],"dicts":[[{"p":0.5,"v":0},{"p":0.5,"v":1}]],"ov... |
| 9 | -1 | 7 | RDT |
| 10 | -2 | 8 | {"feature":["ATTR1","ATTR2","ATTR3","ATTR4","LABEL"],"featureNum":5,"i... |

## 5.13.2.6 Missing Value Handling

Missing values in each column of the table can be handled by one of the following imputation types:

- MODE: Fills all missing values by the mode of the column, that is, the value that appears most often in the column. This type only suits column with categorical variables.
- MEAN: Fills all missing values by the mean of the column, that is, the average of all non-missing values in the column. This type only suits column with numerical variables.
- MEDIAN: Fills all missing values by the median of the column, that is, the value that separates the higher half of the non-missing values from the lower half. If there is an even number of observations, then the median is given by the mean of the two middle values. This type only suits column with numerical variables.
- ALS: Fills each missing value by the value imputed by a matrix completion model trained using alternating least squares method. This type only suits column with numerical variables.
- SPECIFIED_CATEGORICAL_VALUE: Fills all missing values by a specified categorical value. This type only suits column with categorical variables.
- SPECIFIED_NUMERICAL_VALUE: Fills all missing values by a specified numerical value. This type only suits column with numerical variables.
- DELETE: Deletes all rows with missing values. The entire row in table will be deleted.
- NONE: Does nothing. Leave the column untouched.

More detailed information is provided with the guidance of Missing Value Handling algorithm (preprocessing algorithms section).

**Parameter Table**

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| HANDLE_MISSING_VALUE | INTEGER | 0 | Whether to handle missing values:<br>• 0: No<br>• 1: Yes | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| IMPUTATION_TYPE | INTEGER | 1 | Chooses the overall default imputation type for all columns:<br>• 0: NONE<br>• 1: MODE_MEAN<br>• 2: MODE_MEDIAN<br>• 3: MODE_ALL-ZERO<br>• 4: MODE_ALS<br>• 5: DELETE<br>NONE and DELETE: set all columns to imputation type NONE/DELETE.<br>MODE_MEAN, MODE_MEDIAN, MODE_ALLZERO, and MODE_ALS: set categorical columns to the first imputation type (the type before “”); set numerical columns to the second imputation type (the type after “”). ALLZERO means to fill all missing values in numerical columns with 0. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| <column name>_IMPUTATION_TYPE | INTEGER | No default value | Sets imputation type of a specified column. This value overwrites the default imputation type deduced by the IMPUTATION_TYPE parameter:<br>• 0: NONE<br>• 1: DELETE<br>• 100: MODE<br>• 101: SPECIFIED_CATEGORICAL_VALUE<br>• 200: MEAN<br>• 201: MEDIAN<br>• 203:SPECIFIED_NUMERICAL_VALUE<br>• 204: ALS<br>For type 101 (SPECIFIED_CATEGORICAL_VALUE), use the following syntax to specify a categorical value to be filled in a column V0:<br>INSERT INTO #PAL_PARAMETER_TBL VALUES('V0_IMPUTATION_TYPE', 101, NULL, '<CATEGORICAL_VALUE>');<br>For type 203 (SPECIFIED_NUMERICAL_VALUE), use the following syntax to specify a numerical value to be filled in a column V1:<br>INSERT INTO #PAL_PARAMETER_TBL VALUES('V1_IMPUTA- | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| | | | TION_TYPE', 203, `<NUMERICAL_VALUE>`, NULL);<br><br>**Note**<br>A column of integer type can be set to type 101 (SPECIFIED_CATEGORICAL_VALUE) or type 203 (SPECIFIED_NUMERICAL_VALUE):<br>• when setting to type 101 (SPECIFIED_CATEGORICAL_VALUE), the string must contain a valid integer value, otherwise an error will be thrown;<br>• when setting to type 203 (SPECIFIED_NUMERICAL_VALUE), the double value will be converted to integer. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| ALS_FACTOR_NUMBER | INTEGER | 3 | Length of factor vectors, that is, the f in ALS model.<br>**Note**<br>ALS_FACTOR_NUMBER should be less than the number of numerical columns to get a meaningful result. | |
| ALS_REGULARIZATION | DOUBLE | 1e-2 | L2 regularization of the factors, that is, the λ in ALS model. | |
| ALS_MAX_ITERATION | INTEGER | 20 | Maximum number of iterations when training ALS model. | |
| ALS_SEED | INTEGER | 0 | Specifies the seed of the random number generator used in the training of ALS model:<br>• 0: Uses the current time as the seed<br>• Others: Uses the specified value as the seed. | |
| ALS_EXIT_THRESHOLD | DOUBLE | 0.0 | The training of ALS model exits if the value of cost function is not decreased more than the value of ALS_EXIT_THRESHOLD since the last check.<br>If ALS_EXIT_THRESHOLD is set to 0, there is no check and the training only exits on reaching the maximum number of iterations. Evaluations of cost function require additional calculations. You can set this parameter to 0 to avoid it. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| ALS_EXIT_INTERVAL | INTEGER | 5 | During the training of ALS model, it will calculate cost function and check the exit criteria every `<ALS_EXIT_INTERVAL>` iterations. Evaluations of cost function require additional calculations. | |
| ALS_LINEAR_SYSTEM_SOLVER | INTEGER | 0 | • 0: cholesky solver<br>• 1: cg solver (recommended when ALS_FACTOR_NUMBER is large) | |
| ALS_CG_MAX_ITERATION | INTEGER | 3 | Maximum number of iteration of cg solver. | |
| ALS_CENTERING | INTEGER | 1 | Whether to center the data by column before training ALS model.<br>• 0: no centering<br>• 1: centering. | |
| ALS_SCALING | INTEGER | 1 | Whether to scale the data by column before training ALS model.<br>• 0: no scaling<br>• 1: scaling | |

### Examples

Assume that:

- DM_PAL is a schema belonging to USER1;
- USER1 has been assigned the AFL__SYS_AFL_AFLPAL_EXECUTE or AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION role.

#### Example 1: Multi-Class Logistic Regression (Training)

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_META_DATA_TBL;
CREATE COLUMN TABLE PAL_META_DATA_TBL(
    VARIABLE_NAME NVARCHAR(100),
    VARIABLE_TYPE NVARCHAR(100)
);
INSERT INTO PAL_META_DATA_TBL VALUES ('ATTR1', 'CONTINUOUS');
INSERT INTO PAL_META_DATA_TBL VALUES ('ATTR2', 'CONTINUOUS');
INSERT INTO PAL_META_DATA_TBL VALUES ('ATTR3', 'CATEGORICAL');
INSERT INTO PAL_META_DATA_TBL VALUES ('ATTR4', 'CATEGORICAL');
INSERT INTO PAL_META_DATA_TBL VALUES ('LABEL', 'TARGET');
DROP TABLE PAL_DATA_TBL;
CREATE COLUMN TABLE PAL_DATA_TBL (
```

```
    DATA_ID INTEGER,
    VARIABLE_NAME NVARCHAR(100),
    VAR_VALUE NVARCHAR(100),
    PURPOSE  INTEGER
);
INSERT INTO PAL_DATA_TBL VALUES(0,'ATTR1','1',2);
INSERT INTO PAL_DATA_TBL VALUES(0,'ATTR2','10',2);
INSERT INTO PAL_DATA_TBL VALUES(0,'ATTR3','100',2);
INSERT INTO PAL_DATA_TBL VALUES(0,'ATTR4','A',2);
INSERT INTO PAL_DATA_TBL VALUES(0,'LABEL','1',2);
INSERT INTO PAL_DATA_TBL VALUES(1,'ATTR1',1.1,1);
INSERT INTO PAL_DATA_TBL VALUES(1,'ATTR2',10.1,1);
INSERT INTO PAL_DATA_TBL VALUES(1,'ATTR3',100.1,1);
INSERT INTO PAL_DATA_TBL VALUES(1,'ATTR4','A',1);
INSERT INTO PAL_DATA_TBL VALUES(1,'LABEL',1,1);
INSERT INTO PAL_DATA_TBL VALUES(2,'ATTR1',1.2,1);
INSERT INTO PAL_DATA_TBL VALUES(2,'ATTR2',10.2,1);
INSERT INTO PAL_DATA_TBL VALUES(2,'ATTR3',100,1);
INSERT INTO PAL_DATA_TBL VALUES(2,'ATTR4','A',1);
INSERT INTO PAL_DATA_TBL VALUES(2,'LABEL',1,1);
INSERT INTO PAL_DATA_TBL VALUES(3,'ATTR1',1.3,1);
INSERT INTO PAL_DATA_TBL VALUES(3,'ATTR2',10.4,1);
INSERT INTO PAL_DATA_TBL VALUES(3,'ATTR3',100,1);
INSERT INTO PAL_DATA_TBL VALUES(3,'ATTR4','A',1);
INSERT INTO PAL_DATA_TBL VALUES(3,'LABEL',1,1);
INSERT INTO PAL_DATA_TBL VALUES(4,'ATTR1',1.2,1);
INSERT INTO PAL_DATA_TBL VALUES(4,'ATTR2',10.3,1);
INSERT INTO PAL_DATA_TBL VALUES(4,'ATTR3',100,1);
INSERT INTO PAL_DATA_TBL VALUES(4,'ATTR4','AB',1);
INSERT INTO PAL_DATA_TBL VALUES(4,'LABEL',1,1);
INSERT INTO PAL_DATA_TBL VALUES(5,'ATTR1',4,1);
INSERT INTO PAL_DATA_TBL VALUES(5,'ATTR2',40,1);
INSERT INTO PAL_DATA_TBL VALUES(5,'ATTR3',400,1);
INSERT INTO PAL_DATA_TBL VALUES(5,'ATTR4','AB',1);
INSERT INTO PAL_DATA_TBL VALUES(5,'LABEL',2,1);
INSERT INTO PAL_DATA_TBL VALUES(6,'ATTR1',4.1,2);
INSERT INTO PAL_DATA_TBL VALUES(6,'ATTR2',40,2);
INSERT INTO PAL_DATA_TBL VALUES(6,'ATTR3',400,2);
INSERT INTO PAL_DATA_TBL VALUES(6,'ATTR4','AB',2);
INSERT INTO PAL_DATA_TBL VALUES(6,'LABEL',2,2);
INSERT INTO PAL_DATA_TBL VALUES(7,'ATTR1',4.2,1);
INSERT INTO PAL_DATA_TBL VALUES(7,'ATTR2',40.2,1);
INSERT INTO PAL_DATA_TBL VALUES(7,'ATTR3',400,1);
INSERT INTO PAL_DATA_TBL VALUES(7,'ATTR4','AB',1);
INSERT INTO PAL_DATA_TBL VALUES(7,'LABEL',2,1);
INSERT INTO PAL_DATA_TBL VALUES(8,'ATTR1',4.3,1);
INSERT INTO PAL_DATA_TBL VALUES(8,'ATTR2',40.4,1);
INSERT INTO PAL_DATA_TBL VALUES(8,'ATTR3',400,1);
INSERT INTO PAL_DATA_TBL VALUES(8,'ATTR4','AB',1);
INSERT INTO PAL_DATA_TBL VALUES(8,'LABEL',2,1);
INSERT INTO PAL_DATA_TBL VALUES(9,'ATTR1',4.2,1);
INSERT INTO PAL_DATA_TBL VALUES(9,'ATTR2',40.3,1);
INSERT INTO PAL_DATA_TBL VALUES(9,'ATTR3',400,1);
INSERT INTO PAL_DATA_TBL VALUES(9,'ATTR4','AB',1);
INSERT INTO PAL_DATA_TBL VALUES(9,'LABEL',2,1);
INSERT INTO PAL_DATA_TBL VALUES(10,'ATTR1',9,1);
--INSERT INTO PAL_DATA_TBL VALUES(10,'ATTR2',90,1);
INSERT INTO PAL_DATA_TBL VALUES(10,'ATTR3',900,1);
INSERT INTO PAL_DATA_TBL VALUES(10,'ATTR4','B',1);
INSERT INTO PAL_DATA_TBL VALUES(10,'LABEL',3,1);
INSERT INTO PAL_DATA_TBL VALUES(11,'ATTR1',9.1,1);
INSERT INTO PAL_DATA_TBL VALUES(11,'ATTR2',90.1,1);
INSERT INTO PAL_DATA_TBL VALUES(11,'ATTR3',900,1);
INSERT INTO PAL_DATA_TBL VALUES(11,'ATTR4','A',1);
INSERT INTO PAL_DATA_TBL VALUES(11,'LABEL',3,1);
INSERT INTO PAL_DATA_TBL VALUES(12,'ATTR1',9.2,2);
--INSERT INTO PAL_DATA_TBL VALUES(12,'ATTR2',90.2,2);
INSERT INTO PAL_DATA_TBL VALUES(12,'ATTR3',900,2);
```

```
INSERT INTO PAL_DATA_TBL VALUES(12,'ATTR4','B',2);
INSERT INTO PAL_DATA_TBL VALUES(12,'LABEL',3,2);
INSERT INTO PAL_DATA_TBL VALUES(13,'ATTR1',9.3,1);
INSERT INTO PAL_DATA_TBL VALUES(13,'ATTR2',90.4,1);
INSERT INTO PAL_DATA_TBL VALUES(13,'ATTR3',900,1);
INSERT INTO PAL_DATA_TBL VALUES(13,'ATTR4','A',1);
INSERT INTO PAL_DATA_TBL VALUES(13,'LABEL',3,1);
INSERT INTO PAL_DATA_TBL VALUES(14,'ATTR1',9.2,1);
INSERT INTO PAL_DATA_TBL VALUES(14,'ATTR2',90.3,1);
INSERT INTO PAL_DATA_TBL VALUES(14,'ATTR3',900,1);
INSERT INTO PAL_DATA_TBL VALUES(14,'ATTR4','A',1);
INSERT INTO PAL_DATA_TBL VALUES(14,'LABEL',3,1);
DROP TABLE PAL_PARAMETER_TBL;
CREATE  COLUMN TABLE PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (100)
);
INSERT INTO PAL_PARAMETER_TBL VALUES ('FUNCTION', NULL, NULL, 'M_LOGR');
INSERT INTO PAL_PARAMETER_TBL VALUES ('PARTITION_METHOD', 2, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('HANDLE_MISSING_VALUE', 1, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('PARTITION_STRATIFIED_VARIABLE', NULL,
NULL, 'LABEL');
INSERT INTO PAL_PARAMETER_TBL VALUES ('PARTITION_TRAINING_PERCENT', NULL, 0.7,
NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('OUTPUT_PARTITION_RESULT', 1, NULL,
NULL);
DROP TABLE PAL_MODEL_TBL;
CREATE COLUMN TABLE PAL_MODEL_TBL (
    "ROW_INDEX" INTEGER,
    "TREE_INDEX" INTEGER,
    "MODEL_CONTENT" NVARCHAR(5000)
);
DROP TABLE PAL_IMP_TBL;
CREATE COLUMN TABLE PAL_IMP_TBL (
    "VARIABLE_NAME" NVARCHAR(256),
    "IMPORTANCE" DOUBLE
);
DROP TABLE PAL_STAT_TBL;
CREATE COLUMN TABLE PAL_STAT_TBL (
    "STAT_NAME" NVARCHAR(256),
    "STAT_VALUE" NVARCHAR(1000),
    "CLASS_NAME" NVARCHAR(256)
);
DROP TABLE PAL_OPT_PARAM_TBL;
CREATE  COLUMN TABLE PAL_OPT_PARAM_TBL (
    "PARAM_NAME" VARCHAR (100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (100)
);
DROP TABLE PAL_CMATRIX_TBL;
CREATE COLUMN TABLE PAL_CMATRIX_TBL (
    "ACTUAL_CLASS" NVARCHAR(256),
    "PREDICTED_CLASS" NVARCHAR(256),
    "COUNT" INTEGER
);
DROP TABLE PAL_METRICS_TBL;
CREATE COLUMN TABLE PAL_METRICS_TBL (
    "NAME" NVARCHAR(256),
    "X" DOUBLE,
    "Y" DOUBLE
);
DROP TABLE PAL_PARTITION_TBL;
CREATE COLUMN TABLE PAL_PARTITION_TBL (
    "ID" INTEGER,
    "TYPE" INTEGER
```

```
);
DO
BEGIN
    lt_meta_data = SELECT * FROM PAL_META_DATA_TBL;
    lt_data = SELECT * FROM PAL_DATA_TBL;
    lt_ctrl = SELECT * FROM PAL_PARAMETER_TBL;
    CALL _SYS_AFL.PAL_UNIFIED_CLASSIFICATION_PIVOT
(:lt_meta_data, :lt_data, :lt_ctrl, lt_model, lt_imp,lt_stat,
lt_opt,lt_cm,lt_metrics,lt_partition,lt_ph1);
    INSERT INTO PAL_MODEL_TBL SELECT * FROM :lt_model;
    INSERT INTO PAL_IMP_TBL SELECT * FROM :lt_imp;
    INSERT INTO PAL_STAT_TBL SELECT * FROM :lt_stat;
    INSERT INTO PAL_OPT_PARAM_TBL SELECT * FROM :lt_opt;
    INSERT INTO PAL_CMATRIX_TBL SELECT * FROM :lt_cm;
    INSERT INTO PAL_METRICS_TBL SELECT * FROM :lt_metrics;
    INSERT INTO PAL_PARTITION_TBL SELECT * FROM :lt_partition;
END;
SELECT * FROM PAL_MODEL_TBL;
```

**Example 2: Multi-Class Logistic Regression (Prediction)**

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_PREDICT_DATA_TBL;
CREATE COLUMN TABLE PAL_PREDICT_DATA_TBL (
    DATA_ID INTEGER,
    VARIABLE_NAME NVARCHAR(100),
    VAR_VALUE NVARCHAR(100)
);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(0,'ATTR1','1');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(0,'ATTR2','10');
--INSERT INTO PAL_PREDICT_DATA_TBL VALUES(0,'ATTR3','100');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(0,'ATTR4','A');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(1,'ATTR1',1.1);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(1,'ATTR2',10.1);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(1,'ATTR3',100.1);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(1,'ATTR4','A');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(2,'ATTR1',1.2);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(2,'ATTR2',10.2);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(2,'ATTR3',100);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(2,'ATTR4','A');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(3,'ATTR1',1.3);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(3,'ATTR2',10.4);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(3,'ATTR3',100);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(3,'ATTR4','A');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(4,'ATTR1',4);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(4,'ATTR2',40);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(4,'ATTR3',400);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(4,'ATTR4','AB');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(5,'ATTR1',9.2);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(5,'ATTR2',90.2);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(5,'ATTR3',900);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES(5,'ATTR4','B');
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (100)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('VERBOSE', 0, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('HANDLE_MISSING_VALUE', 1, NULL, NULL);
CALL _SYS_AFL.PAL_UNIFIED_CLASSIFICATION_PREDICT_PIVOT(PAL_PREDICT_DATA_TBL,
PAL_MODEL_TBL, #PAL_PARAMETER_TBL, ?, ?);
```

**Example 3: Multi-Class Logistic Regression (Scoring)**

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_SCORING_DATA_TBL;
CREATE COLUMN TABLE PAL_SCORING_DATA_TBL (
    DATA_ID INTEGER,
    VARIABLE_NAME NVARCHAR(100),
    VAR_VALUE NVARCHAR(100)
);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(0,'ATTR1','1');
INSERT INTO PAL_SCORING_DATA_TBL VALUES(0,'ATTR2','10');
INSERT INTO PAL_SCORING_DATA_TBL VALUES(0,'ATTR3','100');
--INSERT INTO PAL_SCORING_DATA_TBL VALUES(0,'ATTR4','A');
INSERT INTO PAL_SCORING_DATA_TBL VALUES(0,'LABEL','1');
INSERT INTO PAL_SCORING_DATA_TBL VALUES(1,'ATTR1',1.1);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(1,'ATTR2',10.1);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(1,'ATTR3',100.1);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(1,'ATTR4','A');
INSERT INTO PAL_SCORING_DATA_TBL VALUES(1,'LABEL','1');
INSERT INTO PAL_SCORING_DATA_TBL VALUES(2,'ATTR1',1.2);
--INSERT INTO PAL_SCORING_DATA_TBL VALUES(2,'ATTR2',10.2);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(2,'ATTR3',100);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(2,'ATTR4','A');
INSERT INTO PAL_SCORING_DATA_TBL VALUES(2,'LABEL','1');
INSERT INTO PAL_SCORING_DATA_TBL VALUES(3,'ATTR1',1.3);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(3,'ATTR2',10.4);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(3,'ATTR3',100);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(3,'ATTR4','A');
INSERT INTO PAL_SCORING_DATA_TBL VALUES(3,'LABEL','1');
--INSERT INTO PAL_SCORING_DATA_TBL VALUES(4,'ATTR1',4);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(4,'ATTR2',40);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(4,'ATTR3',400);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(4,'ATTR4','AB');
INSERT INTO PAL_SCORING_DATA_TBL VALUES(4,'LABEL','2');
INSERT INTO PAL_SCORING_DATA_TBL VALUES(5,'ATTR1',9.2);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(5,'ATTR2',90.2);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(5,'ATTR3',900);
INSERT INTO PAL_SCORING_DATA_TBL VALUES(5,'ATTR4','B');
INSERT INTO PAL_SCORING_DATA_TBL VALUES(5,'LABEL','3');
CALL _SYS_AFL.PAL_UNIFIED_CLASSIFICATION_SCORE_PIVOT(PAL_SCORING_DATA_TBL,
PAL_MODEL_TBL, #PAL_PARAMETER_TBL, ?, ?,?,?);
```

## 5.13.2.7 Parameter Tables

This topic lists all parameters for unified classification with pivoted input data.

### _SYS_AFL.PAL_UNIFIED_CLASSIFICATION_PIVOT

#### UNIFIED CLASSIFICATION

Mandatory

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION |
|---|---|---|---|
| FUNCTION | VARCHAR | No default value | • SVM: support vector machine<br>• RDT: random decision trees<br>• NB: naïve bayes<br>• MLP: multilayer perceptron<br>• M_LOGR:multi-class logistic regression<br>• LOGR:logistic regression<br>• HGBT:hybrid gradient boosting tree<br>• DT: decision trees |

Optional

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| PARTITION_METHOD | INTEGER | 0 | Defines the way to divide the dataset:<br>• 0: no partition<br>• 1: user defined partition<br>• 2: stratified partition | |
| PARTITION_RANDOM_SEED | INTEGER | 0 | Indicates the seed used to initialize the random number generator:<br>• 0: Uses the system time<br>• Not 0: Uses the specified seed | Valid only when PARTITION_METHOD is set to 2. |
| PARTITION_STRATIFIED_VARIABLE | VARCHAR | No default value | Indicates which column is used for stratification. | Valid only when PARTITION_METHOD is set to 2. |
| PARTITION_TRAINING_PERCENT | DOUBLE | 0.8 | The percentage of data used for training.<br>Value: 0 - 1 | |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| PARTITION_TRAINING_SIZE | INTEGER | No default value | Row size of data used for training.<br>Value: ≥ 0 | If both TRAINING_PERCENT and TRAINING_SIZE are specified, TRAINING_PERCENT takes precedence. |
| OUTPUT_PARTITION_RESULT | INTEGER | 0 | Controls whether to output the partition result.<br>• 0: No<br>• 1: Yes | Valid only when PARTITION_METHOD is 1 or 2. |
| NTILES | INTEGER | 1 or 20 | Used to control the population tiles in metrics output.<br>The validation value should be at least 1 and no larger than the row size of the validation data.<br>**Note**<br>For AUC, this parameter means the maximum tiles. | If the row size of data for metrics evaluation is less than 20, the default value is 1; otherwise it is 20. |
| BACKGROUND_SIZE | INTEGER | 0 | The row size of background data. | It should not be larger than the row size of train data.<br>Valid for the functions:<br>• Naive Bayes<br>• Support Vector Machine<br>• Multilayer Perceptron<br>• Multi-class Logistic Regression |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| BACKGROUND_SAMPLING_SEED | INTEGER | 0 | Specifies the seed for random number generator in the background samping.<br>• 0: Uses the current time (in second) as seed<br>• Others: Uses the specified value as seed | Valid for the functions:<br>• Naive Bayes<br>• Support Vector Machine<br>• Multilayer Perceptron<br>• Multi-class Logistic Regression |
| HANDLE_MISSING_VALUE | INTEGER | 0 | Whether to handle missing values:<br>• 0: No<br>• 1: Yes | |
| IMPUTATION_TYPE | INTEGER | 1 | Chooses the overall default imputation type for all columns:<br>• 0: NONE<br>• 1: MODE_MEAN<br>• 2: MODE_MEDIAN<br>• 3: MODE_ALLZERO<br>• 4: MODE_ALS<br>• 5: DELETE<br>NONE and DELETE: set all columns to imputation type NONE/DELETE.<br>MODE_MEAN, MODE_MEDIAN, MODE_ALLZERO, and MODE_ALS: set categorical columns to the first imputation type (the type before “”); set numerical columns to the second imputation type (the type after “”). ALLZERO means to fill all missing values in numerical columns with 0. | Only valid when HANDLE_MISSING_VALUE is 1. |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| `<column name>`_IMPUTATION_TYPE | INTEGER | No default value | Sets imputation type of a specified column. This value overwrites the default imputation type deduced by the IMPUTATION_TYPE parameter:<br>• 0: NONE<br>• 1: DELETE<br>• 100: MODE<br>• 101: SPECIFIED_CATEGORICAL_VALUE<br>• 200: MEAN<br>• 201: MEDIAN<br>• 203:SPECIFIED_NUMERICAL_VALUE<br>• 204: ALS<br>For type 101 (SPECIFIED_CATEGORICAL_VALUE), use the following syntax to specify a categorical value to be filled in a column V0:<br>INSERT INTO #PAL_PARAMETER_TBL VALUES('V0_IMPUTATION_TYPE', 101, NULL, '`<CATEGORICAL_VALUE>`');<br>For type 203 (SPECIFIED_NUMERICAL_VALUE), use the following syntax to specify a numerical value to be filled in a column V1:<br>INSERT INTO #PAL_PARAMETER_TBL VALUES('V1_IMPUTA- | Only valid when HANDLE_MISSING_VALUE is 1. |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| | | | TION_TYPE', 203, `<NUMERICAL_VALUE>`, NULL);<br><br>**Note**<br>A column of integer type can be set to type 101 (SPECIFIED_CATEGORICAL_VALUE) or type 203 (SPECIFIED_NUMERICAL_VALUE):<br>• when setting to type 101 (SPECIFIED_CATEGORICAL_VALUE), the string must contain a valid integer value, otherwise an error will be thrown;<br>• when setting to type 203 (SPECIFIED_NUMERICAL_VALUE), the double value will be converted to integer. | |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| ALS_FACTOR_NUMBER | INTEGER | 3 | Length of factor vectors, that is, the f in ALS model.<br>**Note**<br>ALS_FACTOR_NUMBER should be less than the number of numerical columns to get a meaningful result. | |
| ALS_REGULARIZATION | DOUBLE | 1e-2 | L2 regularization of the factors, that is, the λ in ALS model. | |
| ALS_MAX_ITERATION | INTEGER | 20 | Maximum number of iterations when training ALS model. | |
| ALS_SEED | INTEGER | 0 | Specifies the seed of the random number generator used in the training of ALS model:<br>• 0: Uses the current time as the seed<br>• Others: Uses the specified value as the seed. | |
| ALS_EXIT_THRESHOLD | DOUBLE | 0.0 | The training of ALS model exits if the value of cost function is not decreased more than the value of ALS_EXIT_THRESHOLD since the last check.<br>If ALS_EXIT_THRESHOLD is set to 0, there is no check and the training only exits on reaching the maximum number of iterations. Evaluations of cost function require additional calculations. You can set this parameter to 0 to avoid it. | |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| ALS_EXIT_INTERVAL | INTEGER | 5 | During the training of ALS model, it will calculate cost function and check the exit criteria every `<ALS_EXIT_INTERVAL>` iterations. Evaluations of cost function require additional calculations. | |
| ALS_LINEAR_SYSTEM_SOLVER | INTEGER | 0 | • 0: cholesky solver<br>• 1: cg solver (recommended when ALS_FACTOR_NUMBER is large) | |
| ALS_CG_MAX_ITERATION | INTEGER | 3 | Maximum number of iteration of cg solver. | |
| ALS_CENTERING | INTEGER | 1 | Whether to center the data by column before training ALS model.<br>• 0: no centering<br>• 1: centering. | |
| ALS_SCALING | INTEGER | 1 | Whether to scale the data by column before training ALS model.<br>• 0: no scaling<br>• 1: scaling | |

### DECISION TREES

Mandatory

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION |
|---|---|---|---|
| ALGORITHM | INTEGER | No default value | Specifies the algorithm used to grow a decision tree:<br>• 1: C45<br>• 2: CHAID<br>• 3: CART |

Optional

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| THREAD_RATIO | DOUBLE | -1 | The ratio of available threads for multi-thread tasks:<br>• 0: single thread<br>• 0–1: uses the specified percentage of available threads. PAL uses all available threads if the number is 1.0.<br>• -1: uses all of available threads. | |
| ALLOW_MISSING_LABEL | INTEGER | 1 | Specifies if missing target value is allowed:<br>• 0: Not allowed. If missing target presents, an error occurs.<br>• 1: Allowed. The datum with missing target is removed. | |
| PERCENTAGE | DOUBLE | 1.0 | Specifies the percentage of the training data used to build the tree model.<br>For example, if you set this parameter to 0.7, then 70% of the training data are used to build the tree model and 30% are used to prune the tree model. | |
| MIN_RECORDS_PARENT | INTEGER | 2 | Specifies the stop condition. If the number of records in one node is less than the specified value, the algorithm stops splitting. | |
| MIN_RECORDS_LEAF | INTEGER | 1 | Promises the minimum number of records in a leaf. | |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| MAX_DEPTH | INTEGER | Unlimited | Specifies the stop condition. If the depth of the tree model is greater than the specified value, the algorithm stops splitting. | |
| SPLIT_THRESHOLD | DOUBLE | 1e-5 for C45 and CART<br>0.05 for CHAID | Specifies the stop condition for a node:<br>• C45: The information gain ratio of the best split is less than this value.<br>• CHAID: The p-value of the best split is greater than or equal to this value.<br>• CART: The reduction of Gini index or relative MSE of the best split is less than this value.<br>The smaller the SPLIT_THRESHOLD value is, the larger a C45 or CART tree grows. On the contrary, CHAID grows a larger tree with larger SPLIT_THRESHOLD value. | |
| `<target_value_name>`_PRIOR_ | DOUBLE | Detected from input data | Specifies the prior probability of each class label. | Valid only for classification. |
| DISCRETIZATION_TYPE | INTEGER | 0 | Specifies the strategy for discretizing continuous attributes:<br>• 0: MDLPC<br>• 1: Equal Frequency | Valid only for C45 and CHAID. |
| `<column_name>`_BIN_ | INTEGER | 10 | Specifies the number of bins for discretisation. | Only valid when DISCRETIZATION_TYPE is 1. |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| MAX_BRANCH | INTEGER | 10 | Specifies the maximum number of branches. | Valid only for CHAID. |
| MERGE_THRESHOLD | DOUBLE | 0.05 | Specifies the merge condition for CHAID. If the metric value is greater than or equal to the specified value, the algorithm merges two branches. | Valid only for CHAID. |
| USE_SURROGATE | INTEGER | 1 | Indicates whether to use surrogate split when NULL values are encountered:<br>• 0: Not to use surrogate split.<br>• 1: Use surrogate split. | Valid only for CART. |
| MODEL_FORMAT | INTEGER | 1 | Specifies the tree model format for store:<br>• 1: JSON<br>• 2: PMML | |

Parameter search and selection

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| RESAMPLING_METHOD | VARCHAR, or NVARCHAR | No default | The resampling method for model evaluation or parameter search:<br>• cv: cross validation<br>• stratified_cv: stratified cross validation<br>• bootstrap: bootstrap method<br>• stratified_bootstrap: stratified bootstrap method | |
| FOLD_NUM | INTEGER | No default | The fold number for cross validation. | Valid only for cross validation; mandatory for cross validation. |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| REPEAT_TIMES | INTEGER | 1 | The number of repeated times for model evaluation or parameter search. | |
| EVALUATION_METRIC | VARCHAR, or NVARCHAR | ERROR_RATE | The evaluation metric, candidates are:<br>• RMSE<br>• MAE<br>• ERROR_RATE<br>• NLL<br>• AUC | Multi-classification does not use RMSE or MAE |
| TIME_OUT | INTEGER | 0 | The time allocated (seconds) for program running. 0 indicates unlimited. | |
| PARAM_SEARCH_STRATEGY | VARCHAR, or NVARCHAR | No default | The search strategy for parameters:<br>• random: random search<br>• grid: grid search | If not specified, only model evaluation is carried out. |
| RANDOM_SEARCH_TIMES | INTEGER | No default | Specifies the number of search times for random search. | Mandatory for random search. |
| PROGRESS_INDICATOR_ID | NVARCHAR | No default | Sets an ID of progress indicator for model evaluation or parameter selection. | No progress indicator is active if no value is provided. |
| DISCRETIZATION_TYPE_VALUES | VARCHAR, or NVARCHAR | No default | The discrete set for the DISCRETIZATION_TYPE candidate. | |
| DISCRETIZATION_TYPE_RANGE | VARCHAR, or NVARCHAR | No default | The range for the DISCRETIZATION_TYPE candidate. | |
| MIN_RECORDS_LEAF_VALUES | VARCHAR, or NVARCHAR | No default | The discrete set for the MIN_RECORDS_OF_LEAF candidate. | |
| MIN_RECORDS_LEAF_RANGE | VARCHAR, or NVARCHAR | No default | The range for the MIN_RECORDS_OF_LEAF candidate. | |
| MIN_RECORDS_PARENT_VALUES | VARCHAR, or NVARCHAR | No default | The discrete set for the MIN_RECORDS_PARENT candidate. | |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| MIN_RECORDS_PARENT_RANGE | VARCHAR, or NVARCHAR | No default | The range for the MIN_RECORDS_ PARENT candidate. | |
| MAX_DEPTH_VALUES | VARCHAR, or NVARCHAR | No default | The discrete set for the MAX_DEPTH candidate. | |
| MAX_DEPTH_RANGE | VARCHAR, or NVARCHAR | No default | The range for the MAX_DEPTH candidate. | |
| SPLIT_THRESHOLD_VALUES | VARCHAR, or NVARCHAR | No default | The discrete set for the SPLIT_THRESHOLD candidate. | |
| SPLIT_THRESHOLD_RANGE | VARCHAR, or NVARCHAR | No default | The range for the SPLIT_THRESHOLD candidate. | |
| MAX_BRANCH_VALUES | VARCHAR, or NVARCHAR | No default | The discrete set for the MAX_BRANCH candidate. | |
| MAX_BRANCH_RANGE | VARCHAR, or NVARCHAR | No default | The range for the MAX_BRANCH candidate. | |
| MERGE_THRESHOLD_VALUES | VARCHAR, or NVARCHAR | No default | The discrete set for the MERGE_THRESHOLD candidate. | |
| MERGE_THRESHOLD_RANGE | VARCHAR, or NVARCHAR | No default | The range for the MERGE_THRESHOLD candidate. | |

### HYBRID GRADIENT BOOSTING TREE

Optional

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| THREAD_RATIO | DOUBLE | -1 | The ratio of available threads.<br>• 0: single thread<br>• 0–1: uses the specified percentage of available threads. PAL uses all available threads if the number is 1.0.<br>• -1: uses all of available threads. | |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| SEED | INTEGER | 0 | The seed for random number generating.<br>• 0: current time<br>• Others: the seed | |
| SPLIT_METHOD | VARCHAR or NVARCHAR | histogram | The method to find split point for numeric feature.<br>• exact: the exact method, trying all possible points<br>• sketch: the sketch method, accounting for the distribution of the sum of hessian<br>• sampling: samples the split point randomly<br>• histogram:builds histogram upon data and uses it as the split point | The exact method comparably has the highest test accuracy, but costs more time. The other three methods have relative higher computational efficiency but might lead to lower test accuracy, and are considered to be adopted as the training data set is huge.<br>Valid only for numeric features. |
| SKETCH_ESP | DOUBLE | 0.1 | The epsilon of the sketch method. It indicates that the sum of hessian between two split points is not larger than this value. That is, the number of bins is approximately 1/eps. | The less is this value, the more split points are tried. |
| N_ESTIMATORS | INTEGER | 10 | Total iteration number, which is equivalent to the number of trees in the final model. | |
| MAX_DEPTH | INTEGER | 6 | The maximum depth of each tree. | |
| MIN_SAMPLES_LEAF | INTEGER | 1 | The minimum number of data in a leaf. | |
| LEARNING_RATE | DOUBLE | 0.3 | Learning rate of each iteration.<br>Range: (0, 1] | |
| SPLIT_THRESHOLD | DOUBLE | 0 | The minimum loss change value to make a split in tree growth (gamma in the equation). | |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| MIN_SAMPLES_WEIGHT_LEAF | DOUBLE | 1 | The minimum summation of sample weights (hessian) in the leaf node. | |
| MAX_W_IN_SPLIT | DOUBLE | 0 | The maximum weight constraint assigned to each tree node. 0 for no constraint. | |
| SUBSAMPLE | DOUBLE | 1 | The sample rate of row (data points).<br>Range: (0, 1] | |
| COL_SUBSAMPLE_SPLIT | DOUBLE | 1 | The sample rate of feature set in each split.<br>Range: (0, 1] | |
| COL_SUBSAMPLE_TREE | DOUBLE | 1 | The sample rate of feature set in each tree growth.<br>Range: (0, 1] | |
| LAMB | DOUBLE | 1 | L2 regularization.<br>Range: [0, 1] | |
| ALPHA | DOUBLE | 0 | L1 regularization.<br>Range: [0, 1] | |
| BASE_SCORE | DOUBLE | 0.5 | Initial prediction score of all instances.<br>Global bias for sufficient number of iterations.<br>Range: (0, 1), only for binary classification. | |
| COMPRESSION | INTEGER | 0 | Indicates whether result model should be compressed:<br>• 0. no;<br>• 1. yes. | |
| MAX_BITS | INTEGER | 12 | Maximum number of bits to quantize continuous features. Equivalent to use $2^{MAX_BITS}$ bins.<br>Must be less than 31. | Valid only when the value of COMPRESSION is 1. |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| FEATURE_GROUPING | INTEGER | 1 | Applies feature grouping by grouping sparse features that only contain one significant value in each row.<br>• 0: no<br>• 1: yes | |
| TOLERANT_RATE | DOUBLE | 0.0001 | While applying feature grouping, features are still merged when there are rows containing more than one significant value. This parameter specifies the rate of such rows that are allowed. | Only available when FEATURE_GROUPING is set to 1. |
| DEFAULT_MISSING_DIRECTION | INTEGER | 1 | Defines the default direction where the missing value goes to while splitting.<br>• 0: left<br>• 1: right | |
| REPLACE_MISSING | INTEGER | 1 | Replaces the missing value by another value in the feature.<br>If it is a continuous feature, the value is the mean value;<br>If it is a categorical feature, the value is the most frequent one.<br>• 0: no<br>• 1: yes | |
| START_FROM_AVERAGE | INTEGER | 0 | Indicates whether to adopt the prior distribution as the initial point. To be specific, use average value if it is a regression problem.<br>• 0: no;<br>• 1: yes. | BASE_SCORE is ignored if this parameter is set to 1. |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| OBJ_FUNC | INTEGER | • 0 for regression problem<br>• 5 for binary classification<br>• 7 for multiple classification | For regression:<br>• 0: Squared error<br>• 1: Squared log error<br>• 2: Pseudo Huber error<br>• 3: Gamma<br>• 4: Tweedie<br>For binary classification:<br>• 5: Logistic<br>• 6: Hinge<br>For multiple classification:<br>• 7: Softmax | |
| TWEEDIE_POWER | DOUBLE | 1.5 | For Tweedie object function; must be in range [1.0, 2.0]. | Only valid when OBJ_FUNC equals 4. |
| VALIDATION_SET_RATE | DOUBLE | 0.0 | Specifies the rate of validation set that is sampled from the data set. If 0.0 is set, then no early stop will be applied. | |
| STRATIFIED_VALIDATION_SET | INTEGER | 0 | Indicates weather or not to apply the stratified method while sampling validation set. | Valid only when VALIDATION_SET_RATE is greater than 0. |
| TOLERANT_ITER_NUM | INTEGER | 10 | Indicates how many consecutive deteriorated iterations are observed before applying early stop. | Valid only when VALIDATION_SET_RATE is greater than 0. |

Parameter search and selection

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| RESAMPLING_METHOD | NVRCHAR | No default value | Specifies the resampling method for model evaluation or parameter selection.<br>• cv<br>• cv_sha<br>• cv_hyperband<br>• stratified_cv<br>• stratified_cv_sha<br>• stratified_cv_hyperband<br>• bootstrap<br>• bootstrap_sha<br>• bootstrap_hyperband<br>• stratified_bootstrap<br>• stratified_bootstrap_sha<br>• stratified_bootstrap_hyperband<br>stratified_* can only apply to classification algorithms. If no value is specified for this parameter, neither model evaluation nor parameter selection is activated. | Options like stratified_* can only be applied to classification algorithms. Options like *_sha and *_hyperband can only be used in parameter selection, can not be used in model evaluation. |
| EVALUATION_METRIC | VARCHAR or NVARCHAR | ERROR_RATE (for classification), | The metrics used for cross-validation. | Classification: RMSE, MAE, NLL, ERROR_RATE, AUC<br>It is mandatory if RESAMPLING_METHOD is set. |
| REF_METRIC | VARCHAR or NVARCHAR | ERROR_RATE (for binary classification), | The metric(s) for reference. Could be of multi-lines, that is, more than one REF_METRIC. | The metric that is identical to EVALUATION_METRIC is omitted |
| FOLD_NUM | INTEGER | No default value | Specifies the fold number for the cross validation method. | Mandatory and valid only when RESAMPLING_METHOD is set to cv or stratified_cv. |
| REPEAT_TIMES | INTEGER | 1 | Specifies the number of repeat times for resampling. | |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| PARAM_SEARCH_STRATEGY | NVRCHAR | No default value | Specifies the parameter search method:<br>• grid<br>• random | |
| RANDOM_SEARCH_TIMES | INTEGER | No default value | Specifies the number of times to randomly select candidate parameters for selection. | Mandatory and valid when PARAM_SEARCH_STRATEGY is set to random. |
| REDUCTION_RATE | DOUBLE | 3.0 | Specifies reduction rate in SHA or Hyperband method. | Only valid when RESAMPLING_METHOD is specified as *_sha or *_hyperband. |
| RESOURCE | NVARCHAR | No default value | Specifies resource definition for SHA or Hyperband method.<br>• Not specified, size of data set is used as resource by default.<br>• 'ITER_NUM' | Only valid when RESAMPLING_METHOD is specified as *_sha or *_hyperband. |
| MIN_RESOURCE_RATE | DOUBLE | 0.0 or 1/MAX_RESOURCE | Specifies the minimum resource rate that should be used in SHA or Hyperband iteration. | Only valid when RESAMPLING_METHOD is specified as *_sha or *_hyperband. Default value 0.0 for default RESOURCE and default value 1/MAX_RESOURCE for RESOURCE = ITER_NUM |
| MAX_RESOURCE | INTEGER | No default value | Specifies the maximum number of ITER_NUM that should be used in SHA or Hyperband method. | Only valid when RESAMPLING_METHOD is specified as *_sha or *_hyperband. It is valid and mandatory if RESOURCE is set. |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| AGGRESSIVE_ELIMINATION | INTEGER | 0 | Specifies whether to apply aggressive elimination while using SHA method. Aggressive elimination happens when the data size and parameters size to be searched does not match and there are still bunch of parameters to be searched while data size reaches its upper limits. If aggressive elimination is applied, lower bound of limit of data size will be used multiple times first to reduce number of parameters.<br>• 0: does not apply aggressive elimination<br>• 1: applies aggressive elimination | Only valid when RESAMPLING_METHOD is specified as *_sha. |
| TIMEOUT | INTEGER | 0 | Specifies maximum running time for model evaluation or parameter selection, in seconds. No timeout when 0 is specified. | |
| PROGRESS_INDICATOR_ID | NVRCHAR | No default value | Sets an ID of progress indicator for model evaluation or parameter selection. No progress indicator is active if no value is provided. | |
| N_ESTIMATORS_RANGE | VARCHAR or NVARCHAR | Null | | |
| N_ESTIMATORS_VALUES | VARCHAR or NVARCHAR | Null | | |
| MAX_DEPTH_RANGE | VARCHAR or NVARCHAR | Null | | |
| MAX_DEPTH_VALUES | VARCHAR or NVARCHAR | Null | | |
| MIN_SAMPLES_LEAF_RANGE | VARCHAR or NVARCHAR | Null | | |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| MIN_SAMPLES_LEAF_VALUES | VARCHAR or NVARCHAR | Null | | |
| LEARINING_RATE_RANGE | VARCHAR or NVARCHAR | Null | | |
| LEARINING_RATE_VALUES | VARCHAR or NVARCHAR | Null | | |
| SPLIT_THRESHOLD_RANGE | VARCHAR or NVARCHAR | Null | | |
| SPLIT_THRESHOLD_VALUE | VARCHAR or NVARCHAR | Null | | |
| MIN_SAMPLES_WEIGHT_LEAF_RANGE | VARCHAR or NVARCHAR | Null | | |
| MIN_SAMPLES_WEIGHT_LEAF_VALUES | VARCHAR or NVARCHAR | Null | | |
| MAX_W_IN_SPLIT_RANGE | VARCHAR or NVARCHAR | Null | | |
| MAX_W_IN_SPLIT_VALUES | VARCHAR or NVARCHAR | Null | | |
| SUBSAMPLE_RANGE | VARCHAR or NVARCHAR | Null | | |
| SUBSAMPLE_VALUES | VARCHAR or NVARCHAR | Null | | |
| COL_SUBSAMPLE_SPLIT_RANGE | VARCHAR or NVARCHAR | Null | | |
| COL_SUBSAMPLE_SPLIT_VALUES | VARCHAR or NVARCHAR | Null | | |
| COL_SUBSAMPLE_TREE_RANGE | VARCHAR or NVARCHAR | Null | | |
| COL_SUBSAMPLE_TREE_VALUES | VARCHAR or NVARCHAR | Null | | |
| LAMB_RANGE | VARCHAR or NVARCHAR | Null | | |
| LAMB_VALUES | VARCHAR or NVARCHAR | Null | | |
| ALPHA_RANGE | VARCHAR or NVARCHAR | Null | | |
| ALPHA_VALUES | VARCHAR or NVARCHAR | Null | | |
| BASE_SCORE_RANGE | VARCHAR or NVARCHAR | Null | | |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| BASE_SCORE_VALUES | VARCHAR or NVARCHAR | Null | | |

### LOGISTIC REGRESSION

Mandatory

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION |
|---|---|---|---|
| CLASS_MAP0 | VARCHAR | No default value | Specifies the dependent variable value which will be mapped to 0.<br>Only valid when the column type of dependent variables is VARCHAR or NVARCHAR. |
| CLASS_MAP1 | VARCHAR | No default value | Specifies the dependent variable value which will be mapped to 1.<br>Only valid when the column type of dependent variables is VARCHAR or NVARCHAR. |

Optional

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| METHOD | INTEGER | -1 | • -1: Auto selection, based on given data and parameters<br>• 0: Newton iteration method<br>• 2: Cyclical coordinate descent method to fit elastic net regularized logistic regression.<br>• 3: LBFGS method (recommended when having many independent variables).<br>• 4: Stochastic gradient descent method (recommended when dealing with very large dataset).<br>• 6: Proximal gradient descent method to fit elastic net regularized logistic regression. | |
| THREAD_RATIO | DOUBLE | -1 | The ratio of available threads.<br>• 0: single thread<br>• 0–1: uses the specified percentage of available threads. PAL uses all available threads if the number is 1.0.<br>• -1: uses all of available threads. | |
| MAX_PASS_NUMBER | INTEGER | 1 | The maximum number of passes over the data. | Only valid when METHOD is 4. |
| SGD_BATCH_NUMBER | INTEGER | 1 | The batch number of Stochastic gradient descent. | Only valid when METHOD is 4. |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| PRECOMPUTE | INTEGER | 1 | Whether to compute Gram matrix:<br>• 0: No<br>• 1: Yes | Only valid when METHOD is 2. |
| LAMB | DOUBLE | 0 | Penalized weight. The value should be equal to or greater than 0. | Only valid when METHOD is 0, 2, 3, or 6.<br>**Note**<br>If you want to solve Ridge penalty problem for METHOD 0, set ENET_ALPHA to 0.0. |
| ALPHA | DOUBLE | 1 | The elastic net mixing parameter. The value range is between 0 and 1 inclusively.<br>• 0: Ridge penalty<br>• 1: LASSO penalty | Only valid when METHOD is 0, 2, 3, or 6. |
| HANDLE_MISSING | INTEGER | 1 | Whether to handle missing values:<br>• 0: No<br>• 1: Yes | |
| MAX_ITER | INTEGER | 100, 1000, or 100000 | When METHOD is 0 or 3, the convex optimizer may return suboptimal results after the maximum number of iterations. When METHOD is 2, if convergence is not reached after the maximum number of passes over training data, an error will be generated. | When METHOD is 2, MAX_ITERATION defaults to 100000; when METHOD is 6, it defaults to 1000; otherwise it defaults to 100. |
| STANDARDIZE | INTEGER | 1 | Controls whether to standardize the data to have zero mean and unit variance.<br>• 0: No<br>• 1: Yes | |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| TOL | DOUBLE | 1.0e-6 or 1.0e-7 | Convergence threshold for exiting iterations. | When METHOD is 2, TOL defaults to 1.0E-7; Otherwise it defaults to 1.0E-6. |
| EPSILON | DOUBLE | 1.0e-5 or 1.0e-6 | The parameter determines the accuracy with which the solution is to be found.<br>When METHOD is 3, the condition is: ||g|| < EPSILON * max {1, ||x||}, where g is gradient of objective function, x is solve of current iteration, and ||·|| denotes the L2 norm.<br>When METHOD is 0, the condition is: ||x-x'|| < EPSILON * sqrt(n), where x is the solve of current iteration, x' is the previous iteration, and n is the number of features. | When METHOD is 0, EPSILON defaults to 1.0e-6. When METHOD is 3, EPSILON defaults to 1.0e-5. |
| LBFGS_M | INTEGER | 6 | Number of past updates needed to be kept. | Only available when METHOD is 3. |
| STAT_INF | INTEGER | 0 | • 0: Does not proceed statistical inference<br>• 1: Proceeds statistical inference | |
| SELECTED_FEATURES | VARCHAR | No default value | A string to specify the features that will be processed. The pattern is "X1,...,Xn", where Xi is the corresponding column name in the data table. If this parameter is not specified, all the features will be processed. | Only used when you need to indicate the needed features. |

Parameter search and selection

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| RESAMPLING_METHOD | NVARCHAR | No default value | Specifies the resampling method for model evaluation or parameter selection.<br>• cv<br>• cv_sha<br>• cv_hyperband<br>• bootstrap<br>• bootstrap_sha<br>• bootstrap_hyperband<br>• stratified_cv<br>• stratified_cv_sha<br>• stratified_cv_hyperband<br>• stratified_bootstrap<br>• stratified_bootstrap_sha<br>• stratified_bootstrap_hyperband<br>If no value is specified for this parameter, neither model evaluation nor parameter selection is activated. | |
| RESOURCE | NVARCHAR | No default value | Resource type used in successive-halving and hyperband algorithm for parameter selection.<br>• MAX_ITERATION<br>• MAX_PASS_NUMBER | Mandatory and valid only when RESAMPLING_METHOD parameter has values with sha or hyperband suffix. The value MAX_PASS_NUMBER is only valid when actual used method is Stochastic Gradient Descent; MAX_ITERATION is used for other methods. When user set METHOD as -1 (Auto Selection), the value to use depends on actual used method; For current implementation users should always choose MAX_ITERATION in this case. |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| MAX_RESOURCE | INTEGER | No default value | Maximum allowed resource budget for single hyper-parameter candidate. Valid value must be larger than 0. | Mandatory and valid only when RESOURCE parameter is set. |
| MIN_RESOURCE_RATE | DOUBLE | 0 | The rate between minimum allowed resource and maximum allowed resource. Valid value is between [0, 1). | Mandatory and valid only when RESOURCE parameter is set. |
| REDUCTION_RATE | DOUBLE | 3.0 | This parameter specifies the reduction rate of available size of hyper-parameter candidates. For each round, the available parameter candidate size will be divided by value of this parameter. Thus valid value for this parameter must be larger than 1.0. | Valid only when RESOURCE parameter is set. |
| AGGRESSIVE_ELIMINATION | INTEGER | 0 | Boolean value indicates whether to perform aggressive elimination behavior for successive-halving algorithm. When set to 1, it will eliminate more parameter candidates than expected, which is defined by REDUCTION_RATE parameter; This can accelerate the run-time performance but may result in sub-optimal hyper-parameter candidate.<br>• 0: Does not perform aggressive elimination;<br>• 1: Performs aggressive elimiination | Valid only when RESAMPLING_METHOD parameter has value with suffix sha. |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| PS_VERBOSE | INTEGER | 1 | Boolean value indicates whether to output optimal hyper-parameter and all involved hyper-parameter candidates related evaluation statistics in Statistics output table.<br>• 0: Does not output evaluation statistics<br>• 1: Outputs evaluation statistics | Valid only when RESAMPLING_METHOD is set. |
| EVALUATION_METRIC | NVARCHAR | No default value | Specifies the evaluation metric for model evaluation or parameter selection.<br>• ACCURACY<br>• F1_SCORE<br>• AUC<br>• NLL | |
| FOLD_NUM | INTEGER | No default value | Specifies the fold number for the cross validation method. | Mandatory and valid only when RESAMPLING_METHOD is set to cv or stratified_cv. |
| REPEAT_TIMES | INTEGER | 1 | Specifies the number of repeat times for resampling. | |
| PARAM_SEARCH_STRATEGY | NVARCHAR | No default value | Specifies the method to activate parameter selection.<br>• grid<br>• random | |
| RANDOM_SEARCH_TIMES | INTEGER | No default value | Specifies the number of times to randomly select candidate parameters for selection. | Mandatory and valid when PARAM_SEARCH_STRATEGY is set to random. |
| SEED | INTEGER | 0 | Specifies the seed for random generation. Use system time when 0 is specified. | |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| TIMEOUT | INTEGER | 0 | Specifies maximum running time for model evaluation or parameter selection, in seconds. No timeout when 0 is specified. | |
| PROGRESS_INDICATOR_ID | NVARCHAR | No default value | Sets an ID of progress indicator for model evaluation or parameter selection. No progress indicator is active if no value is provided. | |
| LAMB_VALUES | NVARCHAR | No default value | Specifies values of LAMB to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| LAMB_RANGE | NVARCHAR | No default value | Specifies the range of LAMB to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| ALPHA_VALUES | NVARCHAR | No default value | Specifies values of ALPHA to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| ALPHA_RANGE | NVARCHAR | No default value | Specifies the range of ALPHA to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |

### MULTI-CLASS LOGISTIC REGRESSION

Optional

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| MAX_ITER | INTEGER | 100 | Maximum number of iterations of the optimization. | |
| MODEL_FORMAT | INTEGER | 1 | Specifies the tree model format for storage:<br>• 1: JSON<br>• 2: PMML | |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| SELECTED_FEATURES | VARCHAR | No default value | A string to specify the features that will be processed. The pattern is "X1,...,Xn", where Xi is the corresponding column name in the data table. If this parameter is not specified, all the features will be processed. | Only used when you need to indicate the needed features. |
| METHOD | INTEGER | 0 | • 0: lbfgs<br>• 1: cycle coordinate descent | |
| LAMB | DOUBLE | 0 | Penalized weight. The value should be equal to or greater than 0. | |
| ALPHA | DOUBLE | 1 | The elastic net mixing parameter. The value range is between 0 and 1 inclusively.<br>• 0: Ridge penalty<br>• 1: LASSO penalty | METHOD 0 only solve Ridge penalty problem. |
| TOL | DOUBLE | 1.00E-05 | Convergence threshold for exiting iterations. | |
| STANDARDIZE | INTEGER | 1 | Controls whether to standardize the data to have zero mean and unit variance.<br>• 0: No<br>• 1: Yes | |
| STAT_INF | INTEGER | 0 | Controls whether to proceed statistical inference.<br>• 0: Does not proceed statistical inference.<br>• 1: Proceeds statistical inference. | |

Parameter search and selection

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| RESAM-PLING_METHOD | NVARCHAR | No default value | Specifies the resampling method for model evaluation or parameter selection.<br>• cv<br>• cv_sha<br>• cv_hyperband<br>• bootstrap<br>• bootstrap_sha<br>• bootstrap_hyperband<br>• stratified_cv<br>• stratified_cv_sha<br>• stratified_cv_hyperband<br>• stratified_bootstrap<br>• stratified_bootstrap_sha<br>• stratified_bootstrap_hyperband<br>If no value is specified for this parameter, neither model evaluation nor parameter selection is activated. | |
| RESOURCE | NVARCHAR | No default value | Resource type used in successive-halving and hyperband algorithm for parameter selection.<br>• MAX_ITERATION | Mandatory and valid only when RESAM-PLING_METHOD parameter has values with sha or hyperband suffix. |
| MAX_RESOURCE | INTEGER | No default value | Maximum allowed resource budget for single hyper-parameter candidate. Valid value must be larger than 0. | Mandatory and valid only when RESOURCE parameter is set. |
| MIN_RE-SOURCE_RATE | DOUBLE | 0 | The rate between minimum allowed resource and maximum allowed resource. Valid value is between [0, 1). | Mandatory and valid only when RESOURCE parameter is set. |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| REDUCTION_RATE | DOUBLE | 3.0 | This parameter specifies the reduction rate of available size of hyper-parameter candidates. For each round, the available parameter candidate size will be divided by value of this parameter. Thus valid value for this parameter must be larger than 1.0. | Valid only when RESOURCE parameter is set. |
| AGGRESSIVE_ELIMINATION | INTEGER | 0 | Boolean value indicates whether to perform aggressive elimination behavior for successive-halving algorithm. When set to 1, it will eliminate more parameter candidates than expected, which is defined by REDUCTION_RATE parameter; This can accelerate the run-time performance but may result in sub-optimal hyper-parameter candidate.<br>• 0: Does not perform aggressive elimination;<br>• 1: Performs aggressive elimiination | Valid only when RESAMPLING_METHOD parameter has value with suffix sha. |
| PS_VERBOSE | INTEGER | 1 | Boolean value indicates whether output optimal hyper-parameter and all involved hyper-parameter candidates related evaluation statistics in Statistics output table.<br>• 0: Does not output evaluation statistics<br>• 1: Outputs evaluation statistics | Valid only when RESAMPLING_METHOD is set. |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| EVALUATION_METRIC | NVARCHAR | No default value | Specifies the evaluation metric for model evaluation or parameter selection.<br>• ACCURACY<br>• F1_SCORE<br>• AUC<br>• NLL | Only ordinal regression use ERROR_RATE. |
| FOLD_NUM | INTEGER | No default value | Specifies the fold number for a cross validation method. | Mandatory and valid only when RESAMPLING_METHOD is set to cv or stratified_cv. |
| REPEAT_TIMES | INTEGER | 1 | Specifies the number of repeat times for resampling. | |
| PARAM_SEARCH_STRATEGY | NVARCHAR | No default value | Specifies the method to activate parameter selection.<br>• grid<br>• random | This parameter is only valid when RESAMPLING_METHOD does not have value with suffix hyperband; Otherwise, this parameter is always considered to have value random. |
| RANDOM_SEARCH_TIMES | INTEGER | No default value | Specifies the number of times to randomly select candidate parameters for selection. | Mandatory and valid when PARAM_SEARCH_STRATEGY is set to random. |
| SEED | INTEGER | 0 | Specifies the seed for random generation. Use system time when 0 is specified. | |
| TIMEOUT | INTEGER | 0 | Specifies maximum running time for model evaluation or parameter selection, in seconds. No timeout when 0 is specified. | |
| PROGRESS_INDICATOR_ID | NVARCHAR | No default value | Sets an ID of progress indicator for model evaluation or parameter selection. No progress indicator is active if no value is provided. | |
| LAMB_VALUES | NVARCHAR | No default value | Specifies values of LAMB to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| LAMB_RANGE | NVARCHAR | No default value | Specifies the range of LAMB to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| ALPHA_VALUES | NVARCHAR | No default value | Specifies values of ALPHA to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| ALPHA_RANGE | NVARCHAR | No default value | Specifies the range of ALPHA to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |

### MULTILAYER PERCEPTRON

Mandatory

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION |
|---|---|---|---|
| ACTIVATION | INTEGER | No default value | Active function code for the hidden layer.<br>• TANH = 1<br>• LINEAR = 2<br>• SIGMOID_ASYMMETRIC = 3<br>• SIGMOID_SYMMETRIC = 4<br>• GAUSSIAN_ASYMMETRIC = 5<br>• GAUSSIAN_SYMMETRIC = 6<br>• ELLIOT_ASYMMETRIC = 7<br>• ELLIOT_SYMMETRIC = 8<br>• SIN_ASYMMETRIC = 9<br>• SIN_SYMMETRIC = 10<br>• COS_ASYMMETRIC = 11<br>• COS_SYMMETRIC = 12<br>• ReLU = 13 |
| OUTPUT_ACTIVATION | INTEGER | No default value | Active function code for the output layer. The code is the same as ACTIVATION. |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION |
|---|---|---|---|
| HIDDEN_LAYER_SIZE | NVARCHAR | No default value | Specifies the size of each hidden layer in the format of `2, 3, 4`. The value 0 will be ignored, for example, `2, 0, 3` is equal to `2, 3`. |

Optional

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| MAX_ITERATION | INTEGER | 100 | Maximum number of iterations. | |
| TRAINING_STYLE | INTEGER | 1 | Specifies the training style:<br>• 0: Batch<br>• 1:Stochastic | |
| LEARNING_RATE | DOUBLE | None | Specifies the learning rate. | Mandatory and valid only when TRAINING_STYLE = 1. |
| MOMENTUM | DOUBLE | None | Specifies the momentum factor. | Mandatory and valid only when TRAINING_STYLE = 1. |
| BATCH_SIZE | INTEGER | 1 | Specifies the size of mini batch. | Valid only when TRAINING_STYLE = 1. |
| NORMALIZATION | INTEGER | 0 | Specifies the normalization type:<br>• 0: None<br>• 1: Z-transform<br>• 2: Scalar | |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| WEIGHT_INIT | INTEGER | 0 | Specifies the weight initial value:<br>• 0: All zeros<br>• 1: Normal distribution<br>• 2: Uniform distribution in range (0, 1)<br>• 3: Variance scale normal<br>• 4: Variance scale uniform | |
| THREAD_RATIO | DOUBLE | -1 | The ratio of available threads.<br>• 0: single thread<br>• 0–1: uses the specified percentage of available threads. PAL uses all available threads if the number is 1.0.<br>• -1: uses all of available threads. | |

Parameter search and selection

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| RESAMPLING_METHOD | NVARCHAR | No default value | Specifies the resampling method for model evaluation or parameter selection.<br>• cv<br>• cv_sha<br>• cv_hyperband<br>• stratified_cv<br>• stratified_cv_sha<br>• stratified_cv_hyperband<br>• bootstrap<br>• bootstrap_sha<br>• bootstrap_sha<br>• stratified_bootstrap<br>• stratified_bootstrap_sha<br>• stratified_bootstrap_hyperband<br>stratified_* can only apply to classification algorithms. If no value is specified for this parameter, neither model evaluation nor parameter selection is activated. | |
| EVALUATION_METRIC | NVARCHAR | No default value | Specifies the evaluation metric for model evaluation or parameter selection.<br>• ACCURACY<br>• F1_SCORE<br>• AUC_1VsRest<br>• AUC_pairwise | |
| FOLD_NUM | INTEGER | No default value | Specifies the fold number for the cross validation method. | Mandatory and valid only when RESAMPLING_METHOD is set to cv or stratified_cv. |
| REPEAT_TIMES | INTEGER | 1 | Specifies the number of repeat times for resampling. | |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| PARAM_SEARCH_STRATEGY | NVARCHAR | No default value | Specifies the method to activate parameter selection.<br>• grid<br>• random | When *_sha is specified in RESAMPLING_METHOD, PARAM_SEARCH_STRATEGY is mandatory. When *_hyperband is specified in RESAMPLING_METHOD, PARAM_SEARCH_STRATEGY is set to random whatever other value is provided. |
| RANDOM_SEARCH_TIMES | INTEGER | No default value | Specifies the number of times to randomly select candidate parameters for selection. | Mandatory and valid when PARAM_SEARCH_STRATEGY is set to random. |
| REDUCTION_RATE | DOUBLE | 3.0 | Specifies reduction rate in SHA or hyperband method. | Only valid when RESAMPLING_METHOD is specified as *_sha or *_hyberband. |
| AGGRESSIVE_ELIMINATION | INTEGER | | Specifies whether to apply aggressive elimination while using SHA method. It happens when the size of data and that of parameters to be searched does not match. And there are still bunch of parameters to be searched while data size reaches its upper limits. If aggressive elimination is applied, lower bound of limit of data size will be used multiple times first to reduce number of parameters.<br>• 0: not apply<br>• 1: apply | Only valid when RESAMPLING_METHOD is specified as *_sha. |
| SEED | INTEGER | 0 | Specifies the seed for random generation. Use system time when 0 is specified. | |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| TIMEOUT | INTEGER | 0 | Specifies maximum running time for model evaluation or parameter selection, in seconds. No timeout when 0 is specified. | |
| PROGRESS_INDICATOR_ID | NVARCHAR | No default value | Sets an ID of progress indicator for model evaluation or parameter selection. No progress indicator is active if no value is provided. | |
| ACTIVATION_OPTIONS | NVARCHAR | No default value | Specifies values of ACTIVATION to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| OUTPUT_ACTIVATION_OPTIONS | NVARCHAR | No default value | Specifies values of OUTPUT_ACTIVATION to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| HIDDEN_LAYER_SIZE_OPTIONS | NVARCHAR | No default value | Specifies values of HIDDEN_LAYER_SIZE to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| LEARNING_RATE_VALUES | NVARCHAR | No default value | Specifies values of LEARNING_RATE to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified and TRAINING_STYLE = 1. |
| LEARNING_RATE_RANGE | NVARCHAR | No default value | Specifies the range of LEARNING_RATE to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified and TRAINING_STYLE = 1. |
| MOMENTUM_VALUES | NVARCHAR | No default value | Specifies values of MOMENTUM to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified and TRAINING_STYLE = 1. |
| MOMENTUM_RANGE | NVARCHAR | No default value | Specifies the range of MOMENTUM to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified and TRAINING_STYLE = 1. |
| BATCH_SIZE_VALUES | NVARCHAR | No default value | Specifies values of BATCH_SIZE to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified and TRAINING_STYLE = 1. |
| BATCH_SIZE_RANGE | NVARCHAR | No default value | Specifies the range of BATCH_SIZE to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified and TRAINING_STYLE = 1. |

## NAÏVE BAYES

Optional

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| MODEL_FORMAT | INTEGER | 1 | Specifies the tree model format for storage:<br>• 1: JSON<br>• 2: PMML | |
| ALPHA | DOUBLE | 0 | Enables or disables Laplace smoothing.<br>• 0: Disables Laplace smoothing<br>• Positive value: Enables Laplace smoothing for discrete values<br>**Note**<br>The ALPHA value is only stored by JSON format models. If the PMML format is chosen, you may need to set the ALPHA value again in the predicting phase. | |
| DISCRETIZATION | INTEGER | 0 | • 0: Disables discretization.<br>• Other values: Uses supervised discretization to all the continuous attributes. | |
| THREAD_RATIO | DOUBLE | -1 | The ratio of available threads.<br>• 0: single thread<br>• 0–1: uses the specified percentage of available threads. PAL uses all available threads if the number is 1.0.<br>• -1: uses all of available threads. | |

Parameter search and selection

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| RESAMPLING_METHOD | NVARCHAR | No default value | Specifies resampling method for model evaluation or parameter selection.<br>• cv<br>• cv_sha<br>• cv_hyperband<br>• bootstrap<br>• bootstrap_sha<br>• bootstrap_hyperband<br>• stratified_cv<br>• stratified_cv_sha<br>• stratified_cv_hyperband<br>• stratified_bootstrap<br>• stratified_bootstrap_sha<br>• stratified_bootstrap_hyperband<br>If no value is specified for this parameter, neither model evaluation nor parameter selection is activated. | Options like stratified_* can only be applied to classification algorithms. Options like *_sha and *_hyperband can only be used in parameter selection, can not be used in model evaluation. |
| EVALUATION_METRIC | NVARCHAR | No default value | Specifies the evaluation metric for model evaluation or parameter selection.<br>Naive Bayes supports the following evaluation metrics:<br>• ACCURACY<br>• F1_SCORE<br>• AUC | Mandatory. |
| FOLD_NUM | INTEGER | No default value | Specifies the fold number for the cross validation method. | Mandatory and valid only when RESAMPLING_METHOD is set to cv or stratified_cv. |
| REPEAT_TIMES | INTEGER | 1 | Specifies the number of repeat times for resampling. | |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| PARAM_SEARCH_STRATEGY | NVARCHAR | No default value | Specifies the parameter search method:<br>• grid<br>• random | |
| RANDOM_SEARCH_TIMES | INTEGER | No default value | Specifies the number of times to randomly select candidate parameters for selection. | Mandatory and valid when PARAM_SEARCH_STRATEGY is set to random. |
| REDUCTION_RATE | DOUBLE | 3.0 | Specifies reduction rate in SHA or hyperband method. | Only valid when RESAMPLING_METHOD is specified as *_sha or *_hyberband. |
| AGGRESSIVE_ELIMINATION | INTEGER | 0 | Specifies whether to apply aggressive elimination while using SHA method. Aggressive elimination happens when the data size and parameters size to be searched does not match and there are still bunch of parameters to be searched while data size reaches its upper limits. If aggressive elimination is applied, lower bound of limit of data size will be used multiple times first to reduce number of parameters.<br>• 0: does not apply aggressive elimination<br>• 1: applies aggressive elimination | Only valid when RESAMPLING_METHOD is specified as *_sha. |
| SEED | INTEGER | 0 | Specifies the seed for random generation. Use system time when 0 is specified. | |
| TIMEOUT | INTEGER | 0 | Specifies maximum running time for model evaluation or parameter selection, in seconds. No timeout when 0 is specified. | |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| PROGRESS_INDICATOR_ID | NVARCHAR | No default value | Sets an ID of progress indicator for model evaluation or parameter selection. No progress indicator is active if no value is provided. | |
| ALPHA_RANGE | NVARCHAR | No default value | Specifies candidate ALPHA values for parameter selection. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| ALPHA_VALUES | NVARCHAR | No default value | Specifies the range for candidate ALPHA values for parameter selection. | Only valid when PARAM_SEARCH_STRATEGY is specified. |

### RANDOM DECISION TREES

Optional

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| THREAD_RATIO | DOUBLE | -1 | The ratio of available threads.<br>• 0: single thread<br>• 0–1: uses the specified percentage of available threads. PAL uses all available threads if the number is 1.0.<br>• -1: uses all of available threads. | |
| N_ESTIMATORS | INTEGER | 100 | Specifies the number of trees to grow. | |
| MAX_FEATURES | INTEGER | sqrt(p), where p is the number of valid independent variables | Specifies the number of randomly selected variables for splitting. | Should not larger than the valid number of features. |
| ALLOW_MISSING_LABEL | INTEGER | 1 | Specifies if missing target value is allowed.<br>• 0: Not allowed. An error occurs if missing dependent presents.<br>• 1: Allowed. The datum with missing dependent is removed. | |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| SEED | INTEGER | 0 | Specifies the seed for random number generator.<br>• 0: Uses the current time (in second) as seed<br>• Others: Uses the specified value as seed | |
| MIN_SAMPLES_LEAF | INTEGER | 1 for classification | Specifies the minimum number of records in a leaf. | |
| SPLIT_THRESHOLD | DOUBLE | 1.00E-05 | Specifies the stop condition: if the improvement value of the best split is less than this value, the tree stops growing. | |
| MAX_DEPTH | INTEGER | Unlimited | The maximum depth of a tree. | Cannot be larger than 56. |
| SAMPLE_FRACTION | DOUBLE | 1 | The fraction of data used for training.<br>Assume there are n pieces of data, SAMPLE_FRACTION is r, then n*r data is selected for training. | In the range of [0,1]. |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| STRATA | (INTEGER, DOUBLE), or (VARCHAR/NVARCHAR, DOUBLE) | No default value | Specifies the stratified sampling for classification.<br>This is a pair parameter, which means a parameter name corresponds to two values. The first value is the class label (INTEGER or VARCHAR/NVARCHAR), and the second one is the proportion of this class occupies in the sampling data. The class with a proportion less than 0 or larger than 1 will be ignored.<br>If some classes are absent, they share the remaining proportion equally.<br>If the summation of strata is less than 1, the remaining is shared by all classes equally. | Valid only for classification.<br>The summation of the proportion of strata should not be larger than 1.<br>If absent, bagging is used. |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| PRIORS | (INTEGER, DOUBLE), (VARCHAR/NVARCHAR, DOUBLE) | Determined by the proportion of every class in the training data | Specifies the prior probabilities for classification.<br>This is also a pair parameter. The first value is the class label (INTEGER or VARCHAR/NVARCHAR), and the second one is the prior probability of this class. The class with a prior probability less than 0 or larger than 1 will be ignored.<br>If some classes are absent, they share the remaining probability equally.<br>If the summation of priors is less than 1, the remaining is shared by all classes equally. | Valid only for classification.<br>The summation of the priors should not be larger than 1. |
| COMPRESSION | INTEGER | 0 | Specifies if the model is stored in compressed format<br>• 0: No<br>• 1: Yes | |
| MAX_BITS | INTEGER | 12 | The maximum number of bits to quantize continuous features. Equivalent to use 2^MAX_BITS bins. Must be less than 31. | Valid only when COMPRESSION equals to 1 |
| QUANTIZE_RATE | DOUBLE | 0.005 | Quantize a feature if the largest class frequency of which is less than QUANTIZE_RATE. | Valid only when COMPRESSION equals to 1 |
| FITTINGS_QUANTIZATION | INTEGER | 0 | Indicate whether to quantize fitting values.<br>• 0: No<br>• 1: Yes | Valid only when COMPRESSION equals to 1 |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| MODEL_FORMAT | INTEGER | 1 | Specifies the tree model format for storage:<br>• 1: JSON<br>• 2: PMML | If both MODEL_FORMAT and COMPRESSION are specified, COMPRESSION takes precedence. |

### SUPPORT VECTOR MACHINE

Optional:

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| KERNEL_TYPE | INTEGER | 2 | Kernel type:<br>• 0: LINEAR KERNEL<br>• 1: POLY KERNEL<br>• 2: RBF KERNEL<br>• 3: SIGMOID KERNEL | |
| THREAD_RATIO | DOUBLE | -1 | The ratio of available threads.<br>• 0: single thread<br>• 0–1: uses the specified percentage of available threads. PAL uses all available threads if the number is 1.0.<br>• -1: uses all of available threads. | Only valid for parameter selection task. |
| POLY_DEGREE | INTEGER | 3 | Coefficient for the PLOY KERNEL type.<br>Value range: ≥ 1 | Only valid when KERNEL_TYPE is 1. |
| RBF_GAMMA | DOUBLE | 1.0/number of features in the dataset | Coefficient for the RBF KERNEL type.<br>Value range: > 0 | Only valid when KERNEL_TYPE is 2. |
| COEF_LIN | DOUBLE | 0 | Coefficient for the POLY/SIGMOID KERNEL type. | Only valid when KERNEL_TYPE is 1 or 3. |
| COEF_CONST | DOUBLE | 0 | Coefficient for the POLY/SIGMOID KERNEL type. | Only valid when KERNEL_TYPE is 1 or 3. |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| SVM_C | DOUBLE | 100 | Trade-off between training error and margin.<br>Value range: > 0 | |
| SCALE_INFO | INTEGER | 1 | No scale<br>1: Standardization<br>The algorithm transforms the data to have zero mean and unit variance. The general formula is given as:<br>$x_{new}=\frac{x-\mu}{\sigma}$,<br>where x is the origin feature vector, μ is the mean of that feature vector, and σ is its standard deviation.<br>2: Rescaling<br>The algorithm rescales the range of the features to scale the range in [-1,1]. The general formula is given as:<br>$x_{new}=-1+\frac{x-\min(x)}{\max(x)-\min(x)}*2$,<br>where x is the origin feature vector. | |
| SCALE_LABEL | INTEGER | 1 | Controls whether to standardize the label for SVR.<br>• 0: No<br>• 1: Yes | Only valid when TYPE is 2 and SCALE_INFO is 1. |
| SHRINK | INTEGER | 1 | Decides whether to use shrink strategy or not:<br>• 0: Does not use shrink strategy<br>• 1: Uses shrink strategy | Using shrink strategy may accelerate the training process. |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| HANDLE_MISSING | INTEGER | 1 | Whether to handle missing values:<br>• 0: No<br>• 1: Yes | |
| CATEGORY_WEIGHT | DOUBLE | 0.707 | Represents the weight of category attributes (γ). The value must be greater than 0. | |
| TOL | DOUBLE | 0.001 | Specifies the error tolerance in the training process. The value must be greater than 0. | |
| EVALUATION_SEED | INTEGER | 0 | The random seed in parameter selection. The value must be greater than 0. | |
| COMPRESSION | INTEGER | 1 | Specifies if the model is stored in compressed format.<br>• 0 : no<br>• 1: yes | |
| MAX_BITS | INTEGER | 12 | The maximum number of bits to quantize continuous features. Equivalent to use 2^MAX_BITS bins. Must be less than 31. | Valid only when the value of COMPRESSION is 1. |
| MAX_QUANTIZATION_ITER | INTEGER | 1000 | The maximum iteration steps for quantization. | Valid only when the value of COMPRESSION is 1. |

Parameter search and selection

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| RESAMPLING_METHOD | NVARCHAR | No default value | Specifies the resampling method for model evaluation or parameter selection.<br>• cv<br>• cv_sha<br>• cv_hyperband<br>• stratified_cv<br>• stratified_cv_sha<br>• stratified_cv_hyperband<br>• bootstrap<br>• bootstrap_sha<br>• bootstrap_hyperband<br>• stratified_bootstrap<br>• stratified_bootstrap_sha<br>• stratified_bootstrap_hyperband<br>stratified_* can only apply to classification algorithms. If no value is specified for this parameter, neither model evaluation nor parameter selection is activated. | |
| EVALUATION_METRIC | NVARCHAR | No default value | Specifies the evaluation metric for model evaluation or parameter selection.<br>• ACCURACY<br>• F1_SCORE<br>• AUC | |
| FOLD_NUM | INTEGER | No default value | Specifies the fold number for the cross validation method. | Mandatory and valid only when RESAMPLING_METHOD is set to cv or stratified_cv. |
| REPEAT_TIMES | INTEGER | 1 | Specifies the number of repeat times for resampling. | |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| PARAM_SEARCH_STRATEGY | NVARCHAR | No default value | Specifies the parameter search method:<br>• grid<br>• random | When *_sha is specified in RESAMPLING_METHOD, PARAM_SEARCH_STRATEGY is mandatory. When *_hyperband is specified in RESAMPLING_METHOD, PARAM_SEARCH_STRATEGY is set to random whatever any other value is provided. |
| RANDOM_SEARCH_TIMES | INTEGER | No default value | Specifies the number of times to randomly select candidate parameters for selection. | Mandatory and valid when PARAM_SEARCH_STRATEGY is set to random. |
| REDUCTION_RATE | DOUBLE | 3.0 | Specifies reduction rate in SHA or Hyperband method. | Only valid when RESAMPLING_METHOD is specified as *_sha or *_hyberband. |
| AGGRESSIVE_ELIMINATION | INTEGER | | Specifies whether to apply aggressive elimination while using SHA method. Aggressive elimination happens when the data size and parameters size to be searched does not match and there are still bunch of parameters to be searched while data size reaches its upper limits. If aggressive elimination is applied, lower bound of limit of data size will be used multiple times first to reduce number of parameters.<br>• 0: does not apply aggressive elimination<br>• 1: applies aggressive elimination | Only valid when RESAMPLING_METHOD is specified as *_sha. |
| SEED | INTEGER | 0 | Specifies the seed for random generation. Use system time when 0 is specified. | |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| TIMEOUT | INTEGER | 0 | Specifies maximum running time for model evaluation or parameter selection, in seconds. No timeout when 0 is specified. | |
| PROGRESS_INDICATOR_ID | NVARCHAR | No default value | Sets an ID of progress indicator for model evaluation or parameter selection. No progress indicator is active if no value is provided. | |
| SVM_C_VALUES | NVARCHAR | No default value | Specifies values of SVM_C to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| SVM_C_RANGE | NVARCHAR | No default value | Specifies the range of SVM_C to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| RBF_GAMMA_VALUES | NVARCHAR | No default value | Specifies values of RBF_GAMMA to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| RBF_GAMMA_RANGE | NVARCHAR | No default value | Specifies the range of RBF_GAMMA to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| POLY_DEGREE_VALUES | NVARCHAR | No default value | Specifies values of POLY_DEGREE to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| POLY_DEGREE _RANGE | NVARCHAR | No default value | Specifies the range of POLY_DEGREE to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| COEF_LIN_VALUES | NVARCHAR | No default value | Specifies values of COEF_LIN to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| COEF_LIN_RANGE | NVARCHAR | No default value | Specifies the range of COEF_LIN to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| COEF_CONST_VALUES | NVARCHAR | No default value | Specifies values of COEF_CONST to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| COEF_CONST_RANGE | NVARCHAR | No default value | Specifies the range of COEF_CONST to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |

## _SYS_AFL. PAL_UNIFIED_CLASSIFICATION_PREDICT_PIVOT

### UNIFIED CLASSIFICATION

Mandatory Parameters

None

Optional Parameters

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| FEATURE_ATTRIBUTION_METHOD | INTEGER | 2 | Controls which method to use in model reasoning.<br>• 0: no reasoning<br>• 1: SAABAS<br>• 2: SHAP | Valid only for the functions: Decision Trees, Random Decision Trees and Hybrid Gradient Boosting Tree to generating the reason code. |
| TOP_K_ATTRIBUTIONS | INTEGER | 10 | Outputs the attributions of top k features which contribute the most. | |
| SAMPLESIZE | INTEGER | 0 | Number of sampled combinations of features. If it equals to 0, it is determined by algorithm heuristically. | It is better to user a number which is larger than the number of the columns of the data table.<br>Available for the following functions:<br>• Naive Bayes<br>• Support Vector Machine<br>• Multilayer Perceptron<br>• Multi-class Logistic Regression |
| SEED | INTEGER | 0 | Specifies the seed for random number generator.<br>• 0: Uses the current time (in second) as seed<br>• Others: Uses the specified value as seed | Available for the following functions:<br>• Naive Bayes<br>• Support Vector Machine<br>• Multilayer Perceptron<br>• Multi-class Logistic Regression |
| HANDLE_MISSING_VALUE | INTEGER | 0 | Whether to handle missing values:<br>• 0: No<br>• 1: Yes | |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| IMPUTATION_TYPE | INTEGER | 1 | Chooses the overall default imputation type for all columns:<br>• 0: NONE<br>• 1: MODE_MEAN<br>• 2: MODE_MEDIAN<br>• 3: MODE_ALL-ZERO<br>• 4: MODE_ALS<br>• 5: DELETE<br>NONE and DELETE: set all columns to imputation type NONE/DELETE.<br>MODE_MEAN, MODE_MEDIAN, MODE_ALLZERO, and MODE_ALS: set categorical columns to the first imputation type (the type before “”); set numerical columns to the second imputation type (the type after “”). ALLZERO means to fill all missing values in numerical columns with 0. | Only valid when HANDLE_MISSING_VALUE is 1. |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| `<column name>`_IMPUTATION_TYPE | INTEGER | No default value | Sets imputation type of a specified column. This value overwrites the default imputation type deduced by the IMPUTATION_TYPE parameter:<br>• 0: NONE<br>• 1: DELETE<br>• 100: MODE<br>• 101: SPECIFIED_CATEGORICAL_VALUE<br>• 200: MEAN<br>• 201: MEDIAN<br>• 203:SPECIFIED_NUMERICAL_VALUE<br>• 204: ALS<br>For type 101 (SPECIFIED_CATEGORICAL_VALUE), use the following syntax to specify a categorical value to be filled in a column V0:<br>INSERT INTO #PAL_PARAMETER_TBL VALUES('V0_IMPUTATION_TYPE', 101, NULL, '`<CATEGORICAL_VALUE>`');<br>For type 203 (SPECIFIED_NUMERICAL_VALUE), use the following syntax to specify a numerical value to be filled in a column V1:<br>INSERT INTO #PAL_PARAMETER_TBL VALUES('V1_IMPUTA- | Only valid when HANDLE_MISSING_VALUE is 1. |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| | | | TION_TYPE', 203, `<NUMERICAL_VALUE>`, NULL);<br><br>**Note**<br>A column of integer type can be set to type 101 (SPECIFIED_CATEGORICAL_VALUE) or type 203 (SPECIFIED_NUMERICAL_VALUE):<br>• when setting to type 101 (SPECIFIED_CATEGORICAL_VALUE), the string must contain a valid integer value, otherwise an error will be thrown;<br>• when setting to type 203 (SPECIFIED_NUMERICAL_VALUE), the double value will be converted to integer. | |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| ALS_FACTOR_NUMBER | INTEGER | 3 | Length of factor vectors, that is, the f in ALS model.<br>**Note**<br>ALS_FACTOR_NUMBER should be less than the number of numerical columns to get a meaningful result. | |
| ALS_REGULARIZATION | DOUBLE | 1e-2 | L2 regularization of the factors, that is, the λ in ALS model. | |
| ALS_MAX_ITERATION | INTEGER | 20 | Maximum number of iterations when training ALS model. | |
| ALS_SEED | INTEGER | 0 | Specifies the seed of the random number generator used in the training of ALS model:<br>• 0: Uses the current time as the seed<br>• Others: Uses the specified value as the seed. | |
| ALS_EXIT_THRESHOLD | DOUBLE | 0.0 | The training of ALS model exits if the value of cost function is not decreased more than the value of ALS_EXIT_THRESHOLD since the last check.<br>If ALS_EXIT_THRESHOLD is set to 0, there is no check and the training only exits on reaching the maximum number of iterations. Evaluations of cost function require additional calculations. You can set this parameter to 0 to avoid it. | |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| ALS_EXIT_INTERVAL | INTEGER | 5 | During the training of ALS model, it will calculate cost function and check the exit criteria every `<ALS_EXIT_INTERVAL>` iterations. Evaluations of cost function require additional calculations. | |
| ALS_LINEAR_SYSTEM_SOLVER | INTEGER | 0 | • 0: cholesky solver<br>• 1: cg solver (recommended when ALS_FACTOR_NUMBER is large) | |
| ALS_CG_MAX_ITERATION | INTEGER | 3 | Maximum number of iteration of cg solver. | |
| ALS_CENTERING | INTEGER | 1 | Whether to center the data by column before training ALS model.<br>• 0: no centering<br>• 1: centering. | |
| ALS_SCALING | INTEGER | 1 | Whether to scale the data by column before training ALS model.<br>• 0: no scaling<br>• 1: scaling | |

## DECISION TREES

Optional

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION |
|---|---|---|---|
| THREAD_RATIO | DOUBLE | -1 | The ratio of available threads.<br>• 0: single thread<br>• 0–1: uses the specified percentage of available threads. PAL uses all available threads if the number is 1.0.<br>• -1: uses all of available threads. |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION |
|---|---|---|---|
| VERBOSE | INTEGER | 0 | Specifies whether to output all classes and the corresponding confidences for each data.<br>• 0: no<br>• 1: yes |

### HYBRID GRADIENT BOOSTING TREE

Optional

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| THREAD_RATIO | DOUBLE | -1 | The ratio of available threads.<br>• 0: single thread<br>• 0–1: uses the specified percentage of available threads. PAL uses all available threads if the number is 1.0.<br>• -1: uses all of available threads. | |
| VERBOSE | INTEGER | 0 | Specifies whether to output all classes and the corresponding confidences for each data.<br>• 0: no<br>• 1: yes | |
| MISSING_REPLACEMENT | INTEGER | 1 | Specifies the missing replacement strategy:<br>• 1: marginalises each missing feature out independently<br>• 2: marginalises all missing features in an instance as a whole corresponding to each category | Strategy 2 is a much coarser approximation. Only apply strategy 2 when there are a great number of missing features and a lot of classed, which might cause memory insufficiency. |

### LOGISTIC REGRESSION

Mandatory

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION |
|---|---|---|---|
| CLASS_MAP0 | VARCHAR | No default value | The same value as the parameter of PAL_LOGISTIC_REGRESSION. |
| CLASS_MAP1 | VARCHAR | No default value | The same value as the parameter of PAL_LOGISTIC_REGRESSION. |

Optional

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION |
|---|---|---|---|
| THREAD_RATIO | DOUBLE | -1 | The ratio of available threads.<br>• 0: single thread<br>• 0–1: uses the specified percentage of available threads. PAL uses all available threads if the number is 1.0.<br>• -1: uses all of available threads. |

### MULTI-CLASS LOGISTIC REGRESSION

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION |
|---|---|---|---|
| THREAD_RATIO | DOUBLE | -1 | The ratio of available threads.<br>• 0: single thread<br>• 0–1: uses the specified percentage of available threads. PAL uses all available threads if the number is 1.0.<br>• -1: uses all of available threads. |
| IGNORE_UNKNOWN_CATEGORY | INTEGER | 1 | • 0: Procedures will report error if unknown category value is found.<br>• 1: Ignores unknown category value |
| VERBOSE | INTEGER | 0 | • 1: Outputs the probability of all label categories<br>• 0: Outputs the category of the highest probability only |

### MULTILAYER PERCEPTRON

Optional

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION |
|---|---|---|---|
| THREAD_RATIO | DOUBLE | -1 | The ratio of available threads.<br>• 0: single thread<br>• 0–1: uses the specified percentage of available threads. PAL uses all available threads if the number is 1.0.<br>• -1: uses all of available threads. |
| VERBOSE | INTEGER | 0 | Output scoring probabilities for each class. |

### NAÏVE BAYES

Optional

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION |
|---|---|---|---|
| LAPLACE | DOUBLE | 0 | Enables or disables Laplace smoothing.<br>• 0: Disables Laplace smoothing.<br>• Positive value: Enables Laplace smoothing for discrete values.<br>**Note**<br>Non-zero LAPLACE value is required if there exist discrete category values that only occur in the test set. The LAPLACE value you set here takes precedence over the values read from JSON models. |
| VERBOSE | INTEGER | 0 | • 1: Outputs the probability of all label categories.<br>• 0: Outputs the category of the highest probability only. |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION |
|---|---|---|---|
| THREAD_RATIO | DOUBLE | -1 | The ratio of available threads.<br>• 0: single thread<br>• 0–1: uses the specified percentage of available threads. PAL uses all available threads if the number is 1.0.<br>• -1: uses all of available threads. |

### RANDOM DECISION TREES

Optional

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| THREAD_RATIO | DOUBLE | -1 | The ratio of available threads.<br>• 0: single thread<br>• 0–1: uses the specified percentage of available threads. PAL uses all available threads if the number is 1.0.<br>• -1: uses all of available threads. | |
| VERBOSE | INTEGER | 0 | Specifies whether to output all classes and the corresponding confidences for each data.<br>• 0: no<br>• 1: yes | |
| BLOCK_SIZE | INTEGER | 0 | Specifies the number of data loaded per time during scoring.<br>• 0: load all data once<br>• Others: the specified number | This parameter is for reducing memory consumption, especially as the predict data is huge, or it consists of a large number of missing independent variables. However, you might lose some efficiency. |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| MISSING_REPLACE-MENT | INTEGER | 1 | Specifies the missing replacement strategy:<br>• 1: marginalises each missing feature out independently<br>• 2: marginalises all missing features in an instance as a whole corresponding to each category | Strategy 2 is a much coarser approximation. Only apply strategy 2 when there are a great number of missing features and a lot of classed, which might cause memory insufficiency. |

### SUPPORT VECTOR MACHINE

Optional

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION |
|---|---|---|---|
| THREAD_RATIO | DOUBLE | -1 | The ratio of available threads.<br>• 0: single thread<br>• 0–1: uses the specified percentage of available threads. PAL uses all available threads if the number is 1.0.<br>• -1: uses all of available threads. |
| VERBOSE | INTEGER | 0 | Output scoring probabilities for each class. |

## _SYS_AFL. PAL_UNIFIED_CLASSIFICATION_SCORE_PIVOT

### UNIFIED CLASSICATION

Mandatory

None

Optional Parameters

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| FEATURE_ATTRIBUTION_METHOD | INTEGER | 2 | Controls which method to use in model reasoning.<br>• 0: no reasoning<br>• 1: SAABAS<br>• 2: SHAP | Valid only for the functions: Decision Trees, Random Decision Trees and Hybrid Gradient Boosting Tree to generating the reason code. |
| TOP_K_ATTRIBUTIONS | INTEGER | 10 | Outputs the attributions of top k features which contribute the most. | |
| MAX_RESULT_NUM | INTEGER | No default value | Specifies the output number of prediction results | |
| NTILES | INTEGER | 1 or 20 | Used to control the population tiles in metrics output.<br>The validation value should be at least 1 and no larger than the row size of the validation data.<br>**Note**<br>For AUC, this parameter means the maximum tiles. | If the row size of data for metrics evaluation is less than 20, the default value is the 1; otherwise it is 20. |
| SAMPLESIZE | INTEGER | 0 | Number of sampled combinations of features. If it equals to 0, it is determined by algorithm heuristically. | It is better to user a number which is larger than the number of the columns of the data table.<br>Available for the following functions:<br>• Naive Bayes<br>• Support Vector Machine<br>• Multilayer Perceptron<br>• Multi-class Logistic Regression |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| SEED | INTEGER | 0 | Specifies the seed for random number generator.<br>• 0: Uses the current time (in second) as seed<br>• Others: Uses the specified value as seed | Available for the following functions:<br>• Naive Bayes<br>• Support Vector Machine<br>• Multilayer Perceptron<br>• Multi-class Logistic Regression |
| HANDLE_MISSING_VALUE | INTEGER | 0 | Whether to handle missing values:<br>• 0: No<br>• 1: Yes | |
| IMPUTATION_TYPE | INTEGER | 1 | Chooses the overall default imputation type for all columns:<br>• 0: NONE<br>• 1: MODE_MEAN<br>• 2: MODE_MEDIAN<br>• 3: MODE_ALLZERO<br>• 4: MODE_ALS<br>• 5: DELETE<br>NONE and DELETE: set all columns to imputation type NONE/DELETE.<br>MODE_MEAN, MODE_MEDIAN, MODE_ALLZERO, and MODE_ALS: set categorical columns to the first imputation type (the type before “”); set numerical columns to the second imputation type (the type after “”). ALLZERO means to fill all missing values in numerical columns with 0. | Only valid when HANDLE_MISSING_VALUE is 1. |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| <column name>_IMPUTATION_TYPE | INTEGER | No default value | Sets imputation type of a specified column. This value overwrites the default imputation type deduced by the IMPUTATION_TYPE parameter:<br>• 0: NONE<br>• 1: DELETE<br>• 100: MODE<br>• 101: SPECIFIED_CATEGORICAL_VALUE<br>• 200: MEAN<br>• 201: MEDIAN<br>• 203:SPECIFIED_NUMERICAL_VALUE<br>• 204: ALS<br>For type 101 (SPECIFIED_CATEGORICAL_VALUE), use the following syntax to specify a categorical value to be filled in a column V0:<br>INSERT INTO #PAL_PARAMETER_TBL VALUES('V0_IMPUTATION_TYPE', 101, NULL, '<CATEGORICAL_VALUE>');<br>For type 203 (SPECIFIED_NUMERICAL_VALUE), use the following syntax to specify a numerical value to be filled in a column V1:<br>INSERT INTO #PAL_PARAMETER_TBL VALUES('V1_IMPUTA- | Only valid when HANDLE_MISSING_VALUE is 1. |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| | | | TION_TYPE', 203, `<NUMERICAL_VALUE>`, NULL);<br><br>**Note**<br>A column of integer type can be set to type 101 (SPECIFIED_CATEGORICAL_VALUE) or type 203 (SPECIFIED_NUMERICAL_VALUE):<br>• when setting to type 101 (SPECIFIED_CATEGORICAL_VALUE), the string must contain a valid integer value, otherwise an error will be thrown;<br>• when setting to type 203 (SPECIFIED_NUMERICAL_VALUE), the double value will be converted to integer. | |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| ALS_FACTOR_NUMBER | INTEGER | 3 | Length of factor vectors, that is, the f in ALS model.<br>**Note**<br>ALS_FACTOR_NUMBER should be less than the number of numerical columns to get a meaningful result. | |
| ALS_REGULARIZATION | DOUBLE | 1e-2 | L2 regularization of the factors, that is, the λ in ALS model. | |
| ALS_MAX_ITERATION | INTEGER | 20 | Maximum number of iterations when training ALS model. | |
| ALS_SEED | INTEGER | 0 | Specifies the seed of the random number generator used in the training of ALS model:<br>• 0: Uses the current time as the seed<br>• Others: Uses the specified value as the seed. | |
| ALS_EXIT_THRESHOLD | DOUBLE | 0.0 | The training of ALS model exits if the value of cost function is not decreased more than the value of ALS_EXIT_THRESHOLD since the last check.<br>If ALS_EXIT_THRESHOLD is set to 0, there is no check and the training only exits on reaching the maximum number of iterations. Evaluations of cost function require additional calculations. You can set this parameter to 0 to avoid it. | |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| ALS_EXIT_INTERVAL | INTEGER | 5 | During the training of ALS model, it will calculate cost function and check the exit criteria every `<ALS_EXIT_INTERVAL>` iterations. Evaluations of cost function require additional calculations. | |
| ALS_LINEAR_SYSTEM_SOLVER | INTEGER | 0 | • 0: cholesky solver<br>• 1: cg solver (recommended when ALS_FACTOR_NUMBER is large) | |
| ALS_CG_MAX_ITERATION | INTEGER | 3 | Maximum number of iteration of cg solver. | |
| ALS_CENTERING | INTEGER | 1 | Whether to center the data by column before training ALS model.<br>• 0: no centering<br>• 1: centering. | |
| ALS_SCALING | INTEGER | 1 | Whether to scale the data by column before training ALS model.<br>• 0: no scaling<br>• 1: scaling | |

### DECISION TREES

Optional

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION |
|---|---|---|---|
| THREAD_RATIO | DOUBLE | -1 | The ratio of available threads.<br>• 0: single thread<br>• 0–1: uses the specified percentage of available threads. PAL uses all available threads if the number is 1.0.<br>• -1: uses all of available threads. |

### HYBRID GRADIENT BOOSTING TREE

Optional

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| THREAD_RATIO | DOUBLE | -1 | The ratio of available threads.<br>• 0: single thread<br>• 0–1: uses the specified percentage of available threads. PAL uses all available threads if the number is 1.0.<br>• -1: uses all of available threads. | |
| MISSING_REPLACEMENT | INTEGER | 1 | Specifies the missing replacement strategy:<br>• 1: marginalises each missing feature out independently<br>• 2: marginalises all missing features in an instance as a whole corresponding to each category | Strategy 2 is a much coarser approximation. Only apply strategy 2 when there are a great number of missing features and a lot of classed, which might cause memory insufficiency. |

### LOGISTIC REGRESSION

Mandatory

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION |
|---|---|---|---|
| CLASS_MAP0 | VARCHAR | No default value | The same value as the parameter of PAL_LOGISTIC_REGRESSION. |
| CLASS_MAP1 | VARCHAR | No default value | The same value as the parameter of PAL_LOGISTIC_REGRESSION. |

Optional

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION |
|---|---|---|---|
| THREAD_RATIO | DOUBLE | -1 | The ratio of available threads.<br>• 0: single thread<br>• 0–1: uses the specified percentage of available threads. PAL uses all available threads if the number is 1.0.<br>• -1: uses all of available threads. |

### MULTI-CLASS LOGISTIC REGRESSION

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION |
|---|---|---|---|
| THREAD_RATIO | DOUBLE | -1 | The ratio of available threads.<br>• 0: single thread<br>• 0–1: uses the specified percentage of available threads. PAL uses all available threads if the number is 1.0.<br>• -1: uses all of available threads. |
| IGNORE_UNKNOWN_CATEGORY | INTEGER | 1 | • 0: Procedures will report error if unknown category value is found.<br>• 1: Ignores unknown category value |

### MULTILAYER PERCEPTRON

Optional

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION |
|---|---|---|---|
| THREAD_RATIO | DOUBLE | -1 | The ratio of available threads.<br>• 0: single thread<br>• 0–1: uses the specified percentage of available threads. PAL uses all available threads if the number is 1.0.<br>• -1: uses all of available threads. |

### NAÏVE BAYES

Optional

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION |
|---|---|---|---|
| LAPLACE | DOUBLE | 0 | Enables or disables Laplace smoothing.<br>• 0: Disables Laplace smoothing.<br>• Positive value: Enables Laplace smoothing for discrete values.<br>**Note**<br>Non-zero LAPLACE value is required if there exist discrete category values that only occur in the test set. The LAPLACE value you set here takes precedence over the values read from JSON models. |
| THREAD_RATIO | DOUBLE | -1 | The ratio of available threads.<br>• 0: single thread<br>• 0–1: uses the specified percentage of available threads. PAL uses all available threads if the number is 1.0.<br>• -1: uses all of available threads. |

### RANDOM DECISION TREES

Optional

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| THREAD_RATIO | DOUBLE | -1 | The ratio of available threads.<br>• 0: single thread<br>• 0–1: uses the specified percentage of available threads. PAL uses all available threads if the number is 1.0.<br>• -1: uses all of available threads. | |

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| BLOCK_SIZE | INTEGER | 0 | Specifies the number of data loaded per time during scoring.<br>• 0: load all data once<br>• Others: the specified number | This parameter is for reducing memory consumption, especially as the predict data is huge, or it consists of a large number of missing independent variables. However, you might lose some efficiency. |
| MISSING_REPLACEMENT | INTEGER | 1 | Specifies the missing replacement strategy:<br>• 1: marginalises each missing feature out independently<br>• 2: marginalises all missing features in an instance as a whole corresponding to each category | Strategy 2 is a much coarser approximation. Only apply strategy 2 when there are a great number of missing features and a lot of classed, which might cause memory insufficiency. |

**SUPPORT VECTOR MACHINE**

Optional

| NAME | DATA TYPE | DEFAULT VALUE | DESCRIPTION |
|---|---|---|---|
| THREAD_RATIO | DOUBLE | -1 | The ratio of available threads.<br>• 0: single thread<br>• 0–1: uses the specified percentage of available threads. PAL uses all available threads if the number is 1.0.<br>• -1: uses all of available threads. |

## 5.13.3 Unified Massive Classification

The unified massive classification execution function is a unified interface to call classification algorithms with unpivot data format.

The unified massive classification algorithms include:

- DT (Decision Trees)
- HGBT (Hybrid gradient boosting tree)

- LOGR (Logistic regression)
- M_LOGR (Multi-class logistic regression)
- NB (Naïve Bayes)
- RDT (Random decision trees)
- SVM (Support Vector Machine)

The keyword group name ('PAL_MASSIVE_PROCESSING_SPECIAL_GROUP_ID') is valid for all groups if the group name is not defined. For example, There are 2 groups ('A', 'B'),

```
INSERT INTO #PAL_PARAMETER_TBL VALUES
('PAL_MASSIVE_PROCESSING_SPECIAL_GROUP_ID', 'ALPHA', NULL, 1.0, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES
('PAL_MASSIVE_PROCESSING_SPECIAL_GROUP_ID', 'VERBOSE', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('B', 'VERBOSE', 0, NULL, NULL);
```

Therefore, A: ALPHA (1.0), VERBOSE (1); B: VERBOSE (0).

For more details on the feature of the unified massive classification, see the corresponding topics (Local Interpretability, Data Partition...) in the chapter on unified classification.

Moreover, it is clear that the models created from the unified massive classification and the unified classification are incompatible.

## Prerequisites

- All prerequisites for the unified classification function apply in the unified massive classification execution interface.

## _SYS_AFL.PAL_UNIFIED_MASSIVE_CLASSIFICATION

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA Table>` | IN |
| 2 | `<PARAMETER Table>` | IN |
| 3 | `<MODEL Table>` | OUT |
| 4 | `<VARIABLE_IMPORTANCE Table>` | OUT |
| 5 | `<STATISTICS Table>` | OUT |
| 6 | `<OPTIMAL_PARAM Table>` | OUT |
| 7 | `<CONFUSION_MATRIX Table>` | OUT |
| 8 | `<METRICS Table>` | OUT |
| 9 | `<PARTITION_TYPE Table>` | OUT |

| Position | Table Name | Parameter Type |
|---|---|---|
| 10 | `<ERRORMSG Table>` | OUT |
| 11 | `<PLACE_HOLDER1 Table>` | OUT |

### Input Table

| Table | Column | Data Type | Description |
|---|---|---|---|
| DATA | 1st column | INTEGER, VRACHAR or NVARCHAR | Group id. |
| | 2nd column | INTEGER, VRACHAR or NVARCHAR | Record id. If this column is absent, the KEY parameter must be set to 0. |
| | Feature columns | INTEGER, DOUBLE, DECIMAL(p,s), VARCHAR, or NVARCHAR | Variable values. |
| | LABEL column | INTEGER, VARCHAR, or NVARCHAR | Dependent variable.<br>Normally, only VARCHAR and NVARCHAR are allowed. If INTEGER is used, CATEGORICAL_VARIABLE needs to be used to specify this column. |
| | Purpose column (Optional) | INTEGER | An optional column used for user-defined data partition. This column is needed if PARTITION_METHOD is set to 1. Otherwise, the column does not exist. The valid values are:<br>• 1: training;<br>• 2: validation |
| PARAMETER | 1st column | INTEGER, VRACHAR or NVARCHAR | Group id. |
| | 2nd column | VARCHAR or NVARCHAR | Parameter name. |
| | 3rd column | INTEGER | Int value. |
| | 4th column | DOUBLE | Double value. |
| | 5th column | VARCHAR or NVARCHAR | String value. |

### Parameter Table

Mandatory Parameters

The following parameters are mandatory and must be given a value.

| Name | Data Type | Description |
|---|---|---|
| FUNCTION | VARCHAR or NVARCHAR | The algorithm name, including:<br>• DT;<br>• HGBT;<br>• LOGR;<br>• M_LOGR;<br>• NB;<br>• RDT;<br>• SVM. |

Optional Parameters

The parameter of the unified massive classification execution function is the same as the unified classification function.

See Parameter Tables [page 1495] for details about parameters for a specific algorithm.

**Output Tables**

| Table | Column | Column Name | Column Data Type |
|---|---|---|---|
| MODEL | 1st column | GROUP_ID | Group id column data type. |
| | 2nd column | ROW_INDEX | INTEGER |
| | 3rd column | PART_INDEX | INTEGER |
| | 4th column | MODEL_CONTENT | NCLOB |
| VARIABLE_IMPORTANCE | 1st column | GROUP_ID | Group id column data type. |
| | 2nd column | VARIABLE_NAME | NVARCHAR (256) |
| | 3rd column | IMPORTANCE | DOUBLE |
| STATISTICS | 1st column | GROUP_ID | Group id column data type. |
| | 2nd column | STAT_NAME | NVARCHAR (256) |
| | 3rd column | STAT_VALUE | NVARCHAR (1000) |
| | 4th column | CLASS_NAME | NVARCHAR (256) |
| OPTIMAL_PARAM | 1st column | GROUP_ID | Group id column data type. |
| | 2nd column | PARAM_NAME | NVARCHAR (256) |
| | 3rd column | INT_VALUE | INTEGER |
| | 4th column | DOUBLE_VALUE | DOUBLE |
| | 5th column | STRING_VALUE | NVARCHAR (1000) |
| CONFUSION_MATRIX | 1st column | GROUP_ID | Group id column data type. |
| | 2nd column | ACTUAL_CLASS | NVARCHAR (1000) |
| | 3rd column | PREDICTED_CLASS | NVARCHAR (1000) |
| | 4th column | COUNT | INTEGER |
| METRICS | 1st column | GROUP_ID | Group id column data type. |

| Table | Column | Column Name | Column Data Type |
|---|---|---|---|
| | 2nd column | NAME | NVARCHAR (256) |
| | 3rd column | X | DOUBLE |
| | 4th column | Y | DOUBLE |
| PARTITION_TYPE | 1st column | GROUP_ID | Group id column data type. |
| | 2nd column | ID column name (if there is no ID column, the name of first column of data table.) | ID column data type (if there is no ID column, the type of first column of data table.) |
| | 3rd column | TYPE | INTEGER |
| ERRORMSG | 1st column | GROUP_ID | Group id column data type. |
| | 2nd column | ERROR_TIMESTAMP | VARCHAR, NVARCHAR |
| | 3rd column | ERRORCODE | INTEGER |
| | 4th column | MASSAGE | VARCHAR, NVARCHAR |
| PLCAE_HOLDER1 | 1st column | GROUP_ID | Group id column data type. |
| | 2nd column | OBJECT | VARCHAR, NVARCHAR |
| | 3rd column | KEY | VARCHAR, NVARCHAR |
| | 4th column | VALUE | VARCHAR, NVARCHAR |

### Example

Assume that:

- DM_PAL is a schema belonging to USER1;
- USER1 has been assigned the AFL__SYS_AFL_AFLPAL_EXECUTE or AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION role.

Example: Naive Bayes

```
SET SCHEMA DM_PAL;
-----------------------------------Unified Massive Classification (Naive
Bayes)--------------------------
DROP TABLE  PAL_DATA_TBL;
CREATE COLUMN TABLE PAL_DATA_TBL (
    "GROUP_ID" INTEGER,
    "ID" INTEGER,
    "OUTLOOK" NVARCHAR(20),
    "TEMP" DOUBLE,
    "HUMIDITY" DOUBLE,
    "WINDY" NVARCHAR(10),
    "CLASS" NVARCHAR(20)
);
INSERT INTO PAL_DATA_TBL VALUES (1, 1, 'Sunny', 75, 70.0, 'Yes', 'Play');
INSERT INTO PAL_DATA_TBL VALUES (1, 2, 'Sunny', 80, 90.0, 'Yes', 'Do not Play');
INSERT INTO PAL_DATA_TBL VALUES (1, 3, 'Sunny', 85, 91.0, 'No', 'Do not Play');
INSERT INTO PAL_DATA_TBL VALUES (1, 4, 'Sunny', 72, 95.0, 'No', 'Do not Play');
INSERT INTO PAL_DATA_TBL VALUES (1, 5, 'Sunny', 73, 70.0, 'No', 'Play');
INSERT INTO PAL_DATA_TBL VALUES (1, 6, 'Overcast', 72.0, 90, 'Yes', 'Do not
Play');
INSERT INTO PAL_DATA_TBL VALUES (1, 7, 'Overcast', 83.0, 78, 'No', 'Play');
INSERT INTO PAL_DATA_TBL VALUES (1, 8, 'Overcast', 64.0, 65, 'Yes', 'Do not
Play');
INSERT INTO PAL_DATA_TBL VALUES (1, 9, 'Overcast', 81.0, 75, 'No', 'Play');
```

```
INSERT INTO PAL_DATA_TBL VALUES (1, 10, 'Rain', 71, 80.0, 'Yes', 'Play');
INSERT INTO PAL_DATA_TBL VALUES (1, 11, 'Rain', 65, 70.0, 'Yes', 'Do not Play');
INSERT INTO PAL_DATA_TBL VALUES (1, 12, 'Rain', 75, 80.0, 'No', 'Play');
INSERT INTO PAL_DATA_TBL VALUES (1, 13, 'Rain', 68, 80.0, 'No', 'Play');
INSERT INTO PAL_DATA_TBL VALUES (1, 14, 'Rain', 70, 96.0, 'No', 'Play');
INSERT INTO PAL_DATA_TBL VALUES (2, 1, 'Overcast', 68, 70.0, 'No', 'Play');
INSERT INTO PAL_DATA_TBL VALUES (2, 2, 'Overcast', 80, 90.0, 'No', 'Do not
Play');
INSERT INTO PAL_DATA_TBL VALUES (2, 3, 'Overcast', 85, 91.0, 'Yes', 'Do not
Play');
INSERT INTO PAL_DATA_TBL VALUES (2, 4, 'Overcast', 78, 95.0, 'Yes', 'Do not
Play');
INSERT INTO PAL_DATA_TBL VALUES (2, 5, 'Overcast', 73, 70.0, 'Yes', 'Play');
INSERT INTO PAL_DATA_TBL VALUES (2, 6, 'Sunny', 68.0, 90, 'No', 'Do not Play');
INSERT INTO PAL_DATA_TBL VALUES (2, 7, 'Sunny', 83.0, 78, 'Yes', 'Play');
INSERT INTO PAL_DATA_TBL VALUES (2, 8, 'Sunny', 64.0, 65, 'Yes', 'Do not Play');
INSERT INTO PAL_DATA_TBL VALUES (2, 9, 'Sunny', 81.0, 75, 'No', 'Play');
INSERT INTO PAL_DATA_TBL VALUES (2, 10, 'Rain', 65, 80.0, 'No', 'Play');
INSERT INTO PAL_DATA_TBL VALUES (2, 11, 'Rain', 65, 70.0, 'No', 'Do not Play');
INSERT INTO PAL_DATA_TBL VALUES (2, 12, 'Rain', 85, 80.0, 'Yes', 'Play');
INSERT INTO PAL_DATA_TBL VALUES (2, 13, 'Rain', 68, 80.0, 'Yes', 'Play');
INSERT INTO PAL_DATA_TBL VALUES (2, 14, 'Rain', 70, 96.0, 'Yes', 'Play');
DROP TABLE PAL_PARAMETER_TBL;
CREATE  COLUMN TABLE PAL_PARAMETER_TBL (
    "GROUP_ID" INTEGER,
    "PARAM_NAME" NVARCHAR (100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR (100)
);
INSERT INTO PAL_PARAMETER_TBL VALUES (1, 'FUNCTION', NULL, NULL, 'NB');
INSERT INTO PAL_PARAMETER_TBL VALUES (1, 'KEY', 1, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES (1, 'BACKGROUND_SIZE', 4, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES (1, 'PARTITION_METHOD', 2, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES (1, 'PARTITION_RANDOM_SEED', 2, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES (1, 'PARTITION_STRATIFIED_VARIABLE', NULL,
NULL, 'CLASS');
INSERT INTO PAL_PARAMETER_TBL VALUES (1, 'PARTITION_TRAINING_PERCENT', NULL,
0.7, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES (1, 'OUTPUT_PARTITION_RESULT', 1, NULL,
NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES (2, 'FUNCTION', NULL, NULL, 'NB');
INSERT INTO PAL_PARAMETER_TBL VALUES (2, 'KEY', 1, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES (2, 'BACKGROUND_SIZE', 4, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES (2, 'PARTITION_METHOD', 2, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES (2, 'PARTITION_RANDOM_SEED', 2, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES (2, 'PARTITION_STRATIFIED_VARIABLE', NULL,
NULL, 'CLASS');
INSERT INTO PAL_PARAMETER_TBL VALUES (2, 'PARTITION_TRAINING_PERCENT', NULL,
0.7, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES (2, 'OUTPUT_PARTITION_RESULT', 1, NULL,
NULL);
DROP TABLE PAL_MODEL_RDT_TBL;
CREATE COLUMN TABLE PAL_MODEL_RDT_TBL (
    "GROUP_ID" INTEGER,
    "ROW_INDEX" INTEGER,
    "PART_INDEX" INTEGER,
    "MODEL_CONTENT" NCLOB
);
DROP TABLE PAL_IMP_TBL;
CREATE COLUMN TABLE PAL_IMP_TBL (
    "GROUP_ID" INTEGER,
    "VARIABLE_NAME" NVARCHAR(256),
    "IMPORTANCE" DOUBLE
);
DROP TABLE PAL_STAT_TBL;
CREATE COLUMN TABLE PAL_STAT_TBL (
    "GROUP_ID" INTEGER,
```

```
    "STAT_NAME" NVARCHAR(256),
    "STAT_VALUE" NVARCHAR(1000),
    "CLASS_NAME" NVARCHAR(256)
);
DROP TABLE PAL_CMATRIX_TBL;
CREATE COLUMN TABLE PAL_CMATRIX_TBL (
    "GROUP_ID" INTEGER,
    "ACTUAL_CLASS" NVARCHAR(256),
    "PREDICTED_CLASS" NVARCHAR(256),
    "COUNT" INTEGER
);
DROP TABLE PAL_METRICS_TBL;
CREATE COLUMN TABLE PAL_METRICS_TBL (
    "GROUP_ID" INTEGER,
    "NAME" NVARCHAR(256),
    "X" DOUBLE,
    "Y" DOUBLE
);
DROP TABLE PAL_PARTITION_TBL;
CREATE COLUMN TABLE PAL_PARTITION_TBL (
    "GROUP_ID" INTEGER,
    "DATA_ID" INTEGER,
    "TYPE" INTEGER
);
DO
BEGIN
    lt_data = SELECT * FROM PAL_DATA_TBL;
    lt_ctrl = SELECT * FROM PAL_PARAMETER_TBL;
    CALL _SYS_AFL.PAL_UNIFIED_MASSIVE_CLASSIFICATION (:lt_data, :lt_ctrl,
lt_model,lt_imp,lt_stat, lt_opt,lt_cm,lt_metrics,lt_partition,lt_errmsg, lt_ph1);
    INSERT INTO PAL_MODEL_RDT_TBL SELECT * FROM :lt_model;
    INSERT INTO PAL_IMP_TBL SELECT * FROM :lt_imp;
    INSERT INTO PAL_STAT_TBL SELECT * FROM :lt_stat;
    INSERT INTO PAL_CMATRIX_TBL SELECT * FROM :lt_cm;
    INSERT INTO PAL_METRICS_TBL SELECT * FROM :lt_metrics;
    INSERT INTO PAL_PARTITION_TBL SELECT * FROM :lt_partition;
    select * from :lt_errmsg;
END;
SELECT * FROM PAL_MODEL_RDT_TBL;
SELECT * FROM PAL_IMP_TBL;
SELECT * FROM PAL_STAT_TBL;
SELECT * FROM PAL_CMATRIX_TBL;
SELECT * FROM PAL_METRICS_TBL;
SELECT * FROM PAL_PARTITION_TBL;
```

**Expected Result**:

MODEL:

Rows (4)

| | GROUP_ID | ROW_INDEX | PART_INDEX | |
|---|---|---|---|---|
| 1 | 2 | 0 | 0 | {"Laplace":0.0,"ColNumb":4,"ClassSize":2,"Class":[{"ClassName":"Play","TransValue":0,"PriorFreq":6},{"ClassName":"Do not Play","TransValue":1,"PriorFreq":4}],"Train":[{"ColTyp |
| 2 | 2 | -10 | 1 | {"cate_vals":[{"cateVal":["Overcast","Rain","Rain"]},{"cateVal":["Yes","No","No"]},{"cateVal":["Play","Do not Play","Play"]}],"categorical_size":3,"class_index":5,"cont_vals":[{"contVa |
| 3 | 1 | 0 | 0 | {"Laplace":0.0,"ColNumb":4,"ClassSize":2,"Class":[{"ClassName":"Play","TransValue":0,"PriorFreq":6},{"ClassName":"Do not Play","TransValue":1,"PriorFreq":4}],"Train":[{"ColTyp |
| 4 | 1 | -10 | 1 | {"cate_vals":[{"cateVal":["Sunny","Rain","Rain"]},{"cateVal":["No","Yes","Yes"]},{"cateVal":["Play","Do not Play","Play"]}],"categorical_size":3,"class_index":5,"cont_vals":[{"contVal": |

## STATISTIC:

Rows (24)

| | GROUP_ID | STAT_NAME | STAT_VALUE | CLASS_NAME |
|---|---|---|---|---|
| 1 | 2 | AUC | 0.5625 | *NULL* |
| 2 | 2 | RECALL | 1 | Do not Play |
| 3 | 2 | PRECISION | 0.6666666666666666 | Do not Play |
| 4 | 2 | F1_SCORE | 0.8 | Do not Play |
| 5 | 2 | SUPPORT | 2 | Do not Play |
| 6 | 2 | RECALL | 0.5 | Play |
| 7 | 2 | PRECISION | 1 | Play |
| 8 | 2 | F1_SCORE | 0.6666666666666666 | Play |
| 9 | 2 | SUPPORT | 2 | Play |
| 10 | 2 | ACCURACY | 0.75 | *NULL* |
| 11 | 2 | KAPPA | 0.5 | *NULL* |
| 12 | 2 | MCC | 0.5773502691896258 | *NULL* |
| 13 | 1 | AUC | 0.375 | *NULL* |
| 14 | 1 | RECALL | 0.5 | Do not Play |
| 15 | 1 | PRECISION | 0.5 | Do not Play |
| 16 | 1 | F1_SCORE | 0.5 | Do not Play |
| 17 | 1 | SUPPORT | 2 | Do not Play |
| 18 | 1 | RECALL | 0.5 | Play |
| 19 | 1 | PRECISION | 0.5 | Play |
| 20 | 1 | F1_SCORE | 0.5 | Play |
| 21 | 1 | SUPPORT | 2 | Play |
| 22 | 1 | ACCURACY | 0.5 | *NULL* |
| 23 | 1 | KAPPA | 0 | *NULL* |
| 24 | 1 | MCC | 0 | *NULL* |

## CONFUSION MATRIX:

Rows (8)

| | GROUP_ID | ACTUAL_CLASS | PREDICTED_CLASS | COUNT |
|---|---|---|---|---|
| 1 | 2 | Do not Play | Do not Play | 2 |
| 2 | 2 | Do not Play | Play | 0 |
| 3 | 2 | Play | Do not Play | 1 |
| 4 | 2 | Play | Play | 1 |
| 5 | 1 | Do not Play | Do not Play | 1 |
| 6 | 1 | Do not Play | Play | 1 |
| 7 | 1 | Play | Do not Play | 1 |
| 8 | 1 | Play | Play | 1 |

## METRICS:

Rows (72)

| | GROUP_ID | NAME | X | Y |
|---|---|---|---|---|
| 21 | 2 | ROC_FPR | 4 | 0.25 |
| 22 | 2 | ROC_TPR | 4 | 0.75 |
| 23 | 2 | ROC_FPR | 5 | 0.5 |
| 24 | 2 | ROC_TPR | 5 | 0.75 |
| 25 | 2 | ROC_FPR | 6 | 0.75 |
| 26 | 2 | ROC_TPR | 6 | 0.75 |
| 27 | 2 | ROC_FPR | 7 | 1 |
| 28 | 2 | ROC_TPR | 7 | 0.75 |
| 29 | 2 | ROC_FPR | 8 | 1 |
| 30 | 2 | ROC_TPR | 8 | 1 |
| 31 | 2 | CUMGAINS | 0 | 0 |
| 32 | 2 | CUMGAINS | 1 | 1 |
| 33 | 2 | LIFT | 0 | 2 |
| 34 | 2 | LIFT | 1 | 1 |
| 35 | 2 | CUMLIFT | 0 | 2 |
| 36 | 2 | CUMLIFT | 1 | 1 |
| 37 | 1 | RANDOM_CUMGAINS | 0 | 0 |
| 38 | 1 | RANDOM_CUMGAINS | 1 | 1 |
| 39 | 1 | RANDOM_LIFT | 0 | 1 |
| 40 | 1 | RANDOM_LIFT | 1 | 1 |
| 41 | 1 | RANDOM_CUMLIFT | 0 | 1 |
| 42 | 1 | RANDOM_CUMLIFT | 1 | 1 |
| 43 | 1 | PERF_CUMGAINS | 0 | 0 |
| 44 | 1 | PERF_CUMGAINS | 1 | 1 |
| 45 | 1 | PERF_LIFT | 0 | 2 |
| 46 | 1 | PERF_LIFT | 1 | 0 |
| 47 | 1 | PERF_CUMLIFT | 0 | 2 |
| 48 | 1 | PERF_CUMLIFT | 1 | 1 |

PARTITION_TYPE:

Rows (28)

| | GROUP_ID | DATA_ID | TYPE |
|---|---|---|---|
| 1 | 2 | 1 | 1 |
| 2 | 2 | 4 | 1 |
| 3 | 2 | 5 | 1 |
| 4 | 2 | 6 | 1 |
| 5 | 2 | 8 | 1 |
| 6 | 2 | 9 | 1 |
| 7 | 2 | 10 | 1 |
| 8 | 2 | 11 | 1 |
| 9 | 2 | 12 | 1 |
| 10 | 2 | 13 | 1 |
| 11 | 2 | 2 | 2 |
| 12 | 2 | 3 | 2 |
| 13 | 2 | 7 | 2 |
| 14 | 2 | 14 | 2 |
| 15 | 1 | 1 | 1 |
| 16 | 1 | 4 | 1 |
| 17 | 1 | 5 | 1 |
| 18 | 1 | 6 | 1 |
| 19 | 1 | 8 | 1 |
| 20 | 1 | 9 | 1 |
| 21 | 1 | 10 | 1 |
| 22 | 1 | 11 | 1 |
| 23 | 1 | 12 | 1 |
| 24 | 1 | 13 | 1 |
| 25 | 1 | 2 | 2 |
| 26 | 1 | 3 | 2 |
| 27 | 1 | 7 | 2 |
| 28 | 1 | 14 | 2 |

## _SYS_AFL.PAL_UNIFIED_MASSIVE_CLASSIFICATION_PREDICT

Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA Table>` | IN |
| 2 | `<MODEL Table>` | IN |
| 3 | `<PARAMETER Table>` | IN |
| 4 | `<RESULT Table>` | OUT |
| 5 | `<ERRORMSG Table>` | OUT |
| 6 | `<PLACE_HOLDER1 Table>` | OUT |

Input Table

| Table | Column | Data Type | Description |
|---|---|---|---|
| DATA | 1st column | INTEGER, VRACHAR or NVARCHAR | Group id. |
| | 2nd column | INTEGER, VRACHAR or NVARCHAR | Record id. This column must be the first column in the table. |
| | Feature columns | INTEGER, DOUBLE, DECIMAL(p,s), VARCHAR, or NVARCHAR | Independent variables. |

| Table | Column | Data Type | Description |
|---|---|---|---|
| MODEL | 1st column | INTEGER, VRACHAR or NVARCHAR | Group id. |
| | 2nd column | INTEGER | Row index. |
| | 3rd column | INTEGER | Part index. |
| | 4th column | NCLOB | Model content. |
| PARAMETER | 1st column | INTEGER, VRACHAR or NVARCHAR | Group id. |
| | 2nd column | VARCHAR or NVARCHAR | Parameter name. |
| | 3rd column | INTEGER | Int value. |
| | 4th column | DOUBLE | Double value. |
| | 5th column | VARCHAR or NVARCHAR | String value. |

### Parameter Table

The parameter of the unified massive classification prediction execution function is the same as the unified classificatin prediction function.

See Parameter Tables [page 1495] for details about parameters for a specific algorithm.

### Output Tables

| Table | Column | Column Name | Column Data Type |
|---|---|---|---|
| RESULT | 1st column | GROUP_ID | Group id column data type. |
| | 2nd column | ID column name | ID column data type |
| | 3rd column | SCORE | NVARCHAR (256) |
| | 4th column | CONFIDENCE | DOUBLE |
| | 5th column | REASON CODE | NCLOB |
| ERRORMSG | 1st column | GROUP_ID | Group id column data type. |
| | 2nd column | ERROR_TIMESTAMP | VARCHAR, NVARCHAR |
| | 3rd column | ERRORCODE | INTEGER |
| | 4th column | MASSAGE | VARCHAR, NVARCHAR |
| PLCAE_HOLDER1 | 1st column | GROUP_ID | Group id column data type. |
| | 2nd column | OBJECT | VARCHAR, NVARCHAR |
| | 3rd column | KEY | VARCHAR, NVARCHAR |
| | 4th column | VALUE | VARCHAR, NVARCHAR |

### Example

Assume that:

- DM_PAL is a schema belonging to USER1;

- USER1 has been assigned the AFL__SYS_AFL_AFLPAL_EXECUTE or AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION role.

Example: Naive Bayes Prediction

```
-----------------------------------------Unified Massive Predict(Naive
Bayes)-------------------------------
DROP TABLE PAL_PREDICT_DATA_TBL;
CREATE COLUMN TABLE PAL_PREDICT_DATA_TBL (
    "GROUP_ID" INTEGER,
    "ID" INTEGER,
    "OUTLOOK" VARCHAR(20),
    "TEMP" DOUBLE,
    "HUMIDITY" DOUBLE,
    "WINDY" VARCHAR(10)
);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES (1, 0, 'Overcast', 75, -10000.0, 'Yes');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES (1, 1, 'Rain', 78, 70.0, 'Yes');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES (1, 2, 'Sunny', -10000.0, 80.0, 'Yes');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES (1, 3, 'Sunny', 69, 70.0, 'Yes');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES (1, 4, 'Rain', 69, 70.0, 'Yes');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES (1, 5, 'Rain', 70, 70.0, 'Yes');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES (1, 6, '***', 70, 70.0, 'Yes');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES (2, 0, 'Overcast', 75, -10000.0, 'Yes');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES (2, 1, 'Sunny', 78, 70.0, 'Yes');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES (2, 2, 'Rain', -10000.0, 80.0, 'Yes');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES (2, 3, 'Overcast', 69, 70.0, 'Yes');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES (2, 4, 'Overcast', 69, 70.0, 'Yes');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES (2, 5, 'Rain', 70, 70.0, 'Yes');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES (2, 6, '***', 70, 70.0, 'Yes');
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "GROUP_ID" INTEGER,
    "PARAM_NAME" VARCHAR (100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (100)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES (1, 'FUNCTION', NULL, NULL, 'NB');
INSERT INTO #PAL_PARAMETER_TBL VALUES (2, 'FUNCTION', NULL, NULL, 'NB');
CALL _SYS_AFL.PAL_UNIFIED_MASSIVE_CLASSIFICATION_PREDICT (PAL_PREDICT_DATA_TBL,
PAL_MODEL_RDT_TBL, #PAL_PARAMETER_TBL, ?, ?, ?);
```

**Expected Result**:

RESULT:

Rows (14)

| | GROUP_ID | ID | SCORE | CONFIDENCE | REASON_CODE |
|---|---|---|---|---|---|
| 1 | 2 | 0 | Play | -1.7976931348623157e+308 | [{"attr":"HUMIDITY","val":-0.2957982994410074 6,"pct":82.0},{"attr":"TEMP","val":0.04490718144996038,"pct":12.4},{"attr":"OUTLOOK","val":-0.013336951191 |
| 2 | 2 | 1 | Play | -9.381066567644327 | [{"attr":"OUTLOOK","val":-0.31741285057225529,"pct":62.300000000000007},{"attr":"TEMP","val":0.16589902081896114,"pct":32.6},{"attr":"HUMIDITY","val": |
| 3 | 2 | 2 | Play | -1.7976931348623157e+308 | [{"attr":"TEMP","val":-0.2937424110501555,"pct":89.6},{"attr":"OUTLOOK","val":0.01900515297741838,"pct":5.8},{"attr":"HUMIDITY","val":0.011173332560985 |
| 4 | 2 | 3 | Play | -8.664574415665168 | [{"attr":"OUTLOOK","val":-0.03893718512816003,"pct":67.2},{"attr":"TEMP","val":-0.012890832097333192,"pct":22.3},{"attr":"HUMIDITY","val":-0.0057123424 |
| 5 | 2 | 4 | Play | -8.664574415665168 | [{"attr":"OUTLOOK","val":-0.03893718512816003,"pct":67.2},{"attr":"TEMP","val":-0.012890832097333192,"pct":22.3},{"attr":"HUMIDITY","val":-0.0057123424 |
| 6 | 2 | 5 | Play | -8.199462384314144 | [{"attr":"OUTLOOK","val":0.03213622015127733,"pct":81.39999999999999},{"attr":"HUMIDITY","val":-0.005338041622461073,"pct":13.5},{"attr":"TEMP","val" |
| 7 | 2 | 6 | Play | -8.017140827520189 | [{"attr":"OUTLOOK","val":-0.01630872404244202,"pct":67.8},{"attr":"HUMIDITY","val":-0.00566205546226548,"pct":23.5},{"attr":"TEMP","val":0.001514491084 |
| 8 | 1 | 0 | Play | -1.7976931348623157e+308 | [{"attr":"HUMIDITY","val":-0.17185154271957465,"pct":37.800000000000007},{"attr":"OUTLOOK","val":-0.1328893234418731,"pct":29.2},{"attr":"TEMP","val": |
| 9 | 1 | 1 | Play | -8.366969411571406 | [{"attr":"TEMP","val":0.26028401967501049,"pct":76.8},{"attr":"WINDY","val":-0.06487245954295358,"pct":19.200000000000004},{"attr":"HUMIDITY","val":-0. |
| 10 | 1 | 2 | Play | -1.7976931348623157e+308 | [{"attr":"TEMP","val":-0.18060021417564565,"pct":63.9},{"attr":"WINDY","val":-0.06217965555685 6018,"pct":22.0},{"attr":"OUTLOOK","val":-0.0325116584834 |
| 11 | 1 | 3 | Do not Play | -8.832980863977898 | [{"attr":"WINDY","val":0.13314881048803896,"pct":59.6},{"attr":"TEMP","val":0.0509958214590252 1,"pct":22.8},{"attr":"OUTLOOK","val":0.0364713146997301 |
| 12 | 1 | 4 | Play | -8.521859785352767 | [{"attr":"WINDY","val":-0.1247092521281075,"pct":60.9},{"attr":"TEMP","val":-0.04420417805455902,"pct":21.6},{"attr":"OUTLOOK","val":0.033560264442095 |
| 13 | 1 | 5 | Play | -8.298108591331102 | [{"attr":"WINDY","val":-0.11613323600617427,"pct":70.7},{"attr":"OUTLOOK","val":0.03033673186936 7928,"pct":18.5},{"attr":"TEMP","val":0.01332361997809 |
| 14 | 1 | 6 | Play | -8.11578703453715 | [{"attr":"WINDY","val":-0.1229409859479081 2,"pct":81.5},{"attr":"OUTLOOK","val":-0.01727486264163084 5,"pct":11.4},{"attr":"TEMP","val":0.0100815830088 |

# _SYS_AFL.PAL_UNIFIED_MASSIVE_CLASSIFICATION_SCORE

## Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA Table>` | IN |
| 2 | `<MODEL Table>` | IN |
| 3 | `<PARAMETER Table>` | IN |
| 4 | `<RESULT Table>` | OUT |
| 5 | `<STATISTIC Table>` | OUT |
| 6 | `<CONFUSION_MATRIX Table>` | OUT |
| 7 | `<METRICS Tabl>` | OUT |
| 8 | `<ERRORMSG Table>` | OUT |

## Input Table

| Table | Column | Data Type | Description |
|---|---|---|---|
| DATA | 1st column | INTEGER, VRACHAR or NVARCHAR | Group id. |
| | 2nd column | INTEGER, VRACHAR or NVARCHAR | Record id. This column must be the first column in the table. |
| | Feature columns | INTEGER, DOUBLE, DECIMAL(p,s), VARCHAR, or NVARCHAR | Independent variables. |
| | LABEL column | INTEGER, VARCHAR, or NVARCHAR | Dependent variable. |
| MODEL | 1st column | INTEGER, VRACHAR or NVARCHAR | Group id. |
| | 2nd column | INTEGER | Row index. |
| | 3rd column | INTEGER | Part index. |
| | 4th column | NCLOB | Model content. |
| PARAMETER | 1st column | INTEGER, VRACHAR or NVARCHAR | Group id. |
| | 2nd column | VARCHAR or NVARCHAR | Parameter name. |
| | 3rd column | INTEGER | Int value. |
| | 4th column | DOUBLE | Double value. |
| | 5th column | VARCHAR or NVARCHAR | String value. |

## Parameter Table

The parameter of the unified massive classification scoring execution function is the same as the unified classificatin scoring function.

See Parameter Tables [page 1495] for details about parameters for a specific algorithm.

### Output Tables

| Table | Column | Column Name | Column Data Type |
|---|---|---|---|
| RESULT | 1st column | GROUP_ID | Group id column data type. |
| | 2nd column | ID column name | ID column data type |
| | 3rd column | SCORE | NVARCHAR (256) |
| | 4th column | CONFIDENCE | DOUBLE |
| | 5th column | REASON CODE | NCLOB |
| STATISTICS | 1st column | GROUP_ID | Group id column data type. |
| | 2nd column | STAT_NAME | NVARCHAR (256) |
| | 3rd column | STAT_VALUE | NVARCHAR (1000) |
| | 4th column | CLASS_NAME | NVARCHAR (256) |
| CONFUSION_MATRIX | 1st column | GROUP_ID | Group id column data type. |
| | 2nd column | ACTUAL_CLASS | NVARCHAR (1000) |
| | 3rd column | PREDICTED_CLASS | NVARCHAR (1000) |
| | 4th column | COUNT | INTEGER |
| METRICS | 1st column | GROUP_ID | Group id column data type. |
| | 2nd column | NAME | NVARCHAR (256) |
| | 3rd column | X | DOUBLE |
| | 4th column | Y | DOUBLE |
| ERRORMSG | 1st column | GROUP_ID | Group id column data type. |
| | 2nd column | ERROR_TIMESTAMP | VARCHAR, NVARCHAR |
| | 3rd column | ERRORCODE | INTEGER |
| | 4th column | MASSAGE | VARCHAR, NVARCHAR |

### Example

Assume that:

- DM_PAL is a schema belonging to USER1;
- USER1 has been assigned the AFL__SYS_AFL_AFLPAL_EXECUTE or AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION role.

Example: Naive Bayes Scoring

```
-------------------------------------------Unified Massive Scoring(Naive
Bayes)--------------------------------
DROP TABLE PAL_SCORING_DATA_TBL;
CREATE COLUMN TABLE PAL_SCORING_DATA_TBL (
    "GROUP_ID" INTEGER,
    "ID" INTEGER,
    "OUTLOOK" NVARCHAR(20),
    "TEMP" DOUBLE,
    "HUMIDITY" DOUBLE,
    "WINDY" NVARCHAR(10),
```

```
    "CLASS" NVARCHAR(20)
);
INSERT INTO PAL_SCORING_DATA_TBL VALUES (1, 0, 'Overcast', 75, -10000.0,
'Yes','Play');
INSERT INTO PAL_SCORING_DATA_TBL VALUES (1, 1, 'Rain', 78, 70.0, 'Yes','Play');
INSERT INTO PAL_SCORING_DATA_TBL VALUES (1, 2, 'Sunny', 63, 70.0, 'Yes', 'Do not
Play');
INSERT INTO PAL_SCORING_DATA_TBL VALUES (1, 3, 'Sunny', 69, 70.0, 'Yes','Do not
Play');
INSERT INTO PAL_SCORING_DATA_TBL VALUES (1, 4, 'Rain', 69, 70.0, 'Yes','Play');
INSERT INTO PAL_SCORING_DATA_TBL VALUES (1, 5, 'Rain', 70, 70.0, 'Yes','Do not
Play');
INSERT INTO PAL_SCORING_DATA_TBL VALUES (1, 6, '***', 70, 70.0, 'Yes','Play');
INSERT INTO PAL_SCORING_DATA_TBL VALUES (2, 0, 'Overcast', 75, -10000.0,
'Yes','Play');
INSERT INTO PAL_SCORING_DATA_TBL VALUES (2, 1, 'Sunny', 78, 70.0, 'Yes','Play');
INSERT INTO PAL_SCORING_DATA_TBL VALUES (2, 2, 'Rain', 63, 70.0, 'Yes', 'Do not
Play');
INSERT INTO PAL_SCORING_DATA_TBL VALUES (2, 3, 'Overcast', 69, 70.0, 'Yes','Do
not Play');
INSERT INTO PAL_SCORING_DATA_TBL VALUES (2, 4, 'Overcast', 69, 70.0,
'Yes','Play');
INSERT INTO PAL_SCORING_DATA_TBL VALUES (2, 5, 'Rain', 70, 70.0, 'Yes','Do not
Play');
INSERT INTO PAL_SCORING_DATA_TBL VALUES (2, 6, '***', 70, 70.0, 'Yes','Play');
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "GROUP_ID" INTEGER,
    "PARAM_NAME" VARCHAR (100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (100)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES (1, 'FUNCTION', NULL, NULL, 'NB');
INSERT INTO #PAL_PARAMETER_TBL VALUES (2, 'FUNCTION', NULL, NULL, 'NB');
CALL _SYS_AFL.PAL_UNIFIED_MASSIVE_CLASSIFICATION_SCORE (PAL_SCORING_DATA_TBL,
PAL_MODEL_RDT_TBL, #PAL_PARAMETER_TBL, ?, ?, ?, ?, ?);
```

**Expected Result**:

RESULT:

Rows (14)

| | GROUP_ID | ID | SCORE | CONFIDENCE | REASON_CODE |
|---|---|---|---|---|---|
| 1 | 1 | 0 | Play | -1.7976931348623157e+308 | [{"attr":"HUMIDITY","val":-0.17185154271957465,"pct":37.80000000000007},{"attr":"OUTLOOK","val":-0.1328893234418731,"pct":29.2},{"attr":"TEMP","val": |
| 2 | 1 | 1 | Play | -8.366969411571406 | [{"attr":"TEMP","val":0.26028401967501049,"pct":76.8},{"attr":"WINDY","val":-0.06487245954295358,"pct":19.200000000000004},{"attr":"HUMIDITY","val":-0. |
| 3 | 1 | 2 | Do not Play | -9.546640352921772 | [{"attr":"TEMP","val":0.405428674438994,"pct":71.3},{"attr":"WINDY","val":0.13311266600008549,"pct":23.4},{"attr":"OUTLOOK","val":0.028802531878711587 |
| 4 | 1 | 3 | Do not Play | -8.832980863977898 | [{"attr":"WINDY","val":0.1331488104880389 6,"pct":59.6},{"attr":"TEMP","val":0.05099582145902521,"pct":22.8},{"attr":"OUTLOOK","val":0.0364713146997303 |
| 5 | 1 | 4 | Play | -8.521859785352767 | [{"attr":"WINDY","val":-0.1247092521281075,"pct":60.9},{"attr":"TEMP","val":-0.0442041780545590 2,"pct":21.6},{"attr":"OUTLOOK","val":0.033560264442095 |
| 6 | 1 | 5 | Play | -8.298108591331102 | [{"attr":"WINDY","val":-0.1161332360061742 7,"pct":70.7},{"attr":"OUTLOOK","val":0.03033673186936 7928,"pct":18.5},{"attr":"TEMP","val":0.01332361997809 |
| 7 | 1 | 6 | Play | -8.11578703453715 | [{"attr":"WINDY","val":-0.1229409859479 0812,"pct":81.5},{"attr":"OUTLOOK","val":-0.01727486264163 0845,"pct":11.4},{"attr":"TEMP","val":0.0100815830088 |
| 8 | 2 | 0 | Play | -1.7976931348623157e+308 | [{"attr":"HUMIDITY","val":-0.29579829944100746,"pct":82.0},{"attr":"TEMP","val":0.04490718144996038,"pct":12.4},{"attr":"OUTLOOK","val":-0.013336951191 |
| 9 | 2 | 1 | Play | -9.381066567644327 | [{"attr":"OUTLOOK","val":-0.31741285057225529,"pct":62.300000000000007},{"attr":"TEMP","val":0.16589902081896114,"pct":32.6},{"attr":"HUMIDITY","val": |
| 10 | 2 | 2 | Play | -8.94375509403373 | [{"attr":"TEMP","val":-0.06422437741826996,"pct":60.2},{"attr":"OUTLOOK","val":0.034883398092852718,"pct":32.7},{"attr":"HUMIDITY","val":-0.0063923188 |
| 11 | 2 | 3 | Play | -8.664574415665168 | [{"attr":"OUTLOOK","val":-0.03893718512816003,"pct":67.2},{"attr":"TEMP","val":-0.012890832097333192,"pct":22.3},{"attr":"HUMIDITY","val":-0.0057123424 |
| 12 | 2 | 4 | Play | -8.664574415665168 | [{"attr":"OUTLOOK","val":-0.03893718512816003,"pct":67.2},{"attr":"TEMP","val":-0.012890832097333192,"pct":22.3},{"attr":"HUMIDITY","val":-0.0057123424 |
| 13 | 2 | 5 | Play | -8.199462384314144 | [{"attr":"OUTLOOK","val":0.03213622015127733,"pct":81.39999999999999},{"attr":"HUMIDITY","val":-0.005338041622461073,"pct":13.5},{"attr":"TEMP","val" |
| 14 | 2 | 6 | Play | -8.017140827520189 | [{"attr":"OUTLOOK","val":-0.01630872404244202,"pct":67.8},{"attr":"HUMIDITY","val":-0.00566205546226548,"pct":23.5},{"attr":"TEMP","val":0.00151449108 |

## STATISTICS:

Rows (23)

| | GROUP_ID | STAT_NAME | STAT_VALUE | CLASS_NAME |
|---|---|---|---|---|
| 1 | 1 | AUC | 0.7142857142857143 | *NULL* |
| 2 | 1 | RECALL | 0.6666666666666666 | Do not Play |
| 3 | 1 | PRECISION | 1 | Do not Play |
| 4 | 1 | F1_SCORE | 0.8 | Do not Play |
| 5 | 1 | SUPPORT | 3 | Do not Play |
| 6 | 1 | RECALL | 1 | Play |
| 7 | 1 | PRECISION | 0.8 | Play |
| 8 | 1 | F1_SCORE | 0.888888888888889 | Play |
| 9 | 1 | SUPPORT | 4 | Play |
| 10 | 1 | ACCURACY | 0.8571428571428571 | *NULL* |
| 11 | 1 | KAPPA | 0.6956521739130435 | *NULL* |
| 12 | 1 | MCC | 0.7302967433402214 | *NULL* |
| 13 | 2 | AUC | 0.40816326530612246 | *NULL* |
| 14 | 2 | RECALL | 0 | Do not Play |
| 15 | 2 | PRECISION | 0 | Do not Play |
| 16 | 2 | F1_SCORE | 0 | Do not Play |
| 17 | 2 | SUPPORT | 3 | Do not Play |
| 18 | 2 | RECALL | 1 | Play |
| 19 | 2 | PRECISION | 0.5714285714285714 | Play |
| 20 | 2 | F1_SCORE | 0.7272727272727273 | Play |
| 21 | 2 | SUPPORT | 4 | Play |
| 22 | 2 | ACCURACY | 0.5714285714285714 | *NULL* |
| 23 | 2 | KAPPA | 0 | *NULL* |

## CONFUSION_MATRIX:

Rows (8)

| | GROUP_ID | ACTUAL_CLASS | PREDICTED_CLASS | COUNT |
|---|---|---|---|---|
| 1 | 1 | Do not Play | Do not Play | 2 |
| 2 | 1 | Do not Play | Play | 1 |
| 3 | 1 | Play | Do not Play | 0 |
| 4 | 1 | Play | Play | 4 |
| 5 | 2 | Do not Play | Do not Play | 0 |
| 6 | 2 | Do not Play | Play | 3 |
| 7 | 2 | Play | Do not Play | 0 |
| 8 | 2 | Play | Play | 4 |

## METRICS:

Rows (92)

| | GROUP_ID | NAME | X | Y |
|---|---|---|---|---|
| 1 | 1 | RANDOM_CUMGAINS | 0 | 0 |
| 2 | 1 | RANDOM_CUMGAINS | 1 | 1 |
| 3 | 1 | RANDOM_LIFT | 0 | 1 |
| 4 | 1 | RANDOM_LIFT | 1 | 1 |
| 5 | 1 | RANDOM_CUMLIFT | 0 | 1 |
| 6 | 1 | RANDOM_CUMLIFT | 1 | 1 |
| 7 | 1 | PERF_CUMGAINS | 0 | 0 |
| 8 | 1 | PERF_CUMGAINS | 1 | 1 |
| 9 | 1 | PERF_LIFT | 0 | 2 |
| 10 | 1 | PERF_LIFT | 1 | 0 |
| 11 | 1 | PERF_CUMLIFT | 0 | 2 |
| 12 | 1 | PERF_CUMLIFT | 1 | 1 |
| 13 | 1 | ROC_FPR | 0 | 0 |
| 14 | 1 | ROC_TPR | 0 | 0 |
| 15 | 1 | ROC_FPR | 1 | 0.14285714285714285 |
| 16 | 1 | ROC_TPR | 1 | 0 |
| 17 | 1 | ROC_FPR | 2 | 0.14285714285714285 |
| 18 | 1 | ROC_TPR | 2 | 0.14285714285714285 |
| 19 | 1 | ROC_FPR | 3 | 0.14285714285714285 |
| 20 | 1 | ROC_TPR | 3 | 0.2857142857142857 |
| 21 | 1 | ROC_FPR | 4 | 0.14285714285714285 |
| 22 | 1 | ROC_TPR | 4 | 0.42857142857142855 |
| 23 | 1 | ROC_FPR | 5 | 0.2857142857142857 |
| 24 | 1 | ROC_TPR | 5 | 0.42857142857142855 |
| 25 | 1 | ROC_FPR | 6 | 0.2857142857142857 |
| 26 | 1 | ROC_TPR | 6 | 0.5714285714285714 |
| 27 | 1 | ROC_FPR | 7 | 0.2857142857142857 |
| 28 | 1 | ROC_TPR | 7 | 0.7142857142857143 |

## 5.13.4 Unified Massive Classification with Pivoted Input Data

The unified massive classification with pivoted execution function is a unified interface to call classification algorithms with pivot data format.

The unified massive classification with pivoted algorithms include:

- DT (Decision Trees)
- HGBT (Hybrid gradient boosting tree)
- LOGR (Logistic regression)
- M_LOGR (Multi-class logistic regression)
- NB (Naïve Bayes)
- RDT (Random decision trees)
- SVM (Support Vector Machine)

The keyword group name ('PAL_MASSIVE_PROCESSING_SPECIAL_GROUP_ID') is valid for all groups if the group name is not defined. For example, There are 2 groups ('A', 'B'),

```
INSERT INTO #PAL_PARAMETER_TBL VALUES
('PAL_MASSIVE_PROCESSING_SPECIAL_GROUP_ID', 'ALPHA', NULL, 1.0, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES
('PAL_MASSIVE_PROCESSING_SPECIAL_GROUP_ID', 'VERBOSE', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('B', 'VERBOSE', 0, NULL, NULL);
```

Therefore, A: ALPHA (1.0), VERBOSE (1); B: VERBOSE (0).

For more details on the feature of the unified massive classification with pivot, see the corresponding topics (Local Interpretability, Data Partition...) in the chapter on unified classification with pivot.

Moreover, it is clear that the models created from the unified massive classification with pivot and the unified classification with pivot are incompatible.

### Prerequisites

- All prerequisites for the unified classification with pivot function apply in the unified massive classification with pivot execution interface.

### _SYS_AFL.PAL_UNIFIED_MASSIVE_CLASSIFICATION_PIVOT

#### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<METADATA Table>` | IN |
| 2 | `<DATA Table>` | IN |
| 3 | `<PARAMETER Table>` | IN |

| Position | Table Name | Parameter Type |
|---|---|---|
| 4 | `<MODEL Table>` | OUT |
| 5 | `<VARIABLE_IMPORTANCE Table>` | OUT |
| 6 | `<STATISTICS Table>` | OUT |
| 7 | `<OPTIMAL_PARAM Table>` | OUT |
| 8 | `<CONFUSION_MATRIX Table>` | OUT |
| 9 | `<METRICS Table>` | OUT |
| 10 | `<PARTITION_TYPE Table>` | OUT |
| 11 | `<ERRORMSG Table>` | OUT |
| 12 | `<PLACE_HOLDER1 Table>` | OUT |

### Input Table

| Table | Column | Data Type | Description |
|---|---|---|---|
| METADATA | 1st column | VARCHAR or NVARCHAR | The name of the variable. |
| | 2nd column | VARCHAR or NVARCHAR | The type of the variable:<br>• CONTINUOUS;<br>• CATEGORICAL;<br>• TARGET. |
| DATA | 1st column | INTEGER, VRACHAR or NVARCHAR | Group id. |
| | 2nd column | INTEGER, VRACHAR or NVARCHAR | Record id. |
| | 3rd column | VARCHAR or NVARCHAR | Variable name. |
| | 4th column | VARCHAR or NVARCHAR | Variable value. |
| | 5th column | INTEGER | Self-defined data partition:<br>• 1:training<br>• 2:validation |
| PARAMETER | 1st column | INTEGER, VRACHAR or NVARCHAR | Group id. |
| | 2nd column | VARCHAR or NVARCHAR | Parameter name. |
| | 3rd column | INTEGER | Int value. |
| | 4th column | DOUBLE | Double value. |
| | 5th column | VARCHAR or NVARCHAR | String value. |

### Parameter Table

Mandatory Parameters

The following parameters are mandatory and must be given a value.

| Name | Data Type | Description |
|---|---|---|
| FUNCTION | VARCHAR or NVARCHAR | The algorithm name, including:<br>• DT;<br>• HGBT;<br>• LOGR;<br>• M_LOGR;<br>• NB;<br>• RDT;<br>• SVM. |

Optional Parameters

The parameter of the unified massive classification execution function with pivot is the same as the unified classification function with pivot.

See Parameter Tables [page 1681] for details about parameters for a specific algorithm.

**Output Tables**

| Table | Column | Column Name | Column Data Type |
|---|---|---|---|
| MODEL | 1st column | GROUP_ID | Group id column data type. |
| | 2nd column | ROW_INDEX | INTEGER |
| | 3rd column | PART_INDEX | INTEGER |
| | 4th column | MODEL_CONTENT | NCLOB |
| VARIABLE_IMPORTANCE | 1st column | GROUP_ID | Group id column data type. |
| | 2nd column | VARIABLE_NAME | NVARCHAR(256) |
| | 3rd column | IMPORTANCE | DOUBLE |
| STATISTICS | 1st column | GROUP_ID | Group id column data type. |
| | 2nd column | STAT_NAME | NVARCHAR (256) |
| | 3rd column | STAT_VALUE | NVARCHAR (1000) |
| | 4th column | CLASS_NAME | NVARCHAR (256) |
| OPTIMAL_PARAM | 1st column | GROUP_ID | Group id column data type. |
| | 2nd column | PARAM_NAME | NVARCHAR (256) |
| | 3rd column | INT_VALUE | INTEGER |
| | 4th column | DOUBLE_VALUE | DOUBLE |
| | 5th column | STRING_VALUE | NVARCHAR (1000) |
| CONFUSION_MATRIX | 1st column | GROUP_ID | Group id column data type. |
| | 2nd column | ACTUAL_CLASS | NVARCHAR (1000) |
| | 3rd column | PREDICTED_CLASS | NVARCHAR (1000) |
| | 4th column | COUNT | INTEGER |
| METRICS | 1st column | GROUP_ID | Group id column data type. |

| Table | Column | Column Name | Column Data Type |
|---|---|---|---|
| | 2nd column | NAME | NVARCHAR (256) |
| | 3rd column | X | DOUBLE |
| | 4th column | Y | DOUBLE |
| PARTITION_TYPE | 1st column | GROUP_ID | Group id column data type. |
| | 2nd column | ID column name (if there is no ID column, the name of first column of data table.) | ID column data type (if there is no ID column, the type of first column of data table.) |
| | 3rd column | TYPE | INTEGER |
| ERRORMSG | 1st column | GROUP_ID | Group id column data type. |
| | 2nd column | ERROR_TIMESTAMP | VARCHAR, NVARCHAR |
| | 3rd column | ERRORCODE | INTEGER |
| | 4th column | MASSAGE | VARCHAR, NVARCHAR |
| PLCAE_HOLDER1 | 1st column | GROUP_ID | Group id column data type. |
| | 2nd column | OBJECT | VARCHAR, NVARCHAR |
| | 3rd column | KEY | VARCHAR, NVARCHAR |
| | 4th column | VALUE | VARCHAR, NVARCHAR |

### Example

Assume that:

- DM_PAL is a schema belonging to USER1;
- USER1 has been assigned the AFL__SYS_AFL_AFLPAL_EXECUTE or AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION role.

Example: Naive Bayes

```
SET SCHEMA DM_PAL;
-----------------------------------Unified Massive Classification Pivot (Naive
Bayes)--------------------------
DROP TABLE PAL_META_DATA_TBL;
CREATE COLUMN TABLE PAL_META_DATA_TBL(
    VARIABLE_NAME NVARCHAR(100),
    VARIABLE_TYPE NVARCHAR(100)
);
INSERT INTO PAL_META_DATA_TBL VALUES ('ATTR1', 'CONTINUOUS');
INSERT INTO PAL_META_DATA_TBL VALUES ('ATTR2', 'CONTINUOUS');
INSERT INTO PAL_META_DATA_TBL VALUES ('ATTR3', 'CATEGORICAL');
INSERT INTO PAL_META_DATA_TBL VALUES ('ATTR4', 'CATEGORICAL');
INSERT INTO PAL_META_DATA_TBL VALUES ('LABEL', 'TARGET');
DROP TABLE PAL_DATA_TBL;
CREATE COLUMN TABLE PAL_DATA_TBL (
    "GROUP_ID" NVARCHAR(50),
    "DATA_ID" INTEGER,
    "VARIABLE_NAME" NVARCHAR(100),
    "VAR_VALUE" NVARCHAR(100),
    "PURPOSE"  INTEGER
);
INSERT INTO PAL_DATA_TBL VALUES('1', 0,'ATTR1','1',2);
INSERT INTO PAL_DATA_TBL VALUES('1', 0,'ATTR2','10',2);
INSERT INTO PAL_DATA_TBL VALUES('1', 0,'ATTR3','100',2);
```

```
INSERT INTO PAL_DATA_TBL VALUES('1', 0,'ATTR4','A',2);
INSERT INTO PAL_DATA_TBL VALUES('1', 0,'LABEL','1',2);
INSERT INTO PAL_DATA_TBL VALUES('1', 1,'ATTR1',1.1,1);
INSERT INTO PAL_DATA_TBL VALUES('1', 1,'ATTR2',10.1,1);
INSERT INTO PAL_DATA_TBL VALUES('1', 1,'ATTR3',100.1,1);
INSERT INTO PAL_DATA_TBL VALUES('1', 1,'ATTR4','A',1);
INSERT INTO PAL_DATA_TBL VALUES('1', 1,'LABEL',1,1);
INSERT INTO PAL_DATA_TBL VALUES('1', 2,'ATTR1',1.2,1);
INSERT INTO PAL_DATA_TBL VALUES('1', 2,'ATTR2',10.2,1);
INSERT INTO PAL_DATA_TBL VALUES('1', 2,'ATTR3',100,1);
INSERT INTO PAL_DATA_TBL VALUES('1', 2,'ATTR4','A',1);
INSERT INTO PAL_DATA_TBL VALUES('1', 2,'LABEL',1,1);
INSERT INTO PAL_DATA_TBL VALUES('1', 3,'ATTR1',1.3,1);
INSERT INTO PAL_DATA_TBL VALUES('1', 3,'ATTR2',10.4,1);
INSERT INTO PAL_DATA_TBL VALUES('1', 3,'ATTR3',100,1);
INSERT INTO PAL_DATA_TBL VALUES('1', 3,'ATTR4','A',1);
INSERT INTO PAL_DATA_TBL VALUES('1', 3,'LABEL',1,1);
INSERT INTO PAL_DATA_TBL VALUES('1', 4,'ATTR1',1.2,1);
INSERT INTO PAL_DATA_TBL VALUES('1', 4,'ATTR2',10.3,1);
INSERT INTO PAL_DATA_TBL VALUES('1', 4,'ATTR3',100,1);
INSERT INTO PAL_DATA_TBL VALUES('1', 4,'ATTR4','AB',1);
INSERT INTO PAL_DATA_TBL VALUES('1', 4,'LABEL',1,1);
INSERT INTO PAL_DATA_TBL VALUES('1', 5,'ATTR1',4,1);
INSERT INTO PAL_DATA_TBL VALUES('1', 5,'ATTR2',40,1);
INSERT INTO PAL_DATA_TBL VALUES('1', 5,'ATTR3',400,1);
INSERT INTO PAL_DATA_TBL VALUES('1', 5,'ATTR4','AB',1);
INSERT INTO PAL_DATA_TBL VALUES('1', 5,'LABEL',2,1);
INSERT INTO PAL_DATA_TBL VALUES('1', 6,'ATTR1',4.1,2);
INSERT INTO PAL_DATA_TBL VALUES('1', 6,'ATTR2',40,2);
INSERT INTO PAL_DATA_TBL VALUES('1', 6,'ATTR3',400,2);
INSERT INTO PAL_DATA_TBL VALUES('1', 6,'ATTR4','AB',2);
INSERT INTO PAL_DATA_TBL VALUES('1', 6,'LABEL',2,2);
INSERT INTO PAL_DATA_TBL VALUES('1', 7,'ATTR1',4.2,1);
INSERT INTO PAL_DATA_TBL VALUES('1', 7,'ATTR2',40.2,1);
INSERT INTO PAL_DATA_TBL VALUES('1', 7,'ATTR3',400,1);
INSERT INTO PAL_DATA_TBL VALUES('1', 7,'ATTR4','AB',1);
INSERT INTO PAL_DATA_TBL VALUES('1', 7,'LABEL',2,1);
INSERT INTO PAL_DATA_TBL VALUES('1', 8,'ATTR1',4.3,1);
INSERT INTO PAL_DATA_TBL VALUES('1', 8,'ATTR2',40.4,1);
INSERT INTO PAL_DATA_TBL VALUES('1', 8,'ATTR3',400,1);
INSERT INTO PAL_DATA_TBL VALUES('1', 8,'ATTR4','AB',1);
INSERT INTO PAL_DATA_TBL VALUES('1', 8,'LABEL',2,1);
INSERT INTO PAL_DATA_TBL VALUES('1', 9,'ATTR1',4.2,1);
INSERT INTO PAL_DATA_TBL VALUES('1', 9,'ATTR2',40.3,1);
INSERT INTO PAL_DATA_TBL VALUES('1', 9,'ATTR3',400,1);
INSERT INTO PAL_DATA_TBL VALUES('1', 9,'ATTR4','AB',1);
INSERT INTO PAL_DATA_TBL VALUES('1', 9,'LABEL',2,1);
INSERT INTO PAL_DATA_TBL VALUES('1', 10,'ATTR1',9,1);
INSERT INTO PAL_DATA_TBL VALUES('1', 10,'ATTR2',90,1);
INSERT INTO PAL_DATA_TBL VALUES('1', 10,'ATTR3',900,1);
INSERT INTO PAL_DATA_TBL VALUES('1', 10,'ATTR4','B',1);
INSERT INTO PAL_DATA_TBL VALUES('1', 10,'LABEL',3,1);
INSERT INTO PAL_DATA_TBL VALUES('1', 11,'ATTR1',9.1,1);
INSERT INTO PAL_DATA_TBL VALUES('1', 11,'ATTR2',90.1,1);
INSERT INTO PAL_DATA_TBL VALUES('1', 11,'ATTR3',900,1);
INSERT INTO PAL_DATA_TBL VALUES('1', 11,'ATTR4','A',1);
INSERT INTO PAL_DATA_TBL VALUES('1', 11,'LABEL',3,1);
INSERT INTO PAL_DATA_TBL VALUES('1', 12,'ATTR1',9.2,2);
INSERT INTO PAL_DATA_TBL VALUES('1', 12,'ATTR2',90.2,2);
INSERT INTO PAL_DATA_TBL VALUES('1', 12,'ATTR3',900,2);
INSERT INTO PAL_DATA_TBL VALUES('1', 12,'ATTR4','B',2);
INSERT INTO PAL_DATA_TBL VALUES('1', 12,'LABEL',3,2);
INSERT INTO PAL_DATA_TBL VALUES('1', 13,'ATTR1',9.3,1);
INSERT INTO PAL_DATA_TBL VALUES('1', 13,'ATTR2',90.4,1);
INSERT INTO PAL_DATA_TBL VALUES('1', 13,'ATTR3',900,1);
INSERT INTO PAL_DATA_TBL VALUES('1', 13,'ATTR4','A',1);
INSERT INTO PAL_DATA_TBL VALUES('1', 13,'LABEL',3,1);
INSERT INTO PAL_DATA_TBL VALUES('1', 14,'ATTR1',9.2,1);
```

```
INSERT INTO PAL_DATA_TBL VALUES('1', 14,'ATTR2',90.3,1);
INSERT INTO PAL_DATA_TBL VALUES('1', 14,'ATTR3',900,1);
INSERT INTO PAL_DATA_TBL VALUES('1', 14,'ATTR4','A',1);
INSERT INTO PAL_DATA_TBL VALUES('1', 14,'LABEL',3,1);
INSERT INTO PAL_DATA_TBL VALUES('2', 0,'ATTR1','5',2);
INSERT INTO PAL_DATA_TBL VALUES('2', 0,'ATTR2','50',2);
INSERT INTO PAL_DATA_TBL VALUES('2', 0,'ATTR3','60',2);
INSERT INTO PAL_DATA_TBL VALUES('2', 0,'ATTR4','A',2);
INSERT INTO PAL_DATA_TBL VALUES('2', 0,'LABEL','1',2);
INSERT INTO PAL_DATA_TBL VALUES('2', 1,'ATTR1',10.1,1);
INSERT INTO PAL_DATA_TBL VALUES('2', 1,'ATTR2',1.1,1);
INSERT INTO PAL_DATA_TBL VALUES('2', 1,'ATTR3',70.1,1);
INSERT INTO PAL_DATA_TBL VALUES('2', 1,'ATTR4','A',1);
INSERT INTO PAL_DATA_TBL VALUES('2', 1,'LABEL',1,1);
INSERT INTO PAL_DATA_TBL VALUES('2', 2,'ATTR1',9.2,1);
INSERT INTO PAL_DATA_TBL VALUES('2', 2,'ATTR2',100.2,1);
INSERT INTO PAL_DATA_TBL VALUES('2', 2,'ATTR3',1,1);
INSERT INTO PAL_DATA_TBL VALUES('2', 2,'ATTR4','A',1);
INSERT INTO PAL_DATA_TBL VALUES('2', 2,'LABEL',1,1);
INSERT INTO PAL_DATA_TBL VALUES('2', 3,'ATTR1',3.3,1);
INSERT INTO PAL_DATA_TBL VALUES('2', 3,'ATTR2',8.4,1);
INSERT INTO PAL_DATA_TBL VALUES('2', 3,'ATTR3',100,1);
INSERT INTO PAL_DATA_TBL VALUES('2', 3,'ATTR4','A',1);
INSERT INTO PAL_DATA_TBL VALUES('2', 3,'LABEL',1,1);
INSERT INTO PAL_DATA_TBL VALUES('2', 4,'ATTR1',10.2,1);
INSERT INTO PAL_DATA_TBL VALUES('2', 4,'ATTR2',45.3,1);
INSERT INTO PAL_DATA_TBL VALUES('2', 4,'ATTR3',90,1);
INSERT INTO PAL_DATA_TBL VALUES('2', 4,'ATTR4','AB',1);
INSERT INTO PAL_DATA_TBL VALUES('2', 4,'LABEL',1,1);
INSERT INTO PAL_DATA_TBL VALUES('2', 5,'ATTR1',9,1);
INSERT INTO PAL_DATA_TBL VALUES('2', 5,'ATTR2',10,1);
INSERT INTO PAL_DATA_TBL VALUES('2', 5,'ATTR3',20,1);
INSERT INTO PAL_DATA_TBL VALUES('2', 5,'ATTR4','AB',1);
INSERT INTO PAL_DATA_TBL VALUES('2', 5,'LABEL',2,1);
INSERT INTO PAL_DATA_TBL VALUES('2', 6,'ATTR1',1.4,2);
INSERT INTO PAL_DATA_TBL VALUES('2', 6,'ATTR2',30,2);
INSERT INTO PAL_DATA_TBL VALUES('2', 6,'ATTR3',900,2);
INSERT INTO PAL_DATA_TBL VALUES('2', 6,'ATTR4','AB',2);
INSERT INTO PAL_DATA_TBL VALUES('2', 6,'LABEL',2,2);
INSERT INTO PAL_DATA_TBL VALUES('2', 7,'ATTR1',4.2,1);
INSERT INTO PAL_DATA_TBL VALUES('2', 7,'ATTR2',40.2,1);
INSERT INTO PAL_DATA_TBL VALUES('2', 7,'ATTR3',400,1);
INSERT INTO PAL_DATA_TBL VALUES('2', 7,'ATTR4','AB',1);
INSERT INTO PAL_DATA_TBL VALUES('2', 7,'LABEL',2,1);
INSERT INTO PAL_DATA_TBL VALUES('2', 8,'ATTR1',4.3,1);
INSERT INTO PAL_DATA_TBL VALUES('2', 8,'ATTR2',40.4,1);
INSERT INTO PAL_DATA_TBL VALUES('2', 8,'ATTR3',400,1);
INSERT INTO PAL_DATA_TBL VALUES('2', 8,'ATTR4','AB',1);
INSERT INTO PAL_DATA_TBL VALUES('2', 8,'LABEL',2,1);
INSERT INTO PAL_DATA_TBL VALUES('2', 9,'ATTR1',4.2,1);
INSERT INTO PAL_DATA_TBL VALUES('2', 9,'ATTR2',40.3,1);
INSERT INTO PAL_DATA_TBL VALUES('2', 9,'ATTR3',400,1);
INSERT INTO PAL_DATA_TBL VALUES('2', 9,'ATTR4','AB',1);
INSERT INTO PAL_DATA_TBL VALUES('2', 9,'LABEL',2,1);
INSERT INTO PAL_DATA_TBL VALUES('2', 10,'ATTR1',9,1);
INSERT INTO PAL_DATA_TBL VALUES('2', 10,'ATTR2',90,1);
INSERT INTO PAL_DATA_TBL VALUES('2', 10,'ATTR3',900,1);
INSERT INTO PAL_DATA_TBL VALUES('2', 10,'ATTR4','B',1);
INSERT INTO PAL_DATA_TBL VALUES('2', 10,'LABEL',3,1);
INSERT INTO PAL_DATA_TBL VALUES('2', 11,'ATTR1',9.1,1);
INSERT INTO PAL_DATA_TBL VALUES('2', 11,'ATTR2',90.1,1);
INSERT INTO PAL_DATA_TBL VALUES('2', 11,'ATTR3',900,1);
INSERT INTO PAL_DATA_TBL VALUES('2', 11,'ATTR4','A',1);
INSERT INTO PAL_DATA_TBL VALUES('2', 11,'LABEL',3,1);
INSERT INTO PAL_DATA_TBL VALUES('2', 12,'ATTR1',9.2,2);
INSERT INTO PAL_DATA_TBL VALUES('2', 12,'ATTR2',90.2,2);
INSERT INTO PAL_DATA_TBL VALUES('2', 12,'ATTR3',900,2);
INSERT INTO PAL_DATA_TBL VALUES('2', 12,'ATTR4','B',2);
```

```
INSERT INTO PAL_DATA_TBL VALUES('2', 12,'LABEL',3,2);
INSERT INTO PAL_DATA_TBL VALUES('2', 13,'ATTR1',9.3,1);
INSERT INTO PAL_DATA_TBL VALUES('2', 13,'ATTR2',90.4,1);
INSERT INTO PAL_DATA_TBL VALUES('2', 13,'ATTR3',900,1);
INSERT INTO PAL_DATA_TBL VALUES('2', 13,'ATTR4','A',1);
INSERT INTO PAL_DATA_TBL VALUES('2', 13,'LABEL',3,1);
INSERT INTO PAL_DATA_TBL VALUES('2', 14,'ATTR1',9.2,1);
INSERT INTO PAL_DATA_TBL VALUES('2', 14,'ATTR2',90.3,1);
INSERT INTO PAL_DATA_TBL VALUES('2', 14,'ATTR3',900,1);
INSERT INTO PAL_DATA_TBL VALUES('2', 14,'ATTR4','A',1);
INSERT INTO PAL_DATA_TBL VALUES('2', 14,'LABEL',3,1);
DROP TABLE PAL_PARAMETER_TBL;
CREATE  COLUMN TABLE PAL_PARAMETER_TBL (
    "GROUP_ID" NVARCHAR(50),
    "PARAM_NAME" VARCHAR (100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (100)
);
INSERT INTO PAL_PARAMETER_TBL VALUES ('PAL_MASSIVE_PROCESSING_SPECIAL_GROUP_ID',
'FUNCTION', NULL, NULL, 'NB');
INSERT INTO PAL_PARAMETER_TBL VALUES ('PAL_MASSIVE_PROCESSING_SPECIAL_GROUP_ID',
'ALPHA', NULL, 1.0, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('PAL_MASSIVE_PROCESSING_SPECIAL_GROUP_ID',
'PARTITION_METHOD', 2, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('PAL_MASSIVE_PROCESSING_SPECIAL_GROUP_ID',
'PARTITION_STRATIFIED_VARIABLE', NULL, NULL, 'LABEL');
INSERT INTO PAL_PARAMETER_TBL VALUES ('PAL_MASSIVE_PROCESSING_SPECIAL_GROUP_ID',
'PARTITION_TRAINING_PERCENT', NULL, 0.7, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('PAL_MASSIVE_PROCESSING_SPECIAL_GROUP_ID',
'OUTPUT_PARTITION_RESULT', 1, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('PAL_MASSIVE_PROCESSING_SPECIAL_GROUP_ID',
'BACKGROUND_SIZE', 4, NULL, NULL);
DROP TABLE PAL_MODEL_TBL;
CREATE COLUMN TABLE PAL_MODEL_TBL (
    "GROUP_ID" NVARCHAR(50),
    "ROW_INDEX" INTEGER,
    "TREE_INDEX" INTEGER,
    "MODEL_CONTENT" NVARCHAR(5000)
);
DROP TABLE PAL_IMP_TBL;
CREATE COLUMN TABLE PAL_IMP_TBL (
    "GROUP_ID" NVARCHAR(50),
    "VARIABLE_NAME" NVARCHAR(256),
    "IMPORTANCE" DOUBLE
);
DROP TABLE PAL_STAT_TBL;
CREATE COLUMN TABLE PAL_STAT_TBL (
    "GROUP_ID" NVARCHAR(50),
    "STAT_NAME" NVARCHAR(256),
    "STAT_VALUE" NVARCHAR(1000),
    "CLASS_NAME" NVARCHAR(256)
);
DROP TABLE PAL_OPT_PARAM_TBL;
CREATE  COLUMN TABLE PAL_OPT_PARAM_TBL (
    "GROUP_ID" NVARCHAR(50),
    "PARAM_NAME" VARCHAR (100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (100)
);
DROP TABLE PAL_CMATRIX_TBL;
CREATE COLUMN TABLE PAL_CMATRIX_TBL (
    "GROUP_ID" NVARCHAR(50),
    "ACTUAL_CLASS" NVARCHAR(256),
    "PREDICTED_CLASS" NVARCHAR(256),
    "COUNT" INTEGER
);
```

```
DROP TABLE PAL_METRICS_TBL;
CREATE COLUMN TABLE PAL_METRICS_TBL (
    "GROUP_ID" NVARCHAR(50),
    "NAME" NVARCHAR(256),
    "X" DOUBLE,
    "Y" DOUBLE
);
DROP TABLE PAL_PARTITION_TBL;
CREATE COLUMN TABLE PAL_PARTITION_TBL (
    "GROUP_ID" NVARCHAR(50),
    "ID" INTEGER,
    "TYPE" INTEGER
);
DO
BEGIN
    lt_meta_data = SELECT * FROM PAL_META_DATA_TBL;
    lt_data = SELECT * FROM PAL_DATA_TBL;
    lt_ctrl = SELECT * FROM PAL_PARAMETER_TBL;
    CALL _SYS_AFL.PAL_UNIFIED_MASSIVE_CLASSIFICATION_PIVOT
(:lt_meta_data, :lt_data, :lt_ctrl, lt_model, lt_imp,lt_stat,
lt_opt,lt_cm,lt_metrics,lt_partition, errormsg, lt_ph1);
    INSERT INTO PAL_MODEL_TBL SELECT * FROM :lt_model;
    INSERT INTO PAL_IMP_TBL SELECT * FROM :lt_imp;
    INSERT INTO PAL_STAT_TBL SELECT * FROM :lt_stat;
    INSERT INTO PAL_OPT_PARAM_TBL SELECT * FROM :lt_opt;
    INSERT INTO PAL_CMATRIX_TBL SELECT * FROM :lt_cm;
    INSERT INTO PAL_METRICS_TBL SELECT * FROM :lt_metrics;
    INSERT INTO PAL_PARTITION_TBL SELECT * FROM :lt_partition;
    select * from :errormsg;
END;
SELECT * FROM PAL_MODEL_TBL;
SELECT * FROM PAL_IMP_TBL;
SELECT * FROM PAL_STAT_TBL;
SELECT * FROM PAL_OPT_PARAM_TBL;
SELECT * FROM PAL_CMATRIX_TBL;
SELECT * FROM PAL_METRICS_TBL;
SELECT * from PAL_PARTITION_TBL;
```

**Expected Result**:

MODEL:

Rows (8)

| | GROUP_ID | ROW_INDEX | TREE_INDEX | |
|---|---|---|---|---|
| 1 | 2 | 0 | 0 | {"Laplace":1.0,"ColNumb":4,"ClassSize":3,"Class":[{"ClassName":"1","TransValue":0,"PriorFreq":3},{"ClassName":"2","TransValue":1,"PriorFreq":3},{"ClassName":"3","TransValue" |
| 2 | 2 | -1 | 1 | NB |
| 3 | 2 | -2 | 2 | {"feature":["ATTR1","ATTR2","ATTR3","ATTR4","LABEL"],"featureNum":5,"index":[0,1,2,3,4],"type":["CONTINUOUS","CONTINUOUS","CATEGORICAL","CATEGORICAL","CATEGOR |
| 4 | 2 | -10 | 3 | {"cate_vals":[{"cateVal":["90","900","900"]},{"cateVal":["AB","B","A"]},{"cateVal":["1","3","3"]}],"categorical_size":3,"class_index":5,"cont_vals":[{"contVal":[10.199999999999999,9.1 |
| 5 | 1 | 0 | 0 | {"Laplace":1.0,"ColNumb":4,"ClassSize":3,"Class":[{"ClassName":"1","TransValue":0,"PriorFreq":3},{"ClassName":"2","TransValue":1,"PriorFreq":3},{"ClassName":"3","TransValue" |
| 6 | 1 | -1 | 1 | NB |
| 7 | 1 | -2 | 2 | {"feature":["ATTR1","ATTR2","ATTR3","ATTR4","LABEL"],"featureNum":5,"index":[0,1,2,3,4],"type":["CONTINUOUS","CONTINUOUS","CATEGORICAL","CATEGORICAL","CATEGOR |
| 8 | 1 | -10 | 3 | {"cate_vals":[{"cateVal":["100","900","900"]},{"cateVal":["AB","B","A"]},{"cateVal":["1","3","3"]}],"categorical_size":3,"class_index":5,"cont_vals":[{"contVal":[1.2,9.199999999999999 |

STATISTIC:

Rows (32)

| | GROUP_ID | STAT_NAME | STAT_VALUE | CLASS_NAME |
|---|---|---|---|---|
| 1 | 2 | AUC | 1 | *NULL* |
| 2 | 2 | RECALL | 1 | 1 |
| 3 | 2 | PRECISION | 1 | 1 |
| 4 | 2 | F1_SCORE | 1 | 1 |
| 5 | 2 | SUPPORT | 2 | 1 |
| 6 | 2 | RECALL | 1 | 2 |
| 7 | 2 | PRECISION | 1 | 2 |
| 8 | 2 | F1_SCORE | 1 | 2 |
| 9 | 2 | SUPPORT | 2 | 2 |
| 10 | 2 | RECALL | 1 | 3 |
| 11 | 2 | PRECISION | 1 | 3 |
| 12 | 2 | F1_SCORE | 1 | 3 |
| 13 | 2 | SUPPORT | 1 | 3 |
| 14 | 2 | ACCURACY | 1 | *NULL* |
| 15 | 2 | KAPPA | 1 | *NULL* |
| 16 | 2 | MCC | 1 | *NULL* |
| 17 | 1 | AUC | 1 | *NULL* |
| 18 | 1 | RECALL | 1 | 1 |
| 19 | 1 | PRECISION | 1 | 1 |
| 20 | 1 | F1_SCORE | 1 | 1 |
| 21 | 1 | SUPPORT | 2 | 1 |
| 22 | 1 | RECALL | 1 | 2 |
| 23 | 1 | PRECISION | 1 | 2 |
| 24 | 1 | F1_SCORE | 1 | 2 |
| 25 | 1 | SUPPORT | 2 | 2 |
| 26 | 1 | RECALL | 1 | 3 |
| 27 | 1 | PRECISION | 1 | 3 |
| 28 | 1 | F1_SCORE | 1 | 3 |

CONFUSION MATRIX:

Rows (18)

| | GROUP_ID | ACTUAL_CLASS | PREDICTED_CLASS | COUNT |
|---|---|---|---|---|
| 1 | 2 | 1 | 1 | 2 |
| 2 | 2 | 1 | 2 | 0 |
| 3 | 2 | 1 | 3 | 0 |
| 4 | 2 | 2 | 1 | 0 |
| 5 | 2 | 2 | 2 | 2 |
| 6 | 2 | 2 | 3 | 0 |
| 7 | 2 | 3 | 1 | 0 |
| 8 | 2 | 3 | 2 | 0 |
| 9 | 2 | 3 | 3 | 1 |
| 10 | 1 | 1 | 1 | 2 |
| 11 | 1 | 1 | 2 | 0 |
| 12 | 1 | 1 | 3 | 0 |
| 13 | 1 | 2 | 1 | 0 |
| 14 | 1 | 2 | 2 | 2 |
| 15 | 1 | 2 | 3 | 0 |
| 16 | 1 | 3 | 1 | 0 |
| 17 | 1 | 3 | 2 | 0 |
| 18 | 1 | 3 | 3 | 1 |

METRICS:

Rows (68)

| | GROUP_ID | NAME | X | Y |
|---|---|---|---|---|
| 31 | 2 | ROC_FPR | 9 | 0.4 |
| 32 | 2 | ROC_TPR | 9 | 1 |
| 33 | 2 | ROC_FPR | 10 | 0.5 |
| 34 | 2 | ROC_TPR | 10 | 1 |
| 35 | 2 | ROC_FPR | 11 | 0.6 |
| 36 | 2 | ROC_TPR | 11 | 1 |
| 37 | 2 | ROC_FPR | 12 | 1 |
| 38 | 2 | ROC_TPR | 12 | 1 |
| 39 | 2 | CUMGAINS | 0 | 0 |
| 40 | 2 | CUMGAINS | 1 | 1 |
| 41 | 2 | LIFT | 0 | 3 |
| 42 | 2 | LIFT | 1 | 0 |
| 43 | 2 | CUMLIFT | 0 | 3 |
| 44 | 2 | CUMLIFT | 1 | 1 |
| 45 | 1 | RANDOM_CUMGAINS | 0 | 0 |
| 46 | 1 | RANDOM_CUMGAINS | 1 | 1 |
| 47 | 1 | RANDOM_LIFT | 0 | 1 |
| 48 | 1 | RANDOM_LIFT | 1 | 1 |
| 49 | 1 | RANDOM_CUMLIFT | 0 | 1 |
| 50 | 1 | RANDOM_CUMLIFT | 1 | 1 |
| 51 | 1 | PERF_CUMGAINS | 0 | 0 |
| 52 | 1 | PERF_CUMGAINS | 1 | 1 |
| 53 | 1 | PERF_LIFT | 0 | 3 |
| 54 | 1 | PERF_LIFT | 1 | 0 |
| 55 | 1 | PERF_CUMLIFT | 0 | 3 |
| 56 | 1 | PERF_CUMLIFT | 1 | 1 |
| 57 | 1 | ROC_FPR | 0 | 0 |
| 58 | 1 | ROC_TPR | 0 | 0 |

PARTITION_TYPE:

Rows (30)

| | GROUP_ID | ID | TYPE |
|---|---|---|---|
| 1 | 2 | 2 | 1 |
| 2 | 2 | 3 | 1 |
| 3 | 2 | 4 | 1 |
| 4 | 2 | 5 | 1 |
| 5 | 2 | 6 | 1 |
| 6 | 2 | 7 | 1 |
| 7 | 2 | 11 | 1 |
| 8 | 2 | 12 | 1 |
| 9 | 2 | 13 | 1 |
| 10 | 2 | 14 | 1 |
| 11 | 2 | 0 | 2 |
| 12 | 2 | 1 | 2 |
| 13 | 2 | 8 | 2 |
| 14 | 2 | 9 | 2 |
| 15 | 2 | 10 | 2 |
| 16 | 1 | 2 | 1 |
| 17 | 1 | 3 | 1 |
| 18 | 1 | 4 | 1 |
| 19 | 1 | 5 | 1 |
| 20 | 1 | 6 | 1 |
| 21 | 1 | 7 | 1 |
| 22 | 1 | 11 | 1 |
| 23 | 1 | 12 | 1 |
| 24 | 1 | 13 | 1 |
| 25 | 1 | 14 | 1 |
| 26 | 1 | 0 | 2 |
| 27 | 1 | 1 | 2 |
| 28 | 1 | 8 | 2 |

# _SYS_AFL.PAL_UNIFIED_MASSIVE_CLASSIFICATION_PREDICT_PIVOT

## Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA Table>` | IN |

| Position | Table Name | Parameter Type |
|---|---|---|
| 2 | `<MODEL Table>` | IN |
| 3 | `<PARAMETER Table>` | IN |
| 4 | `<RESULT Table>` | OUT |
| 5 | `<ERRORMSG Table>` | OUT |
| 6 | `<PLACE_HOLDER1 Table>` | OUT |

### Input Table

| Table | Column | Data Type | Description |
|---|---|---|---|
| DATA | 1st column | INTEGER, VRACHAR or NVARCHAR | Group id. |
| | 2nd column | INTEGER, VRACHAR or NVARCHAR | Record id. |
| | 3rd column | VARCHAR or NVARCHAR | Variable name. |
| | 4th column | VARCHAR or NVARCHAR | Variable value. |
| MODEL | 1st column | INTEGER, VRACHAR or NVARCHAR | Group id. |
| | 2nd column | INTEGER | Row index. |
| | 3rd column | INTEGER | Part index. |
| | 4th column | NCLOB | Model content. |
| PARAMETER | 1st column | INTEGER, VRACHAR or NVARCHAR | Group id. |
| | 2nd column | VARCHAR or NVARCHAR | Parameter name. |
| | 3rd column | INTEGER | Int value. |
| | 4th column | DOUBLE | Double value. |
| | 5th column | VARCHAR or NVARCHAR | String value. |

### Parameter Table

The parameter of the unified massive classification prediction with pivot execution function is the same as the unified classificatin prediction with pivot function.

See Parameter Tables [page 1681]for details about parameters for a specific algorithm.

### Output Tables

| Table | Column | Column Name | Column Data Type |
|---|---|---|---|
| RESULT | 1st column | GROUP_ID | Group id column data type. |
| | 2nd column | ID column name | ID column data type |
| | 3rd column | SCORE | NVARCHAR (256) |
| | 4th column | CONFIDENCE | DOUBLE |

| Table | Column | Column Name | Column Data Type |
|---|---|---|---|
| | 5th column | REASON CODE | NCLOB |
| ERRORMSG | 1st column | GROUP_ID | Group id column data type. |
| | 2nd column | ERROR_TIMESTAMP | VARCHAR, NVARCHAR |
| | 3rd column | ERRORCODE | INTEGER |
| | 4th column | MASSAGE | VARCHAR, NVARCHAR |
| PLCAE_HOLDER1 | 1st column | GROUP_ID | Group id column data type. |
| | 2nd column | OBJECT | VARCHAR, NVARCHAR |
| | 3rd column | KEY | VARCHAR, NVARCHAR |
| | 4th column | VALUE | VARCHAR, NVARCHAR |

### Example

Assume that:

- DM_PAL is a schema belonging to USER1;
- USER1 has been assigned the AFL__SYS_AFL_AFLPAL_EXECUTE or AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION role.

Example: Naive Bayes Prediction

```
-------------------------------------------Unified Massive Predict Pivot(Naive
Bayes)--------------------------------
DROP TABLE PAL_PREDICT_DATA_TBL;
CREATE COLUMN TABLE PAL_PREDICT_DATA_TBL (
    "GROUP_ID" NVARCHAR(50),
    "DATA_ID" INTEGER,
    "VARIABLE_NAME" NVARCHAR(100),
    "VAR_VALUE" NVARCHAR(100)
);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES('1', 0,'ATTR1','1');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES('1', 0,'ATTR2','10');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES('1', 0,'ATTR3','100');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES('1', 0,'ATTR4','A');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES('1', 1,'ATTR1',1.1);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES('1', 1,'ATTR2',10.1);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES('1', 1,'ATTR3',100.1);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES('1', 1,'ATTR4','A');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES('1', 2,'ATTR1',1.2);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES('1', 2,'ATTR2',10.2);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES('1', 2,'ATTR3',100);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES('1', 2,'ATTR4','A');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES('1', 3,'ATTR1',1.3);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES('1', 3,'ATTR2',10.4);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES('1', 3,'ATTR3',100);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES('1', 3,'ATTR4','A');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES('1', 4,'ATTR1',4);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES('1', 4,'ATTR2',40);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES('1', 4,'ATTR3',400);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES('1', 4,'ATTR4','AB');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES('1', 5,'ATTR1',9.2);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES('1', 5,'ATTR2',90.2);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES('1', 5,'ATTR3',900);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES('1', 5,'ATTR4','B');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES('2', 0,'ATTR1','5');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES('2', 0,'ATTR2','20');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES('2', 0,'ATTR3','300');
```

```
INSERT INTO PAL_PREDICT_DATA_TBL VALUES('2', 0,'ATTR4','A');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES('2', 1,'ATTR1',3.1);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES('2', 1,'ATTR2',100.1);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES('2', 1,'ATTR3',10.1);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES('2', 1,'ATTR4','A');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES('2', 2,'ATTR1',9.2);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES('2', 2,'ATTR2',12.2);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES('2', 2,'ATTR3',90);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES('2', 2,'ATTR4','A');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES('2', 3,'ATTR1',3.1);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES('2', 3,'ATTR2',3.4);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES('2', 3,'ATTR3',9);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES('2', 3,'ATTR4','A');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES('2', 4,'ATTR1',4);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES('2', 4,'ATTR2',40);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES('2', 4,'ATTR3',400);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES('2', 4,'ATTR4','AB');
INSERT INTO PAL_PREDICT_DATA_TBL VALUES('2', 5,'ATTR1',9.2);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES('2', 5,'ATTR2',90.2);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES('2', 5,'ATTR3',900);
INSERT INTO PAL_PREDICT_DATA_TBL VALUES('2', 5,'ATTR4','B');
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "GROUP_ID" NVARCHAR(50),
    "PARAM_NAME" VARCHAR (100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (100)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES
('PAL_MASSIVE_PROCESSING_SPECIAL_GROUP_ID', 'VERBOSE', 0, NULL, NULL);
CALL
_SYS_AFL.PAL_UNIFIED_MASSIVE_CLASSIFICATION_PREDICT_PIVOT(PAL_PREDICT_DATA_TBL,
PAL_MODEL_TBL, #PAL_PARAMETER_TBL, ?, ?, ?);
```

**Expected Result**:

RESULT:

Rows (12)

| | GROUP_ID | DATA_ID | SCORE | CONFIDENCE | REASON_CODE |
|---|---|---|---|---|---|
| 1 | 1 | 0 | 1 | -11.652426313037237 | [{"attr":"ATTR1","val":0.23333333333333337,"pct":46.7},{"attr":"ATTR2","val":0.23333333333333335,"pct":46.7},{"attr":"ATTR4","val":-0.01666666666666665,"p |
| 2 | 1 | 1 | 1 | -4.451028880593471 | [{"attr":"ATTR1","val":0.23333333333333337,"pct":46.7},{"attr":"ATTR2","val":0.23333333333333335,"pct":46.7},{"attr":"ATTR4","val":-0.01666666666666665,"p |
| 3 | 1 | 2 | 1 | 0.3473811467832276 | [{"attr":"ATTR1","val":0.23333333333333337,"pct":46.7},{"attr":"ATTR2","val":0.23333333333333335,"pct":46.7},{"attr":"ATTR4","val":-0.01666666666666665,"p |
| 4 | 1 | 3 | 1 | -0.1527193195604387 2 | [{"attr":"ATTR1","val":0.23333333333333337,"pct":46.7},{"attr":"ATTR2","val":0.23333333333333335,"pct":46.7},{"attr":"ATTR4","val":-0.01666666666666665,"p |
| 5 | 1 | 4 | 2 | -0.057784126045327344 | [{"attr":"ATTR2","val":0.666666666666666,"pct":66.69999999999999},{"attr":"ATTR1","val":0.3333333333333339,"pct":33.3},{"attr":"ATTR4","val":4.0029660 |
| 6 | 1 | 5 | 3 | 0.1340905653304243 | [{"attr":"ATTR2","val":0.23333333333333335,"pct":46.7},{"attr":"ATTR1","val":0.2333333333333332,"pct":46.7},{"attr":"ATTR4","val":-0.0166666666666665,"p |
| 7 | 2 | 0 | 2 | -10.212584367777902 | [{"attr":"ATTR2","val":0.4047948277681892,"pct":51.8},{"attr":"ATTR4","val":-0.19884970169113196,"pct":25.5},{"attr":"ATTR1","val":0.14684175742650375,"p |
| 8 | 2 | 1 | 1 | -11.363384408643883 | [{"attr":"ATTR2","val":0.5223332183941376,"pct":58.9},{"attr":"ATTR1","val":0.22453347400888708,"pct":25.3},{"attr":"ATTR4","val":0.12998047241516778,"pc |
| 9 | 2 | 2 | 1 | -10.842144660940724 | [{"attr":"ATTR2","val":0.2049103470997974,"pct":42.7},{"attr":"ATTR4","val":0.18175166548328115,"pct":37.9},{"attr":"ATTR3","val":0.05140523416437795,"pc |
| 10 | 2 | 3 | 1 | -11.342991237179948 | [{"attr":"ATTR4","val":0.255543507790645,"pct":63.1},{"attr":"ATTR1","val":0.0925517162680886 4,"pct":22.8},{"attr":"ATTR2","val":0.04232398479324221,"pct" |
| 11 | 2 | 4 | 2 | -9.428038024603229 | [{"attr":"ATTR2","val":0.4094958398071168,"pct":58.0},{"attr":"ATTR1","val":0.12843257808905135,"pct":18.200000000000004},{"attr":"ATTR3","val":0.126209 |
| 12 | 2 | 5 | 3 | -0.22258437860830815 | [{"attr":"ATTR1","val":0.19999665143812787,"pct":40.0},{"attr":"ATTR2","val":0.19999624786912024,"pct":40.0},{"attr":"ATTR4","val":-0.05000018165997264,"p |

# _SYS_AFL.PAL_UNIFIED_MASSIVE_CLASSIFICATION_SCORE_PIVOT

## Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA Table>` | IN |
| 2 | `<MODEL Table>` | IN |

| Position | Table Name | Parameter Type |
|---|---|---|
| 3 | `<PARAMETER Table>` | IN |
| 4 | `<RESULT Table>` | OUT |
| 5 | `<STATISTIC Table>` | OUT |
| 6 | `<CONFUSION_MATRIX Table>` | OUT |
| 7 | `<METRICS Table>` | OUT |
| 8 | `<ERRORMSG Table>` | OUT |

### Input Table

| Table | Column | Data Type | Description |
|---|---|---|---|
| DATA | 1st column | INTEGER, VRACHAR or NVARCHAR | Group id. |
| | 2nd column | INTEGER, VRACHAR or NVARCHAR | Record ID. |
| | 3rd column | VARCHAR or NVARCHAR | Variable name. |
| | 4th column | VARCHAR or NVARCHAR | Variable value. |
| MODEL | 1st column | INTEGER, VRACHAR or NVARCHAR | Group id. |
| | 2nd column | INTEGER | Row index. |
| | 3rd column | INTEGER | Part index. |
| | 4th column | NCLOB | Model content. |
| PARAMETER | 1st column | INTEGER, VRACHAR or NVARCHAR | Group id. |
| | 2nd column | VARCHAR or NVARCHAR | Parameter name. |
| | 3rd column | INTEGER | Int value. |
| | 4th column | DOUBLE | Double value. |
| | 5th column | VARCHAR or NVARCHAR | String value. |

### Parameter Table

The parameter of the unified massive classification scoring with pivot execution function is the same as the unified classificatin scoring with pivot function.

See Parameter Tables [page 1681] for details about parameters for a specific algorithm.

### Output Tables

| Table | Column | Column Name | Column Data Type |
|---|---|---|---|
| RESULT | 1st column | GROUP_ID | Group id column data type. |
| | 2nd column | ID column name | ID column data type |
| | 3rd column | SCORE | NVARCHAR (256) |

| Table | Column | Column Name | Column Data Type |
|---|---|---|---|
| | 4th column | CONFIDENCE | DOUBLE |
| | 5th column | REASON CODE | NCLOB |
| STATISTICS | 1st column | GROUP_ID | Group id column data type. |
| | 2nd column | STAT_NAME | NVARCHAR (256) |
| | 3rd column | STAT_VALUE | NVARCHAR (1000) |
| | 4th column | CLASS_NAME | NVARCHAR (256) |
| CONFUSION_MATRIX | 1st column | GROUP_ID | Group id column data type. |
| | 2nd column | ACTUAL_CLASS | NVARCHAR (1000) |
| | 3rd column | PREDICTED_CLASS | NVARCHAR (1000) |
| | 4th column | COUNT | INTEGER |
| METRICS | 1st column | GROUP_ID | Group id column data type. |
| | 2nd column | NAME | NVARCHAR (256) |
| | 3rd column | X | DOUBLE |
| | 4th column | Y | DOUBLE |
| ERRORMSG | 1st column | GROUP_ID | Group id column data type. |
| | 2nd column | ERROR_TIMESTAMP | VARCHAR, NVARCHAR |
| | 3rd column | ERRORCODE | INTEGER |
| | 4th column | MASSAGE | VARCHAR, NVARCHAR |

### Example

Assume that:

- DM_PAL is a schema belonging to USER1;
- USER1 has been assigned the AFL__SYS_AFL_AFLPAL_EXECUTE or AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION role.

Example: Naive Bayes Scoring

```
-------------------------------------------Unified Massive Scoring Pivot(Naive
Bayes)--------------------------------
DROP TABLE PAL_SCORING_DATA_TBL;
CREATE COLUMN TABLE PAL_SCORING_DATA_TBL (
    "GROUP_ID" NVARCHAR(50),
    "DATA_ID" INTEGER,
    "VARIABLE_NAME" NVARCHAR(100),
    "VAR_VALUE" NVARCHAR(100)
);
INSERT INTO PAL_SCORING_DATA_TBL VALUES('1', 0,'ATTR1','1');
INSERT INTO PAL_SCORING_DATA_TBL VALUES('1', 0,'ATTR2','10');
INSERT INTO PAL_SCORING_DATA_TBL VALUES('1', 0,'ATTR3','100');
INSERT INTO PAL_SCORING_DATA_TBL VALUES('1', 0,'ATTR4','A');
INSERT INTO PAL_SCORING_DATA_TBL VALUES('1', 0,'LABEL','1');
INSERT INTO PAL_SCORING_DATA_TBL VALUES('1', 1,'ATTR1',1.1);
INSERT INTO PAL_SCORING_DATA_TBL VALUES('1', 1,'ATTR2',10.1);
INSERT INTO PAL_SCORING_DATA_TBL VALUES('1', 1,'ATTR3',100.1);
INSERT INTO PAL_SCORING_DATA_TBL VALUES('1', 1,'ATTR4','A');
INSERT INTO PAL_SCORING_DATA_TBL VALUES('1', 1,'LABEL','1');
```

```
INSERT INTO PAL_SCORING_DATA_TBL VALUES('1', 2,'ATTR1',1.2);
INSERT INTO PAL_SCORING_DATA_TBL VALUES('1', 2,'ATTR2',10.2);
INSERT INTO PAL_SCORING_DATA_TBL VALUES('1', 2,'ATTR3',100);
INSERT INTO PAL_SCORING_DATA_TBL VALUES('1', 2,'ATTR4','A');
INSERT INTO PAL_SCORING_DATA_TBL VALUES('1', 2,'LABEL','1');
INSERT INTO PAL_SCORING_DATA_TBL VALUES('1', 3,'ATTR1',1.3);
INSERT INTO PAL_SCORING_DATA_TBL VALUES('1', 3,'ATTR2',10.4);
INSERT INTO PAL_SCORING_DATA_TBL VALUES('1', 3,'ATTR3',100);
INSERT INTO PAL_SCORING_DATA_TBL VALUES('1', 3,'ATTR4','A');
INSERT INTO PAL_SCORING_DATA_TBL VALUES('1', 3,'LABEL','1');
INSERT INTO PAL_SCORING_DATA_TBL VALUES('1', 4,'ATTR1',4);
INSERT INTO PAL_SCORING_DATA_TBL VALUES('1', 4,'ATTR2',40);
INSERT INTO PAL_SCORING_DATA_TBL VALUES('1', 4,'ATTR3',400);
INSERT INTO PAL_SCORING_DATA_TBL VALUES('1', 4,'ATTR4','AB');
INSERT INTO PAL_SCORING_DATA_TBL VALUES('1', 4,'LABEL','2');
INSERT INTO PAL_SCORING_DATA_TBL VALUES('1', 5,'ATTR1',9.2);
INSERT INTO PAL_SCORING_DATA_TBL VALUES('1', 5,'ATTR2',90.2);
INSERT INTO PAL_SCORING_DATA_TBL VALUES('1', 5,'ATTR3',900);
INSERT INTO PAL_SCORING_DATA_TBL VALUES('1', 5,'ATTR4','B');
INSERT INTO PAL_SCORING_DATA_TBL VALUES('1', 5,'LABEL','3');
INSERT INTO PAL_SCORING_DATA_TBL VALUES('2', 0,'ATTR1','5');
INSERT INTO PAL_SCORING_DATA_TBL VALUES('2', 0,'ATTR2','20');
INSERT INTO PAL_SCORING_DATA_TBL VALUES('2', 0,'ATTR3','300');
INSERT INTO PAL_SCORING_DATA_TBL VALUES('2', 0,'ATTR4','A');
INSERT INTO PAL_SCORING_DATA_TBL VALUES('2', 0,'LABEL','1');
INSERT INTO PAL_SCORING_DATA_TBL VALUES('2', 1,'ATTR1',3.1);
INSERT INTO PAL_SCORING_DATA_TBL VALUES('2', 1,'ATTR2',100.1);
INSERT INTO PAL_SCORING_DATA_TBL VALUES('2', 1,'ATTR3',10.1);
INSERT INTO PAL_SCORING_DATA_TBL VALUES('2', 1,'ATTR4','A');
INSERT INTO PAL_SCORING_DATA_TBL VALUES('2', 1,'LABEL','1');
INSERT INTO PAL_SCORING_DATA_TBL VALUES('2', 2,'ATTR1',9.2);
INSERT INTO PAL_SCORING_DATA_TBL VALUES('2', 2,'ATTR2',12.2);
INSERT INTO PAL_SCORING_DATA_TBL VALUES('2', 2,'ATTR3',90);
INSERT INTO PAL_SCORING_DATA_TBL VALUES('2', 2,'ATTR4','A');
INSERT INTO PAL_SCORING_DATA_TBL VALUES('2', 2,'LABEL','1');
INSERT INTO PAL_SCORING_DATA_TBL VALUES('2', 3,'ATTR1',3.1);
INSERT INTO PAL_SCORING_DATA_TBL VALUES('2', 3,'ATTR2',3.4);
INSERT INTO PAL_SCORING_DATA_TBL VALUES('2', 3,'ATTR3',9);
INSERT INTO PAL_SCORING_DATA_TBL VALUES('2', 3,'ATTR4','A');
INSERT INTO PAL_SCORING_DATA_TBL VALUES('2', 3,'LABEL','1');
INSERT INTO PAL_SCORING_DATA_TBL VALUES('2', 4,'ATTR1',4);
INSERT INTO PAL_SCORING_DATA_TBL VALUES('2', 4,'ATTR2',40);
INSERT INTO PAL_SCORING_DATA_TBL VALUES('2', 4,'ATTR3',400);
INSERT INTO PAL_SCORING_DATA_TBL VALUES('2', 4,'ATTR4','AB');
INSERT INTO PAL_SCORING_DATA_TBL VALUES('2', 4,'LABEL','2');
INSERT INTO PAL_SCORING_DATA_TBL VALUES('2', 5,'ATTR1',9.2);
INSERT INTO PAL_SCORING_DATA_TBL VALUES('2', 5,'ATTR2',90.2);
INSERT INTO PAL_SCORING_DATA_TBL VALUES('2', 5,'ATTR3',900);
INSERT INTO PAL_SCORING_DATA_TBL VALUES('2', 5,'ATTR4','B');
INSERT INTO PAL_SCORING_DATA_TBL VALUES('2', 5,'LABEL','3');
CALL
_SYS_AFL.PAL_UNIFIED_MASSIVE_CLASSIFICATION_SCORE_PIVOT(PAL_SCORING_DATA_TBL,
PAL_MODEL_TBL, #PAL_PARAMETER_TBL, ?, ?,?,?,?);
```

**Expected Result**:

RESULT:

Rows (12)

| | GROUP_ID | DATA_ID | SCORE | CONFIDENCE | REASON_CODE |
|---|---|---|---|---|---|
| 1 | 2 | 0 | 2 | -10.212584367777902 | [{"attr":"ATTR2","val":0.4047948277681892,"pct":51.8},{"attr":"ATTR4","val":-0.1988497016911319 6,"pct":25.5},{"attr":"ATTR1","val":0.1468417574265037 5,"pc |
| 2 | 2 | 1 | 1 | -11.363384408643883 | [{"attr":"ATTR2","val":0.5223332183941376,"pct":58.9},{"attr":"ATTR1","val":0.2245334740088870 8,"pct":25.3},{"attr":"ATTR4","val":0.1299804724151677 8,"pc |
| 3 | 2 | 2 | 1 | -10.842144660940724 | [{"attr":"ATTR2","val":0.2049103470997974,"pct":42.7},{"attr":"ATTR4","val":0.1817516654832811 5,"pct":37.9},{"attr":"ATTR3","val":0.0514052341643779 5,"pc |
| 4 | 2 | 3 | 1 | -11.342991237179948 | [{"attr":"ATTR4","val":0.255543507790645,"pct":63.1},{"attr":"ATTR1","val":0.09255171626808864,"pct":22.8},{"attr":"ATTR2","val":0.04232398479324221,"pct" |
| 5 | 2 | 4 | 2 | -9.428038024603229 | [{"attr":"ATTR2","val":0.4094958398071168,"pct":58.0},{"attr":"ATTR1","val":0.12843257808905135,"pct":18.200000000000004},{"attr":"ATTR3","val":0.126209 |
| 6 | 2 | 5 | 3 | -0.22258437860830815 | [{"attr":"ATTR1","val":0.19999665143812787,"pct":40.0},{"attr":"ATTR2","val":0.19999624786912024,"pct":40.0},{"attr":"ATTR4","val":-0.05000018165997264,"p |
| 7 | 1 | 0 | 1 | -11.652426313037237 | [{"attr":"ATTR1","val":0.23333333333333337,"pct":46.7},{"attr":"ATTR2","val":0.23333333333333335,"pct":46.7},{"attr":"ATTR4","val":-0.01666666666666665,"p |
| 8 | 1 | 1 | 1 | -4.451028880593471 | [{"attr":"ATTR1","val":0.23333333333333337,"pct":46.7},{"attr":"ATTR2","val":0.23333333333333335,"pct":46.7},{"attr":"ATTR4","val":-0.01666666666666665,"p |
| 9 | 1 | 2 | 1 | 0.3473811467832276 | [{"attr":"ATTR1","val":0.23333333333333337,"pct":46.7},{"attr":"ATTR2","val":0.23333333333333335,"pct":46.7},{"attr":"ATTR4","val":-0.01666666666666665,"p |
| 10 | 1 | 3 | 1 | -0.15271931956043872 | [{"attr":"ATTR1","val":0.23333333333333337,"pct":46.7},{"attr":"ATTR2","val":0.23333333333333335,"pct":46.7},{"attr":"ATTR4","val":-0.01666666666666665,"p |
| 11 | 1 | 4 | 2 | -0.05778412604532734 4 | [{"attr":"ATTR2","val":0.6666666666666666,"pct":66.69999999999999},{"attr":"ATTR1","val":0.3333333333333339,"pct":33.3},{"attr":"ATTR4","val":4.0029660 |
| 12 | 1 | 5 | 3 | 0.1340905653304243 | [{"attr":"ATTR2","val":0.23333333333333335,"pct":46.7},{"attr":"ATTR1","val":0.2333333333333332,"pct":46.7},{"attr":"ATTR4","val":-0.01666666666666665,"p |

STATISTICS:

Rows (32)

| | GROUP_ID | STAT_NAME | STAT_VALUE | CLASS_NAME |
|---|---|---|---|---|
| 1 | 2 | AUC | 0.9583333333333334 | *NULL* |
| 2 | 2 | RECALL | 0.75 | 1 |
| 3 | 2 | PRECISION | 1 | 1 |
| 4 | 2 | F1_SCORE | 0.8571428571428571 | 1 |
| 5 | 2 | SUPPORT | 4 | 1 |
| 6 | 2 | RECALL | 1 | 2 |
| 7 | 2 | PRECISION | 0.5 | 2 |
| 8 | 2 | F1_SCORE | 0.6666666666666666 | 2 |
| 9 | 2 | SUPPORT | 1 | 2 |
| 10 | 2 | RECALL | 1 | 3 |
| 11 | 2 | PRECISION | 1 | 3 |
| 12 | 2 | F1_SCORE | 1 | 3 |
| 13 | 2 | SUPPORT | 1 | 3 |
| 14 | 2 | ACCURACY | 0.8333333333333334 | *NULL* |
| 15 | 2 | KAPPA | 0.7142857142857143 | *NULL* |
| 16 | 2 | MCC | 0.753778361444409 | *NULL* |
| 17 | 1 | AUC | 1 | *NULL* |
| 18 | 1 | RECALL | 1 | 1 |
| 19 | 1 | PRECISION | 1 | 1 |
| 20 | 1 | F1_SCORE | 1 | 1 |
| 21 | 1 | SUPPORT | 4 | 1 |
| 22 | 1 | RECALL | 1 | 2 |
| 23 | 1 | PRECISION | 1 | 2 |
| 24 | 1 | F1_SCORE | 1 | 2 |
| 25 | 1 | SUPPORT | 1 | 2 |
| 26 | 1 | RECALL | 1 | 3 |
| 27 | 1 | PRECISION | 1 | 3 |
| 28 | 1 | F1_SCORE | 1 | 3 |

CONFUSION_MATRIX:

Rows (18)

| | GROUP_ID | ACTUAL_CLASS | PREDICTED_CLASS | COUNT |
|---|---|---|---|---|
| 1 | 2 | 1 | 1 | 3 |
| 2 | 2 | 1 | 2 | 1 |
| 3 | 2 | 1 | 3 | 0 |
| 4 | 2 | 2 | 1 | 0 |
| 5 | 2 | 2 | 2 | 1 |
| 6 | 2 | 2 | 3 | 0 |
| 7 | 2 | 3 | 1 | 0 |
| 8 | 2 | 3 | 2 | 0 |
| 9 | 2 | 3 | 3 | 1 |
| 10 | 1 | 1 | 1 | 4 |
| 11 | 1 | 1 | 2 | 0 |
| 12 | 1 | 1 | 3 | 0 |
| 13 | 1 | 2 | 1 | 0 |
| 14 | 1 | 2 | 2 | 1 |
| 15 | 1 | 2 | 3 | 0 |
| 16 | 1 | 3 | 1 | 0 |
| 17 | 1 | 3 | 2 | 0 |
| 18 | 1 | 3 | 3 | 1 |

METRICS:

Rows (72)

| | GROUP_ID | NAME | X | Y |
|---|---|---|---|---|
| 34 | 2 | ROC_TPR | 10 | 1 |
| 35 | 2 | ROC_FPR | 11 | 0.4166666666666667 |
| 36 | 2 | ROC_TPR | 11 | 1 |
| 37 | 2 | ROC_FPR | 12 | 0.5 |
| 38 | 2 | ROC_TPR | 12 | 1 |
| 39 | 2 | ROC_FPR | 13 | 0.5833333333333334 |
| 40 | 2 | ROC_TPR | 13 | 1 |
| 41 | 2 | ROC_FPR | 14 | 1 |
| 42 | 2 | ROC_TPR | 14 | 1 |
| 43 | 2 | CUMGAINS | 0 | 0 |
| 44 | 2 | CUMGAINS | 1 | 1 |
| 45 | 2 | LIFT | 0 | 3 |
| 46 | 2 | LIFT | 1 | 0 |
| 47 | 2 | CUMLIFT | 0 | 3 |
| 48 | 2 | CUMLIFT | 1 | 1 |
| 49 | 1 | RANDOM_CUMGAINS | 0 | 0 |
| 50 | 1 | RANDOM_CUMGAINS | 1 | 1 |
| 51 | 1 | RANDOM_LIFT | 0 | 1 |
| 52 | 1 | RANDOM_LIFT | 1 | 1 |
| 53 | 1 | RANDOM_CUMLIFT | 0 | 1 |
| 54 | 1 | RANDOM_CUMLIFT | 1 | 1 |
| 55 | 1 | PERF_CUMGAINS | 0 | 0 |
| 56 | 1 | PERF_CUMGAINS | 1 | 1 |
| 57 | 1 | PERF_LIFT | 0 | 3 |
| 58 | 1 | PERF_LIFT | 1 | 0 |
| 59 | 1 | PERF_CUMLIFT | 0 | 3 |
| 60 | 1 | PERF_CUMLIFT | 1 | 1 |
| 61 | 1 | ROC_FPR | 0 | 0 |
| 62 | 1 | ROC_TPR | 0 | 0 |

## 5.13.5 Unified Clustering

Unified clustering is a unified interface to call some clustering algorithms with unpivot data format.

The clustering algorithms include:

- AHC (Agglomerate Hierarchical Clustering)
- DBSCAN (Density-Based Spatial Clustering of Applications with Noise)
- GMM (Gaussian Mixture Model)
- AKMEANS (Accelerated K-Means)
- KMEANS(K-Means)
- KMEDIANS(K-Medians)
- KMEDOIDS(K-Medoids)
- SOM (Self-Organizing Maps)
- AP (Affinity Propagation)
- SP (Spectral Clustering)

For GMM, you must configure the INITIALIZE_PARAMETER parameters to define the INITIALIZE_PARAMETER table of the original method.

For AP, you must configure the INITIALIZE_SEED parameters to define the SEED table of the original method.

Compared to the original K-Medians and K-Medoids, unified clustering creates models after a training and then performs cluster assignment through the model. See Compatibility [page 1795] for details on compatibility.

See the corresponding topic for details about a specific clustering method.

## Prerequisites

- All prerequisites for the original corresponding cluster function apply in the unified cluster interface.

## _SYS_AFL.PAL_UNIFIED_CLUSTERING

### Overviews of ALL Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA Table>` | IN |
| 2 | `<PARAMETER Table>` | IN |
| 3 | `<RESULT Table>` | OUT |
| 4 | `<CENTERS Table>` | OUT |
| 5 | `<MODEL Table>` | OUT |
| 6 | `<STATISTICS Table>` | OUT |
| 7 | `<OPTIMAL_PARAMETER Table>` | OUT |
| 8 | `<PLACE_HOLDER_1 Table>` | OUT |
| 9 | `<PLACE_HOLDER_2 Table>` | OUT |

### Input Tables

| Table | Column | Reference for Output Table | Data Type | Description | Constraint |
|---|---|---|---|---|---|
| DATA | 1st column | DATA_ID | BIGINT, INTEGER, VARCHAR, or NVARCHAR | Data ID | This must be the first column. |
| | Other columns | FEATURES | INTEGER, DOUBLE, DECIMAL(p,s), VARCHAR, or NVARCHAR | Attribute data | |

### Parameter Table

Mandatory Parameters

The following parameters are mandatory and must be given a value.

| Name | Data Type | Description | Dependency |
|---|---|---|---|
| FUNCTION | VARHCAR or NVARCHAR | The algorithm name, including:<br>• KMEANS<br>• AKMEANS<br>• KMEDIANS<br>• KMEDOIDS<br>• AHC<br>• DBSCAN<br>• SOM<br>• GMM<br>• AP<br>• SP | |

See Parameter Tables [page 1799] for details about parameters for a specific algorithm.

### Output Table

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| RESULT | 1st column | DATA_ID (in input table) data type | DATA_ID (in input table) column name | Data ID |
| | 2nd column | INTEGER | CLUSTER_ID | The assigned cluster ID. |
| | 3rd column | DOUBLE | DISTANCE | Distance between a given point and the<br>• cluster center (k-means)<br>• nearest core object (DBSCAN)<br>• weight vector (SOM)<br>Or probability of a given point belonging to the corresponding cluster (GMM). |
| | 4th column | DOUBLE | SLIGHT_SILHOUETTE | Estimated value (slight silhouette) |
| CENTERS | 1st column | INTEGER | CLUSTER_ID | |
| | 2nd column | VARCHAR, NVARCHAR | VARIABLE_NAME | The name of variable. |
| | 3rd column | VARCHAR or NVARCHAR | VALUE | The value of variable. |
| MODEL | 1st column | INTEGER | ROW_INDEX | Row index. |

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| | 2nd column | INTEGER | PART_INDEX | Specifically for GMM's CLUSTER_ID. |
| | 3rd column | NCLOB | MODEL_CONTENT | Model content. |
| STATISTICS | 1st column | VARHCAR or NVARCHAR | STATISTIC_NAME | |
| | 2nd column | VARHCAR or NVARCHAR | STATISTIC_VALUE | |
| OPTIMAL_PARAMETER | 1st column | VARHCAR or NVARCHAR | NAME | Placeholder |
| | 2nd column | INTEGER | INT_VALUE | |
| | 3rd column | DOUBLE | DOUBLE_VALUE | |
| | 4th column | VARHCAR or NVARCHAR | STRING_VALUE | |
| PLACE_HOLDER_1 | 1st column | DATA_ID (in input table) data type | DATA_ID (in input table) column name | |
| | 2nd column | INTEGER | TYPE | |
| PLACE_HOLDER_2 | 1st column | VARHCAR or NVARCHAR | OBJECT | Placeholder for future features. |
| | 2nd column | VARHCAR or NVARCHAR | KEY | |
| | 3rd column | VARHCAR or NVARCHAR | VALUE | |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

Example: K-Means

```
SET SCHEMA DM_PAL;
DROP TABLE MY_DATA_TABLE;
CREATE COLUMN TABLE MY_DATA_TABLE (
    "DATA_ID" INTEGER,
    "V000" DOUBLE,
    "V001" VARCHAR(2),
    "V002" DOUBLE
);
INSERT INTO MY_DATA_TABLE VALUES (0, 0.5, 'A', 0.5);
INSERT INTO MY_DATA_TABLE VALUES (1, 1.5, 'A', 0.5);
INSERT INTO MY_DATA_TABLE VALUES (2, 1.5, 'A', 1.5);
INSERT INTO MY_DATA_TABLE VALUES (3, 0.5, 'A', 1.5);
INSERT INTO MY_DATA_TABLE VALUES (4, 1.1, 'B', 1.2);
INSERT INTO MY_DATA_TABLE VALUES (5, 0.5, 'B', 15.5);
INSERT INTO MY_DATA_TABLE VALUES (6, 1.5, 'B', 15.5);
INSERT INTO MY_DATA_TABLE VALUES (7, 1.5, 'B', 16.5);
INSERT INTO MY_DATA_TABLE VALUES (8, 0.5, 'B', 16.5);
INSERT INTO MY_DATA_TABLE VALUES (9, 1.2, 'C', 16.1);
```

```
INSERT INTO MY_DATA_TABLE VALUES (10, 15.5, 'C', 15.5);
INSERT INTO MY_DATA_TABLE VALUES (11, 16.5, 'C', 15.5);
INSERT INTO MY_DATA_TABLE VALUES (12, 16.5, 'C', 16.5);
INSERT INTO MY_DATA_TABLE VALUES (13, 15.5, 'C', 16.5);
INSERT INTO MY_DATA_TABLE VALUES (14, 15.6, 'D', 16.2);
INSERT INTO MY_DATA_TABLE VALUES (15, 15.5, 'D', 0.5);
INSERT INTO MY_DATA_TABLE VALUES (16, 16.5, 'D', 0.5);
INSERT INTO MY_DATA_TABLE VALUES (17, 16.5, 'D', 1.5);
INSERT INTO MY_DATA_TABLE VALUES (18, 15.5, 'D', 1.5);
INSERT INTO MY_DATA_TABLE VALUES (19, 15.7, 'A', 1.6);
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL (
"NAME" VARCHAR (50),
"INT_VALUE" INTEGER,
"DOUBLE_VALUE" DOUBLE,
"STRING_VALUE" VARCHAR (100)
);
INSERT INTO PAL_PARAMETER_TBL VALUES ('FUNCTION', NULL, NULL, 'KMEANS');
INSERT INTO PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 1.0, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('N_CLUSTERS', 4, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('INIT', 1, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('DISTANCE_LEVEL',2, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('MAX_ITER', 100, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('TOL', NULL, 1.0E-6, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('CATEGORY_WEIGHTS', NULL, 0.5, NULL);
CALL
_SYS_AFL.PAL_UNIFIED_CLUSTERING(MY_DATA_TABLE,PAL_PARAMETER_TBL, ?, ?,?,?,?,?,?);
```

**Expected Result**

RESULT:

| | ID | CLUSTER_ID | DISTANCE | SLIGHT_SILHOUETE |
|---|---|---|---|---|
| 1 | 0 | 0 | 0.8910879487969621 | 0.9443700340289641 |
| 2 | 1 | 0 | 0.8639170309648472 | 0.9424783519324079 |
| 3 | 2 | 0 | 0.8062521617810958 | 0.9462881150703033 |
| 4 | 3 | 0 | 0.8356835545457009 | 0.944941534308603 |
| 5 | 4 | 0 | 0.7445708631492213 | 0.9502343526704448 |
| 6 | 5 | 3 | 0.8910879487969616 | 0.9407325552117667 |
| 7 | 6 | 3 | 0.8356835545457004 | 0.9444124645366334 |
| 8 | 7 | 3 | 0.8062521617810963 | 0.9465187213724425 |
| 9 | 8 | 3 | 0.8639170309648476 | 0.9461211197877126 |
| 10 | 9 | 3 | 0.744570863149222 | 0.9498994935338706 |
| 11 | 10 | 2 | 0.8255266112967905 | 0.945092293345542 |
| 12 | 11 | 2 | 0.9338858664836682 | 0.9379022885331548 |
| 13 | 12 | 2 | 0.8816915771701814 | 0.9450081614922042 |
| 14 | 13 | 2 | 0.7643178163332063 | 0.9491595311531508 |
| 15 | 14 | 2 | 0.9234563013492032 | 0.939282992463176 |
| 16 | 15 | 1 | 0.9016844685872387 | 0.940436243125126 |
| 17 | 16 | 1 | 0.9768852992055255 | 0.9393860344624405 |
| 18 | 17 | 1 | 0.8181783288648536 | 0.9458778291291957 |
| 19 | 18 | 1 | 0.7227990303872558 | 0.9521697340450603 |
| 20 | 19 | 1 | 1.1023417395491886 | 0.9256786403653229 |

CENTERS:

| | CLUSTER_ID | VARIABLE_NAME | VALUE |
|---|---|---|---|
| 1 | 0 | V000 | 1.02 |
| 2 | 0 | V001 | A |
| 3 | 0 | V002 | 1.04 |
| 4 | 1 | V000 | 15.94... |
| 5 | 1 | V001 | D |
| 6 | 1 | V002 | 1.119... |
| 7 | 2 | V000 | 15.91... |
| 8 | 2 | V001 | C |
| 9 | 2 | V002 | 16.04 |
| 10 | 3 | V000 | 1.04 |
| 11 | 3 | V001 | B |
| 12 | 3 | V002 | 16.02 |

MODEL:

| | ROW_INDEX | PART_INDEX | MODEL_CONTENT |
|---|---|---|---|
| 1 | 0 | ? | {"Algorithm":"KMEANS","Category":[{"CategoryCol":2,"CategoryName":["A","B","C","D"],"... |

STATISTICS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | Center 0 Slight Silhouette | 0.9456624776021446 |
| 2 | Center 1 Slight Silhouette | 0.9407096962254291 |
| 3 | Center 2 Slight Silhouette | 0.9432890533974454 |
| 4 | Center 3 Slight Silhouette | 0.9455368708884853 |
| 5 | SlightSilhouette | 0.943799524528376 |

## _SYS_AFL.PAL_UNIFIED_CLUSTERING_ASSIGNMENT

Clustering assignment is a unified interface to call a cluster assignment algorithm to assign data to clusters that are previously generated by some clustering methods, including K-Means, Accelerated K-Means, K-Medians, K- Medoids, DBSCAN, SOM, and GMM.

For details about compatibility with the training models of original algorithms, see Compatibility [page 1795].

**Overviews of ALL Tables**

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<MODEL table>` | IN |
| 3 | `<PARAMETER table>` | IN |
| 4 | `<ASSIGNMENT table>` | OUT |
| 5 | `<PLACE_HOLDER_1 table>` | OUT |

### Input Tables

| Table | Column | Reference for Output Table | Data Type | Description | Constraint |
|---|---|---|---|---|---|
| DATA | 1st column | DATA_ID | BIGINT, INTEGER, VARCHAR, NVARCHAR | Data ID. | This must be the first column. |
| | Other columns | | INTEGER, DOUBLE, DECIMAL(p,s), VARCHAR, or NVARCHAR | Attribute data. | Should be identical to the model.<br>For SOM, attribute data supports only numeric. Therefore its column data type is INTEGER, DOUBLE, or DECIMAL(p, s). |
| MODEL | 1st column | ROW_INDEX | INTEGER | Row index. | This must be the first column. |
| | 2nd column | PART_INDEX | INTEGER | CLUSTER_ID, for GMM only. | |
| | 3rd column | MODEL_CONTENT | NCLOB | Model content. | |

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| THREAD_RATIO | DOUBLE | 0.0 | The ratio of available threads for multi-thread tasks:<br>• 0: single thread;<br>• 0~1: percentage.<br>Other values are ignored and this function heuristically determines the number of threads to use. | |

### Output Table

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| ASSIGNMENT | 1st column | DATA_ID (in input table) data type | DATA_ID (in input table) column name | Data ID. |
| | 2nd column | INTEGER | CLUSTER_ID | The assigned cluster ID. |
| | 3rd column | DOUBLE | DISTANCE | Distance between a given point and the<br>• cluster center (k-means)<br>• nearest core object (DBSCAN)<br>• weight vector (SOM)<br>Or probability of a given point belonging to the corresponding cluster (GMM). |
| PLACE_HOLDER_1 | 1st column | VARCHAR or NVARCHAR | OBJECT | Placeholder for future features. |
| | 2nd column | VARCHAR or NVARCHAR | KEY | |
| | 3rd column | VARCHAR or NVARCHAR | VALUE | |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

Example: DBSCAN

```
SET SCHEMA DM_PAL;
DROP TABLE MY_DATA_TABLE;
CREATE COLUMN TABLE MY_DATA_TABLE (
"DATA_ID" INTEGER,
"V1" DOUBLE,
"V2" DOUBLE,
"V3" VARCHAR(10)
);
INSERT INTO MY_DATA_TABLE VALUES(1, 0.10, 0.10,  'B');
INSERT INTO MY_DATA_TABLE VALUES(2, 0.11, 0.10,  'A');
INSERT INTO MY_DATA_TABLE VALUES(3, 0.10, 0.11,  'C');
INSERT INTO MY_DATA_TABLE VALUES(4, 0.11, 0.11,  'B');
INSERT INTO MY_DATA_TABLE VALUES(5, 0.12, 0.11,  'A');
INSERT INTO MY_DATA_TABLE VALUES(6, 0.11, 0.12,  'E');
INSERT INTO MY_DATA_TABLE VALUES(7, 0.12, 0.12,  'A');
INSERT INTO MY_DATA_TABLE VALUES(8, 0.12, 0.13,  'C');
INSERT INTO MY_DATA_TABLE VALUES(9, 0.13, 0.12,  'D');
INSERT INTO MY_DATA_TABLE VALUES(10, 0.13, 0.13,  'D');
INSERT INTO MY_DATA_TABLE VALUES(11, 0.13, 0.14,  'A');
INSERT INTO MY_DATA_TABLE VALUES(12, 0.14, 0.13,  'C');
INSERT INTO MY_DATA_TABLE VALUES(13, 10.10, 10.10,  'A');
INSERT INTO MY_DATA_TABLE VALUES(14, 10.11, 10.10,  'F');
INSERT INTO MY_DATA_TABLE VALUES(15, 10.10, 10.11,  'E');
INSERT INTO MY_DATA_TABLE VALUES(16, 10.11, 10.11,  'E');
INSERT INTO MY_DATA_TABLE VALUES(17, 10.11, 10.12,  'A');
INSERT INTO MY_DATA_TABLE VALUES(18, 10.12, 10.11,  'B');
INSERT INTO MY_DATA_TABLE VALUES(19, 10.12, 10.12,  'B');
INSERT INTO MY_DATA_TABLE VALUES(20, 10.12, 10.13,  'D');
INSERT INTO MY_DATA_TABLE VALUES(21, 10.13, 10.12,  'F');
INSERT INTO MY_DATA_TABLE VALUES(22, 10.13, 10.13,  'A');
INSERT INTO MY_DATA_TABLE VALUES(23, 10.13, 10.14,  'A');
INSERT INTO MY_DATA_TABLE VALUES(24, 10.14, 10.13,  'D');
INSERT INTO MY_DATA_TABLE VALUES(25, 4.10, 4.10,  'A');
INSERT INTO MY_DATA_TABLE VALUES(26, 7.11, 7.10,  'C');
INSERT INTO MY_DATA_TABLE VALUES(27, -3.10, -3.11,  'C');
INSERT INTO MY_DATA_TABLE VALUES(28, 16.11, 16.11,  'A');
INSERT INTO MY_DATA_TABLE VALUES(29, 20.11, 20.12,  'C');
INSERT INTO MY_DATA_TABLE VALUES(30, 15.12, 15.11,  'A');
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL (
"NAME" VARCHAR (50),
"INT_VALUE" INTEGER,
"DOUBLE_VALUE" DOUBLE,
"STRING_VALUE" VARCHAR (100)
);
INSERT INTO PAL_PARAMETER_TBL VALUES ('FUNCTION', NULL, NULL, 'DBSCAN');
INSERT INTO PAL_PARAMETER_TBL VALUES ('AUTO_PARAM', NULL, NULL, 'true');
INSERT INTO PAL_PARAMETER_TBL VALUES ('METRIC', 1, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 1.0, NULL);
DROP TABLE PAL_CLUSTERING_MODEL_TBL;
CREATE COLUMN TABLE PAL_CLUSTERING_MODEL_TBL (
"ROW_INDEX" INTEGER,
"PART_INDEX" INTEGER,
"MODEL_CONTENT" VARCHAR (5000)
);
DO BEGIN
lt_data = SELECT * FROM MY_DATA_TABLE;
lt_param = SELECT * FROM PAL_PARAMETER_TBL;
CALL "_SYS_AFL"."PAL_UNIFIED_CLUSTERING"(:lt_data, :lt_param, lt_result,
lt_centers, lt_model, lt_stat, lt_optimal_parameter, lt_place1, lt_place2);
INSERT INTO PAL_CLUSTERING_MODEL_TBL SELECT * FROM :lt_model;
END;
DROP TABLE MY_DATA_ASSIGNMENT_TABLE;
CREATE COLUMN TABLE MY_DATA_ASSIGNMENT_TABLE (
    "DATA_ID" INTEGER,
    "V1" DOUBLE,
```

```
    "V2" DOUBLE,
    "V3" VARCHAR(10)
);
INSERT INTO MY_DATA_ASSIGNMENT_TABLE VALUES (1, 0.10, 0.10, 'B');
INSERT INTO MY_DATA_ASSIGNMENT_TABLE VALUES (2, 0.10, -2.10, 'A');
INSERT INTO MY_DATA_ASSIGNMENT_TABLE VALUES (3, 3.10, 0.10, 'B');
INSERT INTO MY_DATA_ASSIGNMENT_TABLE VALUES (4, 10.10, 10.10, 'D');
INSERT INTO MY_DATA_ASSIGNMENT_TABLE VALUES (5, 3.10, -0.50, 'C');
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL (
"NAME" VARCHAR (50),
"INT_VALUE" INTEGER,
"DOUBLE_VALUE" DOUBLE,
"STRING_VALUE" VARCHAR (100)
);
CALL
_SYS_AFL.PAL_UNIFIED_CLUSTERING_ASSIGNMENT(MY_DATA_ASSIGNMENT_TABLE,PAL_CLUSTERIN
G_MODEL_TBL, PAL_PARAMETER_TBL, ?,?);
```

**Expected Result**

ASSIGNMENT:

| | DATA_ID | CLUSTER_ID | DISTANCE |
|---|---|---|---|
| 1 | 1 | 0 | 0 |
| 2 | 2 | 0 | 2.21 |
| 3 | 3 | -1 | 3 |
| 4 | 4 | 1 | 0.050000... |
| 5 | 5 | -1 | 3.59 |

## 5.13.5.1 Parameters Overview

The following table shows mapping parameters between unified cluster procedures and original procedures.

| Algorithm | New Name | Original Name |
|---|---|---|
| AHC | LINKAGE | CLUSTER_METHOD |
| | N_CLUSTERS | CLUSTER_NUM |
| | DISTANCE_LEVEL | DISTANCE_FUNC |
| | NORMALIZATION | NORMALIZE_TYPE |
| DBSCAN | DISTANCE_LEVEL | DISTANCE_METHOD |
| | ALGORITHM | METHOD |
| | MINKOWSKI_POWER | MINKOW_P |
| | EPS | RADIUS |
| GMM | MAX_ITER | MAX_ITERATION |
| KMEDIANS | TOL | EXIT_THRESHOLD |
| | N_CLUSTERS | GROUP_NUMBER |

| Algorithm | New Name | Original Name |
|---|---|---|
| | INIT | INIT_TYPE |
| | MAX_ITER | MAX_ITERATION |
| KMEDOIDS | TOL | EXIT_THRESHOLD |
| | N_CLUSTERS | GROUP_NUMBER |
| | INIT | INIT_TYPE |
| | MAX_ITER | MAX_ITERATION |
| KMEANS | TOL | EXIT_THRESHOLD |
| | N_CLUSTERS | GROUP_NUMBER |
| | N_CLUSTERS_MAX | GROUP_NUMBER_MAX |
| | N_CLUSTERS_MIN | GROUP_NUMBER_MIN |
| | INIT | INIT_TYPE |
| | MAX_ITER | MAX_ITERATION |
| AKMEANS | TOL | EXIT_THRESHOLD |
| | N_CLUSTERS | GROUP_NUMBER |
| | N_CLUSTERS_MAX | GROUP_NUMBER_MAX |
| | N_CLUSTERS_MIN | GROUP_NUMBER_MIN |
| | INIT | INIT_TYPE |
| | MAX_ITER | MAX_ITERATION |
| SOM | MAX_ITER | MAX_ITERATION |
| AP | DISTANCE_LEVEL | DISTANCE_METHOD |
| | MAX_ITER | MAX_ITERATION |
| SP | N_CLUSTERS | GROUP_NUMBER |
| | MAX_ITER | MAX_ITERATION |
| | INIT | INIT_TYPE |
| | SPLIT_THRESHOLD | GAMMA |
| | TOL | EXIT_THRESHOLD |

## 5.13.5.2 Compatibility

Models generated from an original PAL procedure except K-Medians and K-Medoids can be used by a unified cluster interface for cluster assignment. These models are accepted by the unified cluster interface without any additional processing.

Likewise, models generated from a unified cluster interface can also be used by an original PAL procedure for cluster assignment, except Accelerated K-Means, K-Medians, and K-Medoids. However, you need to remove or add the second column before some original PAL procedures can use the models, as shown in the following table.

| Model Source | Model Destination | Model Compatibility | Removal (Yes/No) |
|---|---|---|---|
| Unified Cluster | DBSCAN | Yes | Yes |
| | Accelerated K-Means | No | N/A |
| | K-Means | Yes | Yes |
| | K-Medians | No | N/A |
| | K-Medoids | No | N/A |
| | SOM | Yes | Yes |
| | GMM | Yes | No |
| | AP | No | N/A |

See Cluster Assignment [page 106] for details about supported procedures for original cluster assignment.

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

Example 1: Using a unified cluster model in an original cluster assignment procedure

```
SET SCHEMA DM_PAL;
-------------------------Unified Cluster Training
Procedure------------------------
--Data Table
DROP TABLE MY_DATA_TABLE;
CREATE COLUMN TABLE MY_DATA_TABLE (
"DATA_ID" INTEGER,
"V1" DOUBLE,
"V2" DOUBLE,
"V3" VARCHAR(10)
);
INSERT INTO MY_DATA_TABLE VALUES(1, 0.10, 0.10,  'B');
INSERT INTO MY_DATA_TABLE VALUES(2, 0.11, 0.10,  'A');
INSERT INTO MY_DATA_TABLE VALUES(3, 0.10, 0.11,  'C');
INSERT INTO MY_DATA_TABLE VALUES(4, 0.11, 0.11,  'B');
INSERT INTO MY_DATA_TABLE VALUES(5, 0.12, 0.11,  'A');
INSERT INTO MY_DATA_TABLE VALUES(6, 0.11, 0.12,  'E');
INSERT INTO MY_DATA_TABLE VALUES(7, 0.12, 0.12,  'A');
INSERT INTO MY_DATA_TABLE VALUES(8, 0.12, 0.13,  'C');
INSERT INTO MY_DATA_TABLE VALUES(9, 0.13, 0.12,  'D');
INSERT INTO MY_DATA_TABLE VALUES(10, 0.13, 0.13,  'D');
INSERT INTO MY_DATA_TABLE VALUES(11, 0.13, 0.14,  'A');
INSERT INTO MY_DATA_TABLE VALUES(12, 0.14, 0.13,  'C');
INSERT INTO MY_DATA_TABLE VALUES(13, 10.10, 10.10,  'A');
INSERT INTO MY_DATA_TABLE VALUES(14, 10.11, 10.10,  'F');
INSERT INTO MY_DATA_TABLE VALUES(15, 10.10, 10.11,  'E');
INSERT INTO MY_DATA_TABLE VALUES(16, 10.11, 10.11,  'E');
INSERT INTO MY_DATA_TABLE VALUES(17, 10.11, 10.12,  'A');
INSERT INTO MY_DATA_TABLE VALUES(18, 10.12, 10.11,  'B');
INSERT INTO MY_DATA_TABLE VALUES(19, 10.12, 10.12,  'B');
INSERT INTO MY_DATA_TABLE VALUES(20, 10.12, 10.13,  'D');
INSERT INTO MY_DATA_TABLE VALUES(21, 10.13, 10.12,  'F');
INSERT INTO MY_DATA_TABLE VALUES(22, 10.13, 10.13,  'A');
INSERT INTO MY_DATA_TABLE VALUES(23, 10.13, 10.14,  'A');
INSERT INTO MY_DATA_TABLE VALUES(24, 10.14, 10.13,  'D');
INSERT INTO MY_DATA_TABLE VALUES(25, 4.10, 4.10,  'A');
```

```
INSERT INTO MY_DATA_TABLE VALUES(26, 7.11, 7.10,  'C');
INSERT INTO MY_DATA_TABLE VALUES(27, -3.10, -3.11,  'C');
INSERT INTO MY_DATA_TABLE VALUES(28, 16.11, 16.11,  'A');
INSERT INTO MY_DATA_TABLE VALUES(29, 20.11, 20.12,  'C');
INSERT INTO MY_DATA_TABLE VALUES(30, 15.12, 15.11,  'A');
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL (
"NAME" VARCHAR (50),
"INT_VALUE" INTEGER,
"DOUBLE_VALUE" DOUBLE,
"STRING_VALUE" VARCHAR (100)
);
INSERT INTO PAL_PARAMETER_TBL VALUES ('FUNCTION', NULL, NULL, 'DBSCAN');
INSERT INTO PAL_PARAMETER_TBL VALUES ('AUTO_PARAM', NULL, NULL, 'true');
INSERT INTO PAL_PARAMETER_TBL VALUES ('METRIC', 1, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 1.0, NULL);
---model table
DROP TABLE PAL_CLUSTERING_MODEL_TBL;
CREATE COLUMN TABLE PAL_CLUSTERING_MODEL_TBL (
"ROW_INDEX" INTEGER,
"PART_INDEX" INTEGER,
"MODEL_CONTENT" VARCHAR (5000)
);
DO BEGIN
  lt_data = SELECT * FROM MY_DATA_TABLE;
  lt_param = SELECT * FROM PAL_PARAMETER_TBL;
  CALL "_SYS_AFL"."PAL_UNIFIED_CLUSTERING"(:lt_data, :lt_param, lt_result,
lt_centers, lt_model, lt_stat, lt_optimal_parameter, lt_place1, lt_place2);
  INSERT INTO PAL_CLUSTERING_MODEL_TBL SELECT * FROM :lt_model;
END;
-------------------------Origin Cluster Assignment
Procedure------------------------
DROP TABLE PAL_ORIGIN_MODEL_TBL;
CREATE COLUMN TABLE PAL_ORIGIN_MODEL_TBL (
"ROW_INDEX" INTEGER,
"MODEL_CONTENT" VARCHAR (5000)
);
------------------Remove Processing-----------------------
INSERT INTO PAL_ORIGIN_MODEL_TBL(ROW_INDEX,MODEL_CONTENT) SELECT ROW_INDEX,
MODEL_CONTENT from PAL_CLUSTERING_MODEL_TBL;
DROP TABLE PAL_DBSCAN_DATA_TBL;
CREATE COLUMN TABLE PAL_DBSCAN_DATA_TBL (
    "ID" INTEGER,
    "V1" DOUBLE,
    "V2" DOUBLE,
    "V3" VARCHAR(10)
);
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES (1, 0.10, 0.10, 'B');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES (2, 0.10, -2.10, 'A');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES (3, 3.10, 0.10, 'B');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES (4, 10.10, 10.10, 'D');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES (5, 3.10, -0.50, 'C');
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR(100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR(100)
);
CALL _SYS_AFL.PAL_CLUSTER_ASSIGNMENT(PAL_DBSCAN_DATA_TBL, PAL_ORIGIN_MODEL_TBL,
"#PAL_PARAMETER_TBL",  ?);
```

**Expected result**

RESULT:

| | ID | CLUSTER_ID | DISTANCE |
|---|---|---|---|
| 1 | 1 | 0 | 0 |
| 2 | 2 | 0 | 2.21 |
| 3 | 3 | -1 | 3 |
| 4 | 4 | 1 | 0.05000000000000071 |
| 5 | 5 | -1 | 3.59 |

Example 2: Using an origin training model in a unified cluster assignment procedure

```
SET SCHEMA DM_PAL;
-------------------------Origin DBSCAN Training Procedure------------------------
DROP TABLE PAL_DBSCAN_DATA_TBL;
CREATE COLUMN TABLE PAL_DBSCAN_DATA_TBL (
    "ID" INTEGER,
    "V1" DOUBLE,
    "V2" DOUBLE,
    "V3" VARCHAR(10)
);
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(1, 0.10, 0.10,  'B');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(2, 0.11, 0.10,  'A');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(3, 0.10, 0.11,  'C');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(4, 0.11, 0.11,  'B');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(5, 0.12, 0.11,  'A');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(6, 0.11, 0.12,  'E');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(7, 0.12, 0.12,  'A');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(8, 0.12, 0.13,  'C');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(9, 0.13, 0.12,  'D');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(10, 0.13, 0.13,  'D');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(11, 0.13, 0.14,  'A');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(12, 0.14, 0.13,  'C');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(13, 10.10, 10.10,  'A');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(14, 10.11, 10.10,  'F');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(15, 10.10, 10.11,  'E');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(16, 10.11, 10.11,  'E');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(17, 10.11, 10.12,  'A');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(18, 10.12, 10.11,  'B');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(19, 10.12, 10.12,  'B');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(20, 10.12, 10.13,  'D');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(21, 10.13, 10.12,  'F');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(22, 10.13, 10.13,  'A');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(23, 10.13, 10.14,  'A');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(24, 10.14, 10.13,  'D');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(25, 4.10, 4.10,  'A');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(26, 7.11, 7.10,  'C');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(27, -3.10, -3.11,  'C');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(28, 16.11, 16.11,  'A');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(29, 20.11, 20.12,  'C');
INSERT INTO PAL_DBSCAN_DATA_TBL VALUES(30, 15.12, 15.11,  'A');
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL (
"NAME" VARCHAR (50),
"INT_VALUE" INTEGER,
"DOUBLE_VALUE" DOUBLE,
"STRING_VALUE" VARCHAR (100)
);
INSERT INTO PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 0.2, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('AUTO_PARAM', NULL, NULL, 'true');
INSERT INTO PAL_PARAMETER_TBL VALUES ('DISTANCE_METHOD', 1, NULL, NULL);
DROP TABLE PAL_DBSCAN_MODEL_TBL;
CREATE COLUMN TABLE PAL_DBSCAN_MODEL_TBL (
    "ROW_INDEX" INTEGER,
```

```
    "MODEL_CONTENT" NVARCHAR(5000)
);
DO BEGIN
  lt_data = SELECT * FROM PAL_DBSCAN_DATA_TBL;
  lt_param = SELECT * FROM PAL_PARAMETER_TBL;
  CALL "_SYS_AFL"."PAL_DBSCAN"(:lt_data, :lt_param, lt_result, lt_model,
lt_stat, lt_place1);
  INSERT INTO PAL_DBSCAN_MODEL_TBL SELECT * FROM :lt_model;
END;
DROP TABLE MY_DATA_ASSIGNMENT_TABLE;
CREATE COLUMN TABLE MY_DATA_ASSIGNMENT_TABLE (
    "DATA_ID" INTEGER,
    "V1" DOUBLE,
    "V2" DOUBLE,
    "V3" VARCHAR(10)
);
INSERT INTO MY_DATA_ASSIGNMENT_TABLE VALUES (1, 0.10, 0.10, 'B');
INSERT INTO MY_DATA_ASSIGNMENT_TABLE VALUES (2, 0.10, -2.10, 'A');
INSERT INTO MY_DATA_ASSIGNMENT_TABLE VALUES (3, 3.10, 0.10, 'B');
INSERT INTO MY_DATA_ASSIGNMENT_TABLE VALUES (4, 10.10, 10.10, 'D');
INSERT INTO MY_DATA_ASSIGNMENT_TABLE VALUES (5, 3.10, -0.50, 'C');
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL (
"NAME" VARCHAR (50),
"INT_VALUE" INTEGER,
"DOUBLE_VALUE" DOUBLE,
"STRING_VALUE" VARCHAR (100)
);
CALL
_SYS_AFL.PAL_UNIFIED_CLUSTERING_ASSIGNMENT(MY_DATA_ASSIGNMENT_TABLE,PAL_DBSCAN_MO
DEL_TBL, PAL_PARAMETER_TBL, ?,?);
```

**Expected result**

RESULT:

| | DATA_ID | CLUSTER_ID | DISTANCE |
|---|---|---|---|
| 1 | 1 | 0 | 0 |
| 2 | 2 | 0 | 2.21 |
| 3 | 3 | -1 | 3 |
| 4 | 4 | 1 | 0.05000000000000071 |
| 5 | 5 | -1 | 3.59 |

## 5.13.5.3 Parameter Tables

This topic lists all parameters for unified clustering.

Mandatory Parameters

None

Optional Parameters

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| PRECALCU-LATED_DISTANCE | INTEGER | 0 | State of pre-computed distance matrix as input data:<br>• 0: invalid<br>• 1: valid | |

## AHC

Mandatory Parameters

None

Optional Parameters

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| CATEGORICAL_VARIA-BLE | VARCHAR | No default value | Indicates whether column data corresponds to a category variable even the data type of this column is INTEGER.<br>By default, VARCHAR and NVARCHAR are category variables, and INTEGER and DOUBLE are continuous variables. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| LINKAGE | INTEGER | 5 | Linkage type between two clusters.<br>• 1: nearest neighbor (single linkage)<br>• 2: furthest neighbor (complete linkage)<br>• 3: group average (UPGMA)<br>• 4: weighted average (WPGMA)<br>• 5: centroid clustering<br>• 6: median clustering<br>• 7: ward method<br>**Note**<br>For clustering method 5, 6, or 7, set the corresponding distance function (DISTANCE_LEVEL) to 8. | |
| N_CLUSTERS | INTEGER | 1 | Number of clusters after agglomerate hierarchical clustering algorithm.<br>Value range: between 1 and the initial number of input data. | |
| DISTANCE_DIMENSION | DOUBLE | 3 | Distance dimension can be set if DISTANCE_LEVEL is set to 3 (Minkowski Distance).<br>The value should be no less than 1. | Only valid when DISTANCE_LEVEL is 3. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| DISTANCE_LEVEL | INTEGER | 8 | Measure of distance between two clusters.<br>• 1: Manhattan Distance<br>• 2: Euclidean Distance<br>• 3: Minkowski Distance<br>• 4: Chebyshev Distance<br>• 6: Cosine<br>• 7: Pearson Correlation<br>• 8: Squared Euclidean Distance<br>• 9: Jaccard Distance<br>• 10: Gower Distance<br>**Note**<br>• For Jaccard Distance, non-zero input data will be treated as 1, and zero input data will be treated as 0.<br>Jaccard Distance = $(M_{01} + M_{10}) / (M_{11} + M_{01} + M_{10})$<br>• Only Gower Distance supports category attributes. | When LINKAGE is 5, 6, or 7, this parameter must be set to 8. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| NORMALIZATION | INTEGER | 0 | Normalization type:<br>• 0: does nothing<br>• 1: z score standardize<br>• 2: transforms to new range: -1–1<br>• 3: transforms to new range: 0–1 | |
| THREAD_RATIO | DOUBLE | -1.0 | The ratio of available threads for multi-thread tasks:<br>• 0: single thread<br>• 0–1: uses the specified percentage of available threads. PAL uses all available threads if the number is 1.0.<br>• -1: uses all of available threads. | |

## DBSCAN

Mandatory Parameters

| Name | Data Type | Description | Dependency |
|---|---|---|---|
| AUTO_PARAM | VARCHAR or NVARCHAR | Specifies whether the MINPTS and EPS parameters are determined automatically or by user input.<br>• true: automatically determines the parameters.<br>• false: uses parameter values provided by you. | |
| MINPTS | INTEGER | Specifies the minimum number of points required to form a cluster. | Only valid when AUTO_PARAM is false. |
| EPS | Double | Specifies the scan radius (eps). | Only valid when AUTO_PARAM is false. |

Optional Parameters

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| CATEGORICAL_VARIABLE | VARCHAR or NVARCHAR | No default value | Indicates whether the column data corresponds to a category variable even the data type of this column is INTEGER.<br>By default, VARCHAR and NVARCHAR are category variables. INTEGER and DOUBLE are continuous variables. | |
| CATEGORY_WEIGHTS | DOUBLE | 0.707 | Represents the weight of categorical variable (γ). | |
| DISTANCE_LEVEL | INTEGER | 2 | Specifies the method to compute the distance between two points.<br>• 1: Manhattan<br>• 2: Euclidean<br>• 3: Minkowski<br>• 4: Chebyshev<br>• 5: Standardized Euclidean<br>• 6: Cosine | |
| ALGORITHM | INTEGER | 1 | Searching method:<br>• 0: Brute force searching<br>• 1: KD-tree searching | |
| MINKOWSKI_POWER | INTEGER | 3 | Specifies the power of the Minkowski method. | Only valid when DISTANCE_LEVEL is 3. |
| SAVE_MODEL | INTEGER | 1 | Indicates whether model should be saved:<br>• 0: not save<br>• 1: save | |
| STRING_VARIABLE | NVARCHAR | No default value | Indicates a string column storing not categorical data. Levenshtein distance is used to calculate similarity between two strings. Ignored if it is not a string column. | If DISTANCE_LEVEL is set to 6 (COSINE), this parameter is not supported. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| VARIABLE_WEIGHT | NVARCHAR for variable name and DOUBLE for weight value | No default value | Specifies the weight of a variable participating in distance calculation.<br>The value must be equal to or greater than 0. Defaults to 1 for variables not specified. | |
| THREAD_RATIO | DOUBLE | -1.0 | The ratio of available threads for multi-thread tasks:<br>• 0: single thread<br>• 0–1: uses the specified percentage of available threads. PAL uses all available threads if the number is 1.0.<br>• -1: uses all of available threads. | |

## GMM

Mandatory Parameters

| Name | Data Type | Description | Dependency |
| --- | --- | --- | --- |
| INIT_MODE | INTEGER | Specifies the initialization mode.<br>• 0: sets the number of the Gaussian distributions of GMM in the initialize parameter table. The initial centers will be given by the farthest-first traversal algorithm.<br>• 1: specifies the ID of the initial centers in the initialize parameter table. The number of the components in GMM will be equal to the number of different initial centers that are given.<br>• 2: sets the number of the Gaussian distributions of GMM in the initialize parameter table. The responsibilities matrix (r_ik) is randomly initialized, which means the initial centers are the means of all the data that are randomly weighted.<br>• 3: sets the number of the Gaussian distributions of GMM in the initialize parameter table. The initial centers are given using the k-means++ approach. | |

| Name | Data Type | Description | Dependency |
|---|---|---|---|
| INITIALIZE_PARAMETER | INTEGER, VARCHAR, or NVARCHAR | Specifies the centers of data or the cluster number;<br>• 2nd column: number of data<br>• 4th column: specifies the cluster number or the center of data | When INIT_MODE is set to 0, 2, or 3, then this value is the number of clusters in GMM. In this case, only the first value of INITIALIZE_PARAMETER is used.<br>When INIT_MODE is set to 1, then values of INITIALIZE_PARAMETER are the "IDs" of the specified initial centers in the data table. Each value represents an initial center in the data table. The number of clusters is equal to the number of the given initial centers. For example, setting initial centers 1, 2, 10 means selecting the rows in the input data table whose value of the ID column is 1, 2, or 10, and the number of clusters will be 3. |

Optional Parameters

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| CATEGORICAL_VARIABLE | VARCHAR | No default value | Indicates whether or not a column data is actually corresponding to a category variable even the data type of this column is INTEGER. |
| CATEGORY_WEIGHT | DOUBLE | 0.707 | Represents the weight of category attributes $(\gamma)$. |
| COVARIANCE_TYPE | INTEGER | 0 | Indicates the type of covariance matrices in the model:<br>• 0: uses full covariance matrices.<br>• 1: uses diagonal covariance matrices.<br>• 2: uses diagonal covariance matrices with all equal diagonal entries. |
| ERROR_TOL | DOUBLE | 1.00E-05 | Specifies the error tolerance, which is the stop condition. |

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| MAX_ITER | INTEGER | 100 | Specifies the maximum iteration number of the EM algorithm. |
| REGULARIZATION | DOUBLE | 1.00E-06 | Regularization to be added to the diagonal of covariance matrices to ensure positive-definite. |
| SEED | INTEGER | 0 | Specifies the seed for random number generator:<br>• 0: uses the current time as the seed.<br>• Others: uses the specified value as the seed. |
| SHARED_COVARIANCE | INTEGER | 0 | Indicates whether all the clusters share the same covariance matrix:<br>• 0: Clusters have different covariance matrices.<br>• 1: Clusters share a same covariance matrix. |
| THREAD_RATIO | DOUBLE | -1.0 | The ratio of available threads for multi-thread tasks:<br>• 0: single thread<br>• 0–1: uses the specified percentage of available threads. PAL uses all available threads if the number is 1.0.<br>• -1: uses all of available threads. |

## SOM

Mandatory Parameters

None

Optional Parameters

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| ALPHA | DOUBLE | 0.5 | Specifies the learning rate. |

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| BATCH_SOM | INTEGER | 0 | Indicates whether batch SOM is carried out:<br>• 0: classical SOM<br>• 1: batch SOM<br>For batch SOM, KERNEL_FUNCTION is always Gaussian, and the LEARNING_RATE factors take no effect. |
| CONVERGENCE_CRITERION | DOUBLE | 1.00E-06 | If the largest difference of the successive maps is less than this value, the calculation is regarded as convergence, and SOM is completed consequently. |
| HEIGHT_OF_MAP | INTEGER | 10 | Indicates the height of the map. |
| KERNEL_FUNCTION | INTEGER | 1 | Represents the neighborhood kernel function:<br>• 1: Gaussian<br>• 2: Bubble/Flat |
| LEARNING_RATE | INTEGER | 1 | Indicates the decay function for learning rate:<br>• 1: exponential<br>• 2: linear |
| MAX_ITER | INTEGER | 1000 plus 500 times the number of neurons in the lattice | Maximum number of iterations.<br>**Note**<br>The training might not converge if this value is too small, for example, less than 1000. |
| NORMALIZATION | INTEGER | 0 | Normalization type:<br>• 0: No<br>• 1: Transforms to new range (0.0, 1.0)<br>• 2: z-score normalization |
| RADIUS | DOUBLE | The bigger value of HEIGHT_OF_MAP and WIDTH_OF_MAP | Specifies the scan radius. |

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| RANDOM_SEED | INTEGER | -1 | • -1: random<br>• 0: sets every weight to zero<br>• Other value: uses this value as seed |
| SHAPE_OF_GRID | INTEGER | 2 | Indicates the shape of the grid:<br>• 1: rectangle<br>• 2: hexagon |
| WIDTH_OF_MAP | INTEGER | 10 | Indicates the width of the map. |

## Accelerated K-Means

Mandatory Parameters

None

Optional Parameters

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| CATEGORICAL_VARIABLE | VARCHAR | No default value | Indicates whether or not a column data is actually corresponding to a category variable even the data type of this column is INTEGER. | |
| CATEGORY_WEIGHTS | DOUBLE | 0.707 | Represents the weight of category attributes ($\gamma$). | |
| DISTANCE_LEVEL | INTEGER | 2 | Computes the distance between the item and the cluster center:<br>• 1: Manhattan distance<br>• 2: Euclidean distance<br>• 3: Minkowski distance<br>• 4: Chebyshev distance<br>• 6: Cosine distance | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| TOL | DOUBLE | 1.00E-06 | Threshold (actual value) for exiting the iterations. | |
| N_CLUSTERS | INTEGER | No default value | Number of groups (k).<br>**Note**<br>If this parameter is not specified, you must specify the range of k using N_CLUSTERS_MIN and N_CLUSTERS_MAX. Then the algorithm will iterate through the range and return the k with the highest slight silhouette. | |
| N_CLUSTERS_MAX | INTEGER | No default value | Upper boundary of the range that k falls in.<br>**Note**<br>You must specify either an exact value or a range for k. If both are specified, the exact value will be used. | |
| N_CLUSTERS_MIN | INTEGER | No default value | Lower boundary of the range that k falls in.<br>**Note**<br>You must specify either an exact value or a range for k. If both are specified, the exact value will be used. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| INIT | INTEGER | 4 | Controls how the initial centers are selected:<br>• 1: first k observations<br>• 2: random with replacement<br>• 3: random without replacement<br>• 4: patent of selecting the init center (US 6,882,998 B1) | |
| MAX_ITER | INTEGER | 100 | Maximum iterations. | |
| MINKOWSKI_POWER | DOUBLE | 3 | When you use the Minkowski distance, this parameter controls the value of power. | Only valid when DISTANCE_LEVEL is 3. |
| NORMALIZATION | INTEGER | 0 | Normalization type:<br>• 0: no<br>• 1: yes. For each point `X (x1,x2,...,xn)`, the normalized value will be `X'(x1/S,x2/S,...,xn/S)`, where `S = \|x1\|+\|x2\|+...\|xn\|`.<br>• 2: for each column `C`, get the min and max value of `C`, and then `C[i] = (C[i]-min)/(max-min)`. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| THREAD_RATIO | DOUBLE | -1.0 | The ratio of available threads for multi-thread tasks:<br>• 0: single thread<br>• 0–1: uses the specified percentage of available threads. PAL uses all available threads if the number is 1.0.<br>• -1: uses all of available threads. | |

## K-Means

Mandatory Parameters

None

Optional Parameters

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| CATEGORICAL_VARIABLE | VARCHAR | No default value | Indicates whether or not a column data is actually corresponding to a category variable even the data type of this column is INTEGER. | |
| CATEGORY_WEIGHTS | DOUBLE | 0.707 | Represents the weight of category attributes ($\gamma$). | |
| DISTANCE_LEVEL | INTEGER | 2 | Computes the distance between the item and the cluster center:<br>• 1: Manhattan distance<br>• 2: Euclidean distance<br>• 3: Minkowski distance<br>• 4: Chebyshev distance<br>• 6: Cosine distance | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| TOL | DOUBLE | 1.00E-06 | Threshold (actual value) for exiting the iterations. | |
| N_CLUSTERS | INTEGER | No default value | Number of groups (k).<br>**Note**<br>If this parameter is not specified, you must specify the range of k using N_CLUSTERS_MIN and N_CLUSTERS_MAX. Then the algorithm will iterate through the range and return the k with the highest slight silhouette. | |
| N_CLUSTERS_MAX | INTEGER | No default value | Upper boundary of the range that k falls in.<br>**Note**<br>You must specify either an exact value or a range for k. If both are specified, the exact value will be used. | |
| N_CLUSTERS_MIN | INTEGER | No default value | Lower boundary of the range that k falls in.<br>**Note**<br>You must specify either an exact value or a range for k. If both are specified, the exact value will be used. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| INIT | INTEGER | 4 | Controls how the initial centers are selected:<br>• 1: first k observations<br>• 2: random with replacement<br>• 3: random without replacement<br>• 4: patent of selecting the init center (US 6,882,998 B1) | |
| MAX_ITER | INTEGER | 100 | Maximum iterations. | |
| MINKOWSKI_POWER | DOUBLE | 3 | When you use the Minkowski distance, this parameter controls the value of power. | Only valid when DISTANCE_LEVEL is 3. |
| NORMALIZATION | INTEGER | 0 | Normalization type:<br>• 0: no<br>• 1: yes. For each point X (x1,x2,...,xn), the normalized value will be X'(x1/S,x2/S,...,xn/S), where S = \|x1\|+\|x2\|+...\|xn\|.<br>• 2: for each column C, get the min and max value of C, and then C[i] = (C[i]-min)/(max-min). | |
| THREAD_RATIO | DOUBLE | -1.0 | The ratio of available threads for multi-thread tasks:<br>• 0: single thread<br>• 0–1: uses the specified percentage of available threads. PAL uses all available threads if the number is 1.0.<br>• -1: uses all of available threads. | |

## K-Medians

Mandatory Parameters

| Name | Data Type | Description |
|---|---|---|
| N_CLUSTERS | INTEGER | Number of groups (k). The value range is from 2 to the number of training records. |

Optional Parameters

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| CATEGORICAL_VARIABLE | NVARCHAR | No default value | Indicates whether or not a column data is actually corresponding to a category variable even the data type of this column is INTEGER. | |
| CATEGORY_WEIGHTS | DOUBLE | 0.707 | Represents the weight of category attributes (γ). | |
| DISTANCE_LEVEL | INTEGER | 2 | Computes the distance between the item and the cluster center:<br>• 1: Manhattan distance<br>• 2: Euclidean distance<br>• 3: Minkowski distance<br>• 4: Chebyshev distance<br>• 6: Cosine distance | |
| TOL | DOUBLE | 1.00E-06 | Threshold (actual value) for exiting the iterations. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| INIT | INTEGER | 4 | Controls how the initial centers are selected:<br>• 1: first k observations<br>• 2: random with replacement<br>• 3: random without replacement<br>• 4: patent of selecting the initial center (US 6,882,998 B1) | |
| MAX_ITER | INTEGER | 100 | Maximum iterations. | |
| MINKOWSKI_POWER | DOUBLE | 3 | When you use the Minkowski distance, this parameter controls the value of power. | Only valid when DISTANCE_LEVEL is 3. |
| NORMALIZATION | INTEGER | 0 | Normalization type:<br>• 0: no<br>• 1: yes. For each point X (x1,x2,...,xn), the normalized value will be X'(x1/S,x2/S,...,xn/S), where S = \|x1\|+\|x2\|+...\|xn\|.<br>• 2: for each column C, get the min and max value of C, and then C[i] = (C[i]-min)/(max-min). | |
| THREAD_RATIO | DOUBLE | -1.0 | The ratio of available threads for multi-thread tasks:<br>• 0: single thread<br>• 0–1: uses the specified percentage of available threads. PAL uses all available threads if the number is 1.0.<br>• -1: uses all of available threads. | |

## K-Medoids

Mandatory Parameters

| Name | Data Type | Description |
|---|---|---|
| N_CLUSTERS | INTEGER | Number of groups (k). The value range is from 2 to the number of training records. |

Optional Parameters

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| CATEGORICAL_VARIABLE | NVARCHAR | No default value | Indicates whether or not a column data is actually corresponding to a category variable even the data type of this column is INTEGER. | |
| CATEGORY_WEIGHTS | DOUBLE | 0.707 | Represents the weight of category attributes (γ). | |
| DISTANCE_LEVEL | INTEGER | 2 | Computes the distance between the item and the cluster center:<br>• 1: Manhattan distance<br>• 2: Euclidean distance<br>• 3: Minkowski distance<br>• 4: Chebyshev distance<br>• 5: Jaccard distance<br>• 6: Cosine distance | When Jaccard distance is used, all columns must be categorical. |
| TOL | DOUBLE | 1.00E-06 | Threshold (actual value) for exiting the iterations. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| INIT | INTEGER | 4 | Controls how the initial centers are selected:<br>• 1: first k observations<br>• 2: random with replacement<br>• 3: random without replacement<br>• 4: patent of selecting the init center (US 6,882,998 B1) | |
| MAX_ITER | INTEGER | 100 | Maximum iterations. | |
| MINKOWSKI_POWER | DOUBLE | 3 | When you use the Minkowski distance, this parameter controls the value of power. | Only valid when DISTANCE_LEVEL is 3. |
| NORMALIZATION | INTEGER | 0 | Normalization type:<br>• 0: no<br>• 1: yes. For each point X (x1,x2,...,xn), the normalized value will be X'(x1/S,x2/S,...,xn/S), where S = \|x1\|+\|x2\|+...\|xn\|.<br>• 2: for each column C, get the min and max value of C, and then C[i] = (C[i]-min)/(max-min). | |
| THREAD_RATIO | DOUBLE | -1.0 | The ratio of available threads for multi-thread tasks:<br>• 0: single thread<br>• 0–1: uses the specified percentage of available threads. PAL uses all available threads if the number is 1.0.<br>• -1: uses all of available threads. | |

## AP

Mandatory Parameters

| Name | Data Type | Description |
|---|---|---|
| DISTANCE_LEVEL | INTEGER | Sepcifies the method to compute the distance between two points.<br>• 1: Manhattan<br>• 2: Euclidean<br>• 3: Minkowski<br>• 4: Chebyshev<br>• 5: Standardized Euclidean<br>• 6: Cosine |
| CLUSTER_NUMBER | INTEGER | Number of cluster.<br>• 0: does not adjust AP Cluster result;<br>• Non-zero INTEGER: if AP cluster number is greater than CLUSTER_NUMBER, PAL will merge the result to set the cluster number to the value specified for CLUSTER_NUMBER. |

Optional Parameters

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| INITIALIZE_SEED | NVARCHAR | No default value | Specifies the record id. | |
| MAX_ITER | INTEGER | 500 | Maximum iterations. | |
| CON_INTERATION | INTEGER | 100 | When the clusters keep a steady one for the specified times, the algorithm ends. | |
| DAMP | DOUBLE | 0.9 | Controls the updating velocity.<br>Value range: 0 < DAMP < 1 | |
| PREFERENCE | DOUBLE | 0.5 | Determines the preference.<br>Value range: 0 <= PREFERENCE <= 1 | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| SEED_RATION | DOUBLE | 1 | Select a portion of (SEED_RATION * data_number) the input data as seed, where data_number is the row_size of the input data.<br>Value range: 0< SEED_RATION <= 1 | Only valid when the INITIALIZE_SEED is not define. If SEED_RATIO is 1, all the input data will be the seed. |
| TIMES | INTEGER | 1 | The sampling times. | Only valid when the INITIALIZE_SEED is not define and SEED_RATIO is less than 1. |
| MINKOW_P | INTEGER | 3 | The power of the Minkowski method. | Only valid when DISTANCE_LEVEL is 3. |
| MINKOWSKI_POWER | DOUBLE | 3 | When you use the Minkowski distance, this parameter controls the value of power. | Only valid when DISTANCE_LEVEL is 3. |
| THREAD_RATIO | DOUBLE | -1.0 | The ratio of available threads for multi-thread tasks:<br>• 0: single thread<br>• 0–1: uses the specified percentage of available threads. PAL uses all available threads if the number is 1.0.<br>• -1: uses all of available threads. | |

## SP

Mandadory Parameters

| Name | Data Type | Description |
|---|---|---|
| N_CLUSTERS | INTEGER | Number of groups. The value range is from 2 to the number of training records. |

Optional Parameters

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| THREAD_RATIO | DOUBLE | -1.0 | Specifies the ratio of total number of threads that can be used by this function.<br>• 0: single thread<br>• 0–1: uses the specified percentage of available threads. PAL uses all available threads if the number is 1.0.<br>• -1: uses all of available threads. | |
| N_COMPONENTS | INTEGER | GROUP_NUMBER | Number of eigenvectors to use for the spectral embedding. | |
| SPLIT_THRESHOLD | DOUBLE | 1.0 | Kernel coefficient of RBF(radia basis function) used in constructing affinity matrix construction with distance d: exp(-gamma*d^2). | |
| AFFINITY | INTEGER | 2 | How to construct the affinity matrix: <ul><li>0: k-nearest neighbor graph</li><li>1: mutual k-nearest neighbor graph</li><li>2: full connected graph</li></ul> | |
| N_NEIGHBOURS | INTEGER | 10 | Number of neighbors to use while constructing the affinity matrix using nearest neighbors method. | Only valid when AFFINITY is 0 or 1. |
| CUT_METHOD | INTEGER | 0 | How to cut the graph:<ul><li>0: RatioCut</li><li>1: NCut</li> | |
| EIGEN_TOL | DOUBLE | 1e-10 | Stopping criterion for eigendecomposition of the Laplacian matrix. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| KRYLOV_DIMENSION | INTEGER | N_CONPONENTS*2 | Parameter that controls the convergence speed of the algorithm. Typically a larger KRYLOY_DIMMENSION means faster convergence, but it may also result in greater memory use and more matrix operations in each iteration.<br>**Note**<br>This parameter must satifiy N_COMPONENTS< KRYLOY_DIMENSION≤ the number of training records. | |
| DISTANCE_LEVEL | INTEGER | 2 | Computes the distance between the item and the cluster center:<ul><li>1: Manhattan distance</li><li>2: Euclidean distance</li><li>3: Minkowski distance</li><li>4: Chebyshev distance</li><li>6: Cosine distance</li></ul> | |
| CATEGORY_WEIGHTS | DOUBLE | 0.707 | Represents the weight of category attributes. | |
| MINKOWSKI_POWER | DOUBLE | 3.0 | When you use the Minkowski distance, this parameter controls the value of power. | Only valid when DISTANCE_LEVEL is 3. |
| MAX_ITER | INTEGER | 100 | Maximum iterations for K-Means. | |
| INIT | INTEGER | 4 | INIT_TYPE for K-Means. | |
| TOL | DOUBLE | 1e-6 | Threshold (actual value) for exiting the iterations of K-Means. | |

## 5.13.5.4 Precomputed Distance Matrix as Input Data

You can use some clustering algorithms in the unified clustering by using precomputed distance matrix as input data.

To perform unified clustering with precomputed distance matrix, the input data must be either upper or lower triangular, which means that the expanded shape needs to be N samples * N samples and the distance value of the pair of (i,j) or (j,i) must be unique.

Currently, unified clustering with pre-computed distance matrix supports K-Medoids, AHC and SP algorithms. Please note:

1. The format of the input data table is different from the format of the feature columns of the input data table;
2. For SP, 0 value means the pair of points is disconnected, but for other algorithms, it means very close.

### Clustering

#### Input Table

| Table | Column | Data Type | Description | Constraint |
|---|---|---|---|---|
| DATA | 1st column | INTEGER, VARCHAR, or NVARCHAR | Left Point | |
| | 2nd column | INTEGER, VARCHAR, or NVARCHAR | Right Point | The type should be the same as the left point type. |
| | 3rd column | DOUBLE | Distance | |

#### Parameter Table

Mandatory Parameters

None

Optional Parameters

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| THREAD_RATIO | DOUBLE | -1.0 | The ratio of available threads for multi-thread tasks:<br>• 0: single thread;<br>• 0~1: percentage;<br>• others: heuristically determines the number of threads to use. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| PRECALCU-LATED_DISTANCE | INTEGER | 0 | State of precomputed distance matrix as input data:<br>• 0: invalid;<br>• 1: valid; | |

### KMEDOIDS

Mandatory Parameters

| Name | Data Type | Description |
|---|---|---|
| N_CLUSTERS | INTEGER | Number of groups (k). The value range is from 2 to the number of training records. |

Optional Parameters

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| TOL | DOUBLE | 1.00E-06 | Threshold (actual value) for exiting the iterations. | |
| INIT | INTEGER | 1 | Controls how the initial centers are selected:<br>• 1: First k observations;<br>• 2: Random with replacement;<br>• 3: Random without replacement. | |
| RANDOM_SEED | INTEGER | 0 | Indicates the seed used to initialize the random number generator. It can be set to 0 or a positive value.<br>• 0: Uses the system time;<br>• Nonzero: Uses the specified seed. | |
| MAX_ITER | INTEGER | 100 | Maximum iterations. | |

### SP

Mandatory Parameters

| Name | Data Type | Description |
|---|---|---|
| N_CLUSTERS | INTEGER | Number of groups (k). The value range is from 2 to the number of training records. |

Optional Parameters

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| N_COMPONENTS | INTEGER | GROUP_NUMBER | Number of eigenvectors to use for the spectral embedding. | |
| CUT_METHOD | INTEGER | 0 | How to cut the graph:<ul><li>0: RatioCut</li><li>1: NCut</li> | |
| EIGEN_TOL | DOUBLE | 1e-10 | Stopping criterion for eigendecomposition of the Laplacian matrix. | |
| KRYLOV_DIMENSION | INTEGER | N_CONPONENTS*2 | Parameter that controls the convergence speed of the algorithm. Typically a larger KRYLOY_DIMMENSION means faster convergence, but it may also result in greater memory use and more matrix operations in each iteration.<br>**Note**<br>This parameter must satifiy N_COMPONENTS< KRYLOY_DIMENSION≤ the number of training records. | |
| MAX_ITER | INTEGER | 100 | Maximum iterations for K-Means. | |
| INIT | INTEGER | 4 | INIT_TYPE for K-Means. | |
| TOL | DOUBLE | 1e-6 | Threshold (actual value) for exiting the iterations of K-Means. | |

## AHC

Mandatory Parameters

None.

Optional Parameters

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| N_CLUSTERS | INTEGER | 1 | Number of clusters after agglomerate hierarchical clustering algorithm. <br>Value range: between 1 and the initial number of input data.</br> | |
| LINKAGE | INTEGER | 5 | Linkage type between two clusters.<ul><li>5: Centroid Clustering</li><li>6: Median Clustering</li><li>7: Ward Method</li></ul> | |

### Examples

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

Example 1: K-Medoids

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_PRECALCULATED_DISTANCE;
CREATE COLUMN TABLE PAL_PRECALCULATED_DISTANCE (
    "LEFT_POINT"  NVARCHAR(5),
    "RIGHT_POINT" NVARCHAR(5),
    "DISTANCE"    DOUBLE
);
--Define the pair of points and the distance between them.
--The type of point could be int or string.
INSERT INTO PAL_PRECALCULATED_DISTANCE VALUES('0', '1',  0.707106781);
INSERT INTO PAL_PRECALCULATED_DISTANCE VALUES('0', '2',  12.74754878);
INSERT INTO PAL_PRECALCULATED_DISTANCE VALUES('0', '3',  12.72792206);
INSERT INTO PAL_PRECALCULATED_DISTANCE VALUES('1', '2',  12.72792206);
INSERT INTO PAL_PRECALCULATED_DISTANCE VALUES('1', '3',  12.74754878);
INSERT INTO PAL_PRECALCULATED_DISTANCE VALUES('2', '3',  0.707106781);
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL (
"NAME" VARCHAR (50),
"INT_VALUE" INTEGER,
"DOUBLE_VALUE" DOUBLE,
"STRING_VALUE" VARCHAR (100)
);
INSERT INTO PAL_PARAMETER_TBL VALUES ('FUNCTION', NULL, NULL, 'KMEDOIDS');
INSERT INTO PAL_PARAMETER_TBL VALUES ('N_CLUSTERS', 2, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('INIT', 2, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('PRECALCULATED_DISTANCE', 1, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('RANDOM_SEED', 10, NULL, NULL);
DROP TABLE PAL_CLUSTERING_MODEL_TBL;
CREATE COLUMN TABLE PAL_CLUSTERING_MODEL_TBL (
```

```
"ROW_INDEX" INTEGER,
"PART_INDEX" INTEGER,
"MODEL_CONTENT" VARCHAR (5000)
);
DO BEGIN
lt_data = SELECT * FROM PAL_PRECALCULATED_DISTANCE;
lt_param = SELECT * FROM PAL_PARAMETER_TBL;
CALL "_SYS_AFL"."PAL_UNIFIED_CLUSTERING"(:lt_data, :lt_param, lt_result,
lt_centers, lt_model, lt_stat, lt_optimal_parameter, lt_place1, lt_place2);
INSERT INTO PAL_CLUSTERING_MODEL_TBL SELECT * FROM :lt_model;
END;
```

**Expected Result**:

RESULT:

Result 1 × Result 2 × Result 3 × Result 4 × Messages × History

Rows (4)

| | LEFT_POINT | CLUSTER_ID | DISTANCE | SLIGHT_SILHOUETTE |
|---|---|---|---|---|
| 1 | 3 | 1 | 0 | *NULL* |
| 2 | 1 | 0 | 0 | *NULL* |
| 3 | 2 | 1 | 0.707106781 | *NULL* |
| 4 | 0 | 0 | 0.707106781 | *NULL* |

CENTERS:

Result 1 × Result 2 × Result 3 × Result 4 × Messages × History

Rows (2)

| | CLUSTER_ID | VARIABLE_NAME | VALUE |
|---|---|---|---|
| 1 | 0 | LEFT_POINT | 1 |
| 2 | 1 | LEFT_POINT | 3 |

MODEL:

Result 1 × Result 2 × Result 3 × Result 4 × Messages × History

Rows (1)

| | ROW_INDEX | PART_INDEX | MODEL_CONTENT |
|---|---|---|---|
| 1 | 0 | *NULL* | {"Algorithm":"KMEDOIDSWITHDIS","ColumnInfo":{"DataType":83},"Cluster":[{"ClusterID":0,"CenterCoor":"1"},{"ClusterID":1,"CenterCoor":"3"}]} |

Example 2: SP

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_PRECALCULATED_DISTANCE;
CREATE COLUMN TABLE PAL_PRECALCULATED_DISTANCE(
"LEFT_POINT" INTEGER,
"RIGHT_POINT" INTEGER,
"DISTANCE" DOUBLE
);
-- If the distance value == 0, there is no need to input this unconnected point.
INSERT INTO PAL_PRECALCULATED_DISTANCE VALUES(0, 1, 0.707106781);
INSERT INTO PAL_PRECALCULATED_DISTANCE VALUES(2, 3, 0.707106781);
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL(
"NAME" VARCHAR(50),
"INT_VALUE" INTEGER,
"DOUBLE_VALUE" DOUBLE,
"STRING_VALUE" VARCHAR(100)
);
INSERT INTO PAL_PARAMETER_TBL VALUES('FUNCTION', NULL, NULL, 'SP');
INSERT INTO PAL_PARAMETER_TBL VALUES('PRECALCULATED_DISTANCE', 1, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES('N_CLUSTERS', 2, NULL, NULL);
```

```
DROP TABLE PAL_CLUSTERING_MODEL_TBL;
CREATE COLUMN TABLE PAL_CLUSTERING_MODEL_TBL(
"ROW_INDEX" INTEGER,
"PART_INDEX" INTEGER,
"MODEL_CONTENT" VARCHAR(5000)
);
DO BEGIN
lt_data = SELECT * FROM PAL_PRECALCULATED_DISTANCE;
lt_param = SELECT * FROM PAL_PARAMETER_TBL;
CALL "_SYS_AFL"."PAL_UNIFIED_CLUSTERING"(:lt_data, :lt_param, lt_result,
lt_centers, lt_model, lt_stat, lt_optimal_parameter, lt_place1, lt_place2);
INSERT INTO PAL_CLUSTERING_MODEL_TBL
SELECT * FROM :lt_model;
SELECT * FROM :lt_result;
END;
```

**Expected Result:**

RESULT:

Result 1 × Result 2 × Result 3 × Result 4 × Messages × History

Rows (4) SQL

| | LEFT_POINT | CLUSTER_ID | DISTANCE | SLIGHT_SILHOUETTE |
|---|---|---|---|---|
| 1 | 3 | 0 | NULL | NULL |
| 2 | 2 | 0 | NULL | NULL |
| 3 | 1 | 1 | NULL | NULL |
| 4 | 0 | 1 | NULL | NULL |

## Clustering Assignment

To perform clustering assignment, the input structure needs to be the same as the clustering input structure and the model needs to be generated by the distance matrix procedure.

**Input Table**

| Table | Column | Data Type | Description | Constraint |
|---|---|---|---|---|
| DATA | 1st column | INTEGER, VARCHAR, or NVARCHAR | Left Point | |
| | 2nd column | INTEGER, VARCHAR, or NVARCHAR | Right Point | The type should be the same as the left point type. |
| | 3rd column | DOUBLE | Distance | |

**Parameter Table**

Mandatory Parameters

None.

Optional Parameters

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| THREAD_RATIO | DOUBLE | -1.0 | The ratio of available threads for multi-thread tasks:<br>• 0: single thread;<br>• 0~1: percentage;<br>• others: heuristically determines the number of threads to use. | |

### Examples

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
DROP TABLE MY_DATA_ASSIGNMENT_TABLE;
CREATE COLUMN TABLE MY_DATA_ASSIGNMENT_TABLE (
    "LEFT_POINT"  NVARCHAR(5),
    "RIGHT_POINT" NVARCHAR(5),
    "DISTANCE"    DOUBLE
);
--The structure of input data needs be the same as the clustering input data.
--At least, The input data should contain the centers point.
--For this example, each assignment point(4, 5 point) should contain '3' and '1'
points.
INSERT INTO MY_DATA_ASSIGNMENT_TABLE VALUES('5', '2',  1.444);
INSERT INTO MY_DATA_ASSIGNMENT_TABLE VALUES('5', '1',  2.444);
INSERT INTO MY_DATA_ASSIGNMENT_TABLE VALUES('5', '3',  1.444);
INSERT INTO MY_DATA_ASSIGNMENT_TABLE VALUES('4', '1',  0.555);
INSERT INTO MY_DATA_ASSIGNMENT_TABLE VALUES('4', '3',  10.444);
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL (
"NAME" VARCHAR (50),
"INT_VALUE" INTEGER,
"DOUBLE_VALUE" DOUBLE,
"STRING_VALUE" VARCHAR (100)
);
CALL
_SYS_AFL.PAL_UNIFIED_CLUSTERING_ASSIGNMENT(MY_DATA_ASSIGNMENT_TABLE,PAL_CLUSTERIN
G_MODEL_TBL, PAL_PARAMETER_TBL, ?,?);
```

**Expected Result**:

RESULT:

Result 1 × Result 2 × Messages × History

Rows (5) SQL

| | LEFT_POINT | CLUSTER_ID | DISTANCE |
|---|---|---|---|
| 1 | 3 | 1 | 0 |
| 2 | 1 | 0 | 0 |
| 3 | 4 | 0 | 0.555 |
| 4 | 2 | -1 | 1.7976931348623157e+308 |
| 5 | 5 | 1 | 1.444 |

## 5.13.6 Unified Clustering with Pivoted Input Data

Unified clustering pivot is a unified interface to call some clustering algorithms with pivot data format.

The clustering algorithms include:

- AHC (Agglomerate Hierarchical Clustering)
- DBSCAN (Density-Based Spatial Clustering of Applications with Noise)
- GMM (Gaussian Mixture Model)
- AKMEANS (Accelerated K-Means)
- KMEANS(K-Means)
- KMEDIANS(K-Medians)
- KMEDOIDS(K-Medoids)
- SOM (Self-Organizing Maps)
- AP (Affinity Propagation)
- SP (Spectral Clustering)

For GMM, you must configure the INITIALIZE_PARAMETER parameters to define the INITIALIZE_PARAMETER table of the original method.

For AP, you must configure the INITIALIZE_SEED parameters to define the SEED table of the original method.

Compared to the original accelerated K-Means, K-Medians, and K-Medoids, unified clustering with pivoted input creates a model after a training and then performs cluster assignment through the previously created model.

See the corresponding topic for details about a specific clustering method.

### Prerequisites

- All prerequisites for the original corresponding cluster function apply in the unified cluster interface.

### _SYS_AFL.PAL_UNIFIED_CLUSTERING_PIVOT

#### Overviews of ALL Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<METADATA Table>` | IN |
| 2 | `<DATA Table>` | IN |
| 3 | `<PARAMETER Table>` | IN |
| 4 | `<RESULT Table>` | OUT |

| Position | Table Name | Parameter Type |
|---|---|---|
| 5 | `<CENTERS Table>` | OUT |
| 6 | `<MODEL Table>` | OUT |
| 7 | `<STATISTICS Table>` | OUT |
| 8 | `<OPTIMAL_PARAMETER Table>` | OUT |
| 9 | `<PLACE_HOLDER_1 Table>` | OUT |
| 10 | `<PLACE_HOLDER_2 Table>` | OUT |

### Input Tables

| Table | Column | Reference for Output Table | Data Type | Description | Constraint |
|---|---|---|---|---|---|
| METADATA | 1st column | VARIABLE_NAME | VARCHAR or NVARCHAR | The name of variable. | |
| | 2nd column | VARIABLE_TYPE | VARCHAR or NVARCHAR | The type of variable:<br>• CONTINUOUS<br>• CATEGORICAL | |
| DATA | 1st column | DATA_ID | BIGINT, INTEGER, VARCHAR, or NVARCHAR | Data ID. | This must be the first column. |
| | 2nd column | VARIABLE_NAME | VARCHAR or NVARCHAR | The name of variable. | Should be identical to the model. |
| | 3rd column | VALUE | VARCHAR or NVARCHAR | The value of variable. | |
| | 4th column | PURPOSE | INTEGER | The purpose of feature:<br>• 1: Clustering<br>• 2: Testing | So far, only support clustering. |

### Parameter Table

Mandatory Parameters

The following parameters are mandatory and must be given a value.

| Name | Data Type | Description | Dependency |
|---|---|---|---|
| FUNCTION | VARHCAR or NVARCHAR | The algorithm name, including:<br>• KMEANS<br>• AKMEANS<br>• KMEDIANS<br>• KMEDOIDS<br>• AHC<br>• DBSCAN<br>• SOM<br>• GMM<br>• AP<br>• SP | |

See Parameter Tables [page 1843] for parameter details about a specific algorithm.

### Output Table

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| RESULT | 1st column | DATA_ID (in input table) data type | DATA_ID (in input table) column name | Data ID. |
| | 2nd column | INTEGER | CLUSTER_ID | The assigned cluster ID. |
| | 3rd column | DOUBLE | DISTANCE | Distance between a given point and the<br>• cluster center (k-means)<br>• nearest core object (DBSCAN)<br>• weight vector (SOM)<br>Or probability of a given point belonging to the corresponding cluster (GMM). |
| | 4th column | DOUBLE | SLIGHT_SILHOUETTE | Estimated value (slight silhouette). |
| CENTERS | 1st column | INTEGER | CLUSTER_ID | |
| | 2nd column | VARCHAR or NVARCHAR | VARIABLE_NAME | The name of variable. |
| | 3rd column | VARCHAR or NVARCHAR | VALUE | The value of variable. |
| MODEL | 1st column | INTEGER | ROW_INDEX | Row index. |
| | 2nd column | INTEGER | PART_INDEX | Specifically for GMM's CLUSTER_ID. |

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| | 3rd column | NCLOB | MODEL_CONTENT | Model content. |
| STATISTICS | 1st column | VARHCAR or NVARCHAR | STATISTIC_NAME | |
| | 2nd column | VARHCAR or NVARCHAR | STATISTIC_VALUE | |
| OPTIMAL_PARAME-TER | 1st column | VARHCAR or NVARCHAR | NAME | Placeholder. |
| | 2nd column | INTEGER | INT_VALUE | |
| | 3rd column | DOUBLE | DOUBLE_VALUE | |
| | 4th column | VARHCAR or NVARCHAR | STRING_VALUE | |
| PLACE_HOLDER_1 | 1st column | DATA_ID (in input table) data type | DATA_ID (in input table) column name | |
| | 2nd column | INTEGER | TYPE | |
| PLACE_HOLDER_2 | 1st column | VARHCAR or NVARCHAR | OBJECT | Placeholder for future features. |
| | 2nd column | VARHCAR or NVARCHAR | KEY | |
| | 3rd column | VARHCAR or NVARCHAR | VALUE | |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

Example: K-Means

```
SET SCHEMA DM_PAL;
DROP TABLE META_DATA_TABLE;
CREATE COLUMN TABLE META_DATA_TABLE (
"VARIABLE_NAME" VARCHAR (100),
"VARIABLE_VALUE" VARCHAR (100)
);
INSERT INTO META_DATA_TABLE VALUES ('V000', 'CONTINUOUS');
INSERT INTO META_DATA_TABLE VALUES ('V001', 'CATEGORICAL');
INSERT INTO META_DATA_TABLE VALUES ('V002', 'CONTINUOUS');
DROP TABLE MY_DATA_TABLE;
CREATE COLUMN TABLE MY_DATA_TABLE (
"ID" INTEGER,
"NAME" VARCHAR (100),
"VALUE" VARCHAR (100),
"PURPOSE" INTEGER
);
INSERT INTO MY_DATA_TABLE VALUES (0, 'V000','0.5',1);
INSERT INTO MY_DATA_TABLE VALUES (0, 'V001','A',1);
INSERT INTO MY_DATA_TABLE VALUES (0, 'V002','0.5',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 'V000','1.5',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 'V001','A',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 'V002','0.5',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 'V000','1.5',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 'V001','A',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 'V002','1.5',1);
INSERT INTO MY_DATA_TABLE VALUES (3, 'V000','0.5',1);
INSERT INTO MY_DATA_TABLE VALUES (3, 'V001','A',1);
INSERT INTO MY_DATA_TABLE VALUES (3, 'V002','1.5',1);
```

```
INSERT INTO MY_DATA_TABLE VALUES (4, 'V000','1.1',1);
INSERT INTO MY_DATA_TABLE VALUES (4, 'V001','B',1);
INSERT INTO MY_DATA_TABLE VALUES (4, 'V002','1.2',1);
INSERT INTO MY_DATA_TABLE VALUES (5, 'V000','0.5',1);
INSERT INTO MY_DATA_TABLE VALUES (5, 'V001','B',1);
INSERT INTO MY_DATA_TABLE VALUES (5, 'V002','15.5',1);
INSERT INTO MY_DATA_TABLE VALUES (6, 'V000','1.5',1);
INSERT INTO MY_DATA_TABLE VALUES (6, 'V001','B',1);
INSERT INTO MY_DATA_TABLE VALUES (6, 'V002','15.5',1);
INSERT INTO MY_DATA_TABLE VALUES (7, 'V000','1.5',1);
INSERT INTO MY_DATA_TABLE VALUES (7, 'V001','B',1);
INSERT INTO MY_DATA_TABLE VALUES (7, 'V002','16.5',1);
INSERT INTO MY_DATA_TABLE VALUES (8, 'V000','0.5',1);
INSERT INTO MY_DATA_TABLE VALUES (8, 'V001','B',1);
INSERT INTO MY_DATA_TABLE VALUES (8, 'V002','16.5',1);
INSERT INTO MY_DATA_TABLE VALUES (9, 'V000','1.2',1);
INSERT INTO MY_DATA_TABLE VALUES (9, 'V001','C',1);
INSERT INTO MY_DATA_TABLE VALUES (9, 'V002','16.1',1);
INSERT INTO MY_DATA_TABLE VALUES (10, 'V000','15.5',1);
INSERT INTO MY_DATA_TABLE VALUES (10, 'V001','C',1);
INSERT INTO MY_DATA_TABLE VALUES (10, 'V002','15.5',1);
INSERT INTO MY_DATA_TABLE VALUES (11, 'V000','16.5',1);
INSERT INTO MY_DATA_TABLE VALUES (11, 'V001','C',1);
INSERT INTO MY_DATA_TABLE VALUES (11, 'V002','15.5',1);
INSERT INTO MY_DATA_TABLE VALUES (12, 'V000','16.5',1);
INSERT INTO MY_DATA_TABLE VALUES (12, 'V001','C',1);
INSERT INTO MY_DATA_TABLE VALUES (12, 'V002','16.5',1);
INSERT INTO MY_DATA_TABLE VALUES (13, 'V000','15.5',1);
INSERT INTO MY_DATA_TABLE VALUES (13, 'V001','C',1);
INSERT INTO MY_DATA_TABLE VALUES (13, 'V002','16.5',1);
INSERT INTO MY_DATA_TABLE VALUES (14, 'V000','15.6',1);
INSERT INTO MY_DATA_TABLE VALUES (14, 'V001','D',1);
INSERT INTO MY_DATA_TABLE VALUES (14, 'V002','16.2',1);
INSERT INTO MY_DATA_TABLE VALUES (15, 'V000','15.5',1);
INSERT INTO MY_DATA_TABLE VALUES (15, 'V001','D',1);
INSERT INTO MY_DATA_TABLE VALUES (15, 'V002','0.5',1);
INSERT INTO MY_DATA_TABLE VALUES (16, 'V000','16.5',1);
INSERT INTO MY_DATA_TABLE VALUES (16, 'V001','D',1);
INSERT INTO MY_DATA_TABLE VALUES (16, 'V002','0.5',1);
INSERT INTO MY_DATA_TABLE VALUES (17, 'V000','16.5',1);
INSERT INTO MY_DATA_TABLE VALUES (17, 'V001','D',1);
INSERT INTO MY_DATA_TABLE VALUES (17, 'V002','1.5',1);
INSERT INTO MY_DATA_TABLE VALUES (18, 'V000','15.5',1);
INSERT INTO MY_DATA_TABLE VALUES (18, 'V001','D',1);
INSERT INTO MY_DATA_TABLE VALUES (18, 'V002','1.5',1);
INSERT INTO MY_DATA_TABLE VALUES (19, 'V000','15.7',1);
INSERT INTO MY_DATA_TABLE VALUES (19, 'V001','A',1);
INSERT INTO MY_DATA_TABLE VALUES (19, 'V002','1.6',1);
--parameter table
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL (
"NAME" VARCHAR (50),
"INT_VALUE" INTEGER,
"DOUBLE_VALUE" DOUBLE,
"STRING_VALUE" VARCHAR (100)
);
INSERT INTO PAL_PARAMETER_TBL VALUES ('FUNCTION', NULL, NULL, 'KMEANS');
INSERT INTO PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 1.0, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('N_CLUSTERS', 4, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('INIT', 1, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('DISTANCE_LEVEL',2, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('MAX_ITER', 100, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('TOL', NULL, 1.0E-6, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('CATEGORY_WEIGHTS', NULL, 0.5, NULL);
CALL _SYS_AFL.PAL_UNIFIED_CLUSTERING_PIVOT(META_DATA_TABLE,MY_DATA_TABLE,
PAL_PARAMETER_TBL, ?,?,?,?,?,?,?);
```

### Expected Result

RESULT:

| | ID | CLUSTER_ID | DISTANCE | SLIGHT_SILHOUETE |
|---|---|---|---|---|
| 1 | 0 | 0 | 0.8910879487969621 | 0.9443700340289641 |
| 2 | 1 | 0 | 0.8639170309648472 | 0.9424783519324079 |
| 3 | 2 | 0 | 0.8062521617810958 | 0.9462881150703033 |
| 4 | 3 | 0 | 0.8356835545457009 | 0.944941534308603 |
| 5 | 4 | 0 | 0.7445708631492213 | 0.9502343526704448 |
| 6 | 5 | 3 | 0.8910879487969616 | 0.9407325552117667 |
| 7 | 6 | 3 | 0.8356835545457004 | 0.9444124645366334 |
| 8 | 7 | 3 | 0.8062521617810963 | 0.9465187213724425 |
| 9 | 8 | 3 | 0.8639170309648476 | 0.9461211197877126 |
| 10 | 9 | 3 | 0.744570863149222 | 0.9498994935338706 |
| 11 | 10 | 2 | 0.8255266112967905 | 0.945092293345542 |
| 12 | 11 | 2 | 0.9338858664836682 | 0.9379022885331548 |
| 13 | 12 | 2 | 0.8816915771701814 | 0.9450081614922042 |
| 14 | 13 | 2 | 0.7643178163332063 | 0.9491595311531508 |
| 15 | 14 | 2 | 0.9234563013492032 | 0.939282992463176 |
| 16 | 15 | 1 | 0.9016844685872387 | 0.940436243125126 |
| 17 | 16 | 1 | 0.9768852992055255 | 0.9393860344624405 |
| 18 | 17 | 1 | 0.8181783288648536 | 0.9458778291291957 |
| 19 | 18 | 1 | 0.7227990303872558 | 0.9521697340450603 |
| 20 | 19 | 1 | 1.1023417395491886 | 0.9256786403653229 |

CENTERS:

| | CLUSTER_ID | VARIABLE_NAME | VALUE |
|---|---|---|---|
| 1 | 0 | V000 | 1.02 |
| 2 | 0 | V001 | A |
| 3 | 0 | V002 | 1.04 |
| 4 | 1 | V000 | 15.94... |
| 5 | 1 | V001 | D |
| 6 | 1 | V002 | 1.119... |
| 7 | 2 | V000 | 15.91... |
| 8 | 2 | V001 | C |
| 9 | 2 | V002 | 16.04 |
| 10 | 3 | V000 | 1.04 |
| 11 | 3 | V001 | B |
| 12 | 3 | V002 | 16.02 |

MODEL:

| | ROW_INDEX | PART_INDEX | MODEL_CONTENT |
|---|---|---|---|
| 1 | 0 | ? | {"Algorithm":"KMEANS","Category":[{"CategoryCol":2,"CategoryName":["A","B","C","D"],"... |
| 2 | -1 | ? | CLUSTER_ASSIGNMENT_PIVOT |
| 3 | -2 | ? | {"feature":["V000","V001","V002"],"featureNum":3,"index":[0,1,2],"type":["CONTINUOUS","... |

STATISTICS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | Center 0 Slight Silhouette | 0.9456624776021446 |
| 2 | Center 1 Slight Silhouette | 0.9407096962254291 |
| 3 | Center 2 Slight Silhouette | 0.9432890533974454 |
| 4 | Center 3 Slight Silhouette | 0.9455368708884853 |
| 5 | SlightSilhouette | 0.943799524528376 |

## _SYS_AFL.PAL_UNIFIED_CLUSTERING_ASSIGNMENT_PIVOT

Clustering assignment pivot is a unified interface to call a cluster assignment algorithm to assign data to clusters that are previously generated by some clustering methods, including K-Means, Accelerated K-Means, K-Medians, K- Medoids, DBSCAN, SOM, and GMM.

### Overviews of ALL Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<MODEL table>` | IN |
| 3 | `<PARAMETER table>` | IN |
| 4 | `<ASSIGNMENT table>` | OUT |
| 5 | `<PLACE_HOLDER_1 table>` | OUT |

### Input Tables

| Table | Column | Reference for Output Table | Data Type | Description | Constraint |
|---|---|---|---|---|---|
| DATA | 1st column | DATA_ID | BIGINT, INTE-GER, VARCHAR, NVARCHAR | Data ID. | This must be the first column. |
| | 2nd column | VARIABLE_NAME | VARCHAR or NVARCHAR | The variable name. | Should be identical to the model. |

| Table | Column | Reference for Output Table | Data Type | Description | Constraint |
|---|---|---|---|---|---|
| | 3rd column | VALUE | VARCHAR or NVARCHAR | The variable value. | |
| MODEL | 1st column | ROW_INDEX | INTEGER | Row index. | This must be the first column. |
| | 2nd column | PART_INDEX | INTEGER | CLUSTER_ID, for GMM only. | |
| | 3rd column | MODEL_CONTENT | NCLOB | Model content. | |

### Parameter Tables

Mandatory Parameters

None.

Optional Parameters

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| THREAD_RATIO | DOUBLE | 0.0 | The ratio of available threads for multi-thread tasks:<br>• 0: single thread;<br>• 0~1: percentage.<br>Other values are ignored and this function heuristically determines the number of threads to use. | |

### Output Table

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| ASSIGNMENT | 1st column | DATA_ID (in input table) data type | DATA_ID (in input table) column name | Data ID |
| | 2nd column | INTEGER | CLUSTER_ID | The assigned cluster ID. |

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| | 3rd column | DOUBLE | DISTANCE | Distance between a given point and the<br>• cluster center (k-means)<br>• nearest core object (DBSCAN)<br>• weight vector (SOM)<br>Or probability of a given point belonging to the corresponding cluster (GMM). |
| PLACE_HOLDER_1 | 1st column | VARCHAR or NVARCHAR | OBJECT | Placeholder for future features. |
| | 2nd column | VARCHAR or NVARCHAR | KEY | |
| | 3rd column | VARCHAR or NVARCHAR | VALUE | |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

Example: DBSCAN

```
SET SCHEMA DM_PAL;
DROP TABLE META_DATA_TABLE;
CREATE COLUMN TABLE META_DATA_TABLE (
"NAME" VARCHAR (100),
"VALUE" VARCHAR (100)
);
INSERT INTO META_DATA_TABLE VALUES ('V1', 'CONTINUOUS');
INSERT INTO META_DATA_TABLE VALUES ('V2', 'CONTINUOUS');
INSERT INTO META_DATA_TABLE VALUES ('V3', 'CATEGORICAL');
DROP TABLE MY_DATA_TABLE;
CREATE COLUMN TABLE MY_DATA_TABLE (
"DATA_ID" INTEGER,
"NAME" VARCHAR (100),
"VALUE" VARCHAR (100),
"PURPOSE" INTEGER
);
INSERT INTO MY_DATA_TABLE VALUES (1, 'V1','0.10',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 'V2','0.10',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 'V3','B',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 'V1','0.11',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 'V2','0.10',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 'V3','A',1);
INSERT INTO MY_DATA_TABLE VALUES (3, 'V1','0.10',1);
INSERT INTO MY_DATA_TABLE VALUES (3, 'V2','0.11',1);
INSERT INTO MY_DATA_TABLE VALUES (3, 'V3','C',1);
INSERT INTO MY_DATA_TABLE VALUES (4, 'V1','0.11',1);
```

```
INSERT INTO MY_DATA_TABLE VALUES (4, 'V2','0.11',1);
INSERT INTO MY_DATA_TABLE VALUES (4, 'V3','B',1);
INSERT INTO MY_DATA_TABLE VALUES (5, 'V1','0.12',1);
INSERT INTO MY_DATA_TABLE VALUES (5, 'V2','0.11',1);
INSERT INTO MY_DATA_TABLE VALUES (5, 'V3','A',1);
INSERT INTO MY_DATA_TABLE VALUES (6, 'V1','0.11',1);
INSERT INTO MY_DATA_TABLE VALUES (6, 'V2','0.12',1);
INSERT INTO MY_DATA_TABLE VALUES (6, 'V3','E',1);
INSERT INTO MY_DATA_TABLE VALUES (7, 'V1','0.12',1);
INSERT INTO MY_DATA_TABLE VALUES (7, 'V2','0.12',1);
INSERT INTO MY_DATA_TABLE VALUES (7, 'V3','A',1);
INSERT INTO MY_DATA_TABLE VALUES (8, 'V1','0.12',1);
INSERT INTO MY_DATA_TABLE VALUES (8, 'V2','0.13',1);
INSERT INTO MY_DATA_TABLE VALUES (8, 'V3','C',1);
INSERT INTO MY_DATA_TABLE VALUES (9, 'V1','0.13',1);
INSERT INTO MY_DATA_TABLE VALUES (9, 'V2','0.12',1);
INSERT INTO MY_DATA_TABLE VALUES (9, 'V3','D',1);
INSERT INTO MY_DATA_TABLE VALUES (10, 'V1','0.13',1);
INSERT INTO MY_DATA_TABLE VALUES (10, 'V2','0.13',1);
INSERT INTO MY_DATA_TABLE VALUES (10, 'V3','D',1);
INSERT INTO MY_DATA_TABLE VALUES (11, 'V1','0.13',1);
INSERT INTO MY_DATA_TABLE VALUES (11, 'V2','0.14',1);
INSERT INTO MY_DATA_TABLE VALUES (11, 'V3','A',1);
INSERT INTO MY_DATA_TABLE VALUES (12, 'V1','0.14',1);
INSERT INTO MY_DATA_TABLE VALUES (12, 'V2','0.13',1);
INSERT INTO MY_DATA_TABLE VALUES (12, 'V3','C',1);
INSERT INTO MY_DATA_TABLE VALUES (13, 'V1','10.10',1);
INSERT INTO MY_DATA_TABLE VALUES (13, 'V2','10.10',1);
INSERT INTO MY_DATA_TABLE VALUES (13, 'V3','A',1);
INSERT INTO MY_DATA_TABLE VALUES (14, 'V1','10.11',1);
INSERT INTO MY_DATA_TABLE VALUES (14, 'V2','10.10',1);
INSERT INTO MY_DATA_TABLE VALUES (14, 'V3','F',1);
INSERT INTO MY_DATA_TABLE VALUES (15, 'V1','10.10',1);
INSERT INTO MY_DATA_TABLE VALUES (15, 'V2','10.11',1);
INSERT INTO MY_DATA_TABLE VALUES (15, 'V3','E',1);
INSERT INTO MY_DATA_TABLE VALUES (16, 'V1','10.11',1);
INSERT INTO MY_DATA_TABLE VALUES (16, 'V2','10.11',1);
INSERT INTO MY_DATA_TABLE VALUES (16, 'V3','E',1);
INSERT INTO MY_DATA_TABLE VALUES (17, 'V1','10.11',1);
INSERT INTO MY_DATA_TABLE VALUES (17, 'V2','10.12',1);
INSERT INTO MY_DATA_TABLE VALUES (17, 'V3','A',1);
INSERT INTO MY_DATA_TABLE VALUES (18, 'V1','10.12',1);
INSERT INTO MY_DATA_TABLE VALUES (18, 'V2','10.11',1);
INSERT INTO MY_DATA_TABLE VALUES (18, 'V3','B',1);
INSERT INTO MY_DATA_TABLE VALUES (19, 'V1','10.12',1);
INSERT INTO MY_DATA_TABLE VALUES (19, 'V2','10.12',1);
INSERT INTO MY_DATA_TABLE VALUES (19, 'V3','B',1);
INSERT INTO MY_DATA_TABLE VALUES (20, 'V1','10.12',1);
INSERT INTO MY_DATA_TABLE VALUES (20, 'V2','10.13',1);
INSERT INTO MY_DATA_TABLE VALUES (20, 'V3','D',1);
INSERT INTO MY_DATA_TABLE VALUES (21, 'V1','10.13',1);
INSERT INTO MY_DATA_TABLE VALUES (21, 'V2','10.12',1);
INSERT INTO MY_DATA_TABLE VALUES (21, 'V3','F',1);
INSERT INTO MY_DATA_TABLE VALUES (22, 'V1','10.13',1);
INSERT INTO MY_DATA_TABLE VALUES (22, 'V2','10.13',1);
INSERT INTO MY_DATA_TABLE VALUES (22, 'V3','F',1);
INSERT INTO MY_DATA_TABLE VALUES (23, 'V1','10.13',1);
INSERT INTO MY_DATA_TABLE VALUES (23, 'V2','10.14',1);
INSERT INTO MY_DATA_TABLE VALUES (23, 'V3','A',1);
INSERT INTO MY_DATA_TABLE VALUES (24, 'V1','10.14',1);
INSERT INTO MY_DATA_TABLE VALUES (24, 'V2','10.13',1);
INSERT INTO MY_DATA_TABLE VALUES (24, 'V3','D',1);
INSERT INTO MY_DATA_TABLE VALUES (25, 'V1','4.10',1);
INSERT INTO MY_DATA_TABLE VALUES (25, 'V2','4.10',1);
INSERT INTO MY_DATA_TABLE VALUES (25, 'V3','A',1);
INSERT INTO MY_DATA_TABLE VALUES (26, 'V1','7.11',1);
INSERT INTO MY_DATA_TABLE VALUES (26, 'V2','7.10',1);
INSERT INTO MY_DATA_TABLE VALUES (26, 'V3','C',1);
```

```
INSERT INTO MY_DATA_TABLE VALUES (27, 'V1','-3.10',1);
INSERT INTO MY_DATA_TABLE VALUES (27, 'V2','-3.11',1);
INSERT INTO MY_DATA_TABLE VALUES (27, 'V3','C',1);
INSERT INTO MY_DATA_TABLE VALUES (28, 'V1','16.11',1);
INSERT INTO MY_DATA_TABLE VALUES (28, 'V2','16.11',1);
INSERT INTO MY_DATA_TABLE VALUES (28, 'V3','A',1);
INSERT INTO MY_DATA_TABLE VALUES (29, 'V1','20.11',1);
INSERT INTO MY_DATA_TABLE VALUES (29, 'V2','20.12',1);
INSERT INTO MY_DATA_TABLE VALUES (29, 'V3','C',1);
INSERT INTO MY_DATA_TABLE VALUES (30, 'V1','15.12',1);
INSERT INTO MY_DATA_TABLE VALUES (30, 'V2','15.11',1);
INSERT INTO MY_DATA_TABLE VALUES (30, 'V3','A',1);
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL (
"NAME" VARCHAR (50),
"INT_VALUE" INTEGER,
"DOUBLE_VALUE" DOUBLE,
"STRING_VALUE" VARCHAR (100)
);
INSERT INTO PAL_PARAMETER_TBL VALUES ('FUNCTION', NULL, NULL, 'DBSCAN');
INSERT INTO PAL_PARAMETER_TBL VALUES ('AUTO_PARAM', NULL, NULL, 'true');
INSERT INTO PAL_PARAMETER_TBL VALUES ('METRIC', 1, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 1.0, NULL);
DROP TABLE PAL_CLUSTERING_MODEL_TBL;
CREATE COLUMN TABLE PAL_CLUSTERING_MODEL_TBL (
"ROW_INDEX" INTEGER,
"PART_INDEX" INTEGER,
"MODEL_CONTENT" VARCHAR (5000)
);
DO BEGIN
lt_mate_data = SELECT * FROM META_DATA_TABLE;
lt_data = SELECT * FROM MY_DATA_TABLE;
lt_param = SELECT * FROM PAL_PARAMETER_TBL;
CALL
"_SYS_AFL"."PAL_UNIFIED_CLUSTERING_PIVOT"(:lt_mate_data, :lt_data, :lt_param,
lt_result, lt_centers, lt_model, lt_stat, lt_optimal_parameter, lt_place1,
lt_place2);
INSERT INTO PAL_CLUSTERING_MODEL_TBL SELECT * FROM :lt_model;
END;
DROP TABLE MY_DATA_ASSIGNMENT_TABLE;
CREATE COLUMN TABLE MY_DATA_ASSIGNMENT_TABLE (
"DATA_ID" INTEGER,
"NAME" VARCHAR (100),
"VALUE" VARCHAR (100)
);
INSERT INTO MY_DATA_ASSIGNMENT_TABLE VALUES (1, 'V1','0.10');
INSERT INTO MY_DATA_ASSIGNMENT_TABLE VALUES (1, 'V2','0.10');
INSERT INTO MY_DATA_ASSIGNMENT_TABLE VALUES (1, 'V3','B');
INSERT INTO MY_DATA_ASSIGNMENT_TABLE VALUES (2, 'V1','0.10');
INSERT INTO MY_DATA_ASSIGNMENT_TABLE VALUES (2, 'V2','-2.10');
INSERT INTO MY_DATA_ASSIGNMENT_TABLE VALUES (2, 'V3','A');
INSERT INTO MY_DATA_ASSIGNMENT_TABLE VALUES (3, 'V1','3.10');
INSERT INTO MY_DATA_ASSIGNMENT_TABLE VALUES (3, 'V2','0.10');
INSERT INTO MY_DATA_ASSIGNMENT_TABLE VALUES (3, 'V3','B');
INSERT INTO MY_DATA_ASSIGNMENT_TABLE VALUES (4, 'V1','10.10');
INSERT INTO MY_DATA_ASSIGNMENT_TABLE VALUES (4, 'V2','10.10');
INSERT INTO MY_DATA_ASSIGNMENT_TABLE VALUES (4, 'V3','D');
INSERT INTO MY_DATA_ASSIGNMENT_TABLE VALUES (5, 'V1','3.10');
INSERT INTO MY_DATA_ASSIGNMENT_TABLE VALUES (5, 'V2','-0.50');
INSERT INTO MY_DATA_ASSIGNMENT_TABLE VALUES (5, 'V3','C');
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL (
"NAME" VARCHAR (50),
"INT_VALUE" INTEGER,
"DOUBLE_VALUE" DOUBLE,
"STRING_VALUE" VARCHAR (100)
);
```

```
CALL
_SYS_AFL.PAL_UNIFIED_CLUSTERING_ASSIGNMENT_PIVOT(MY_DATA_ASSIGNMENT_TABLE,PAL_CLU
STERING_MODEL_TBL, PAL_PARAMETER_TBL, ?,?);
```

**Expected Result**

ASSIGNMENT:

| | DATA_ID | CLUSTER_ID | DISTANCE |
|---|---|---|---|
| 1 | 1 | 0 | 0 |
| 2 | 2 | 0 | 2.21 |
| 3 | 3 | -1 | 3 |
| 4 | 4 | 1 | 0.050000... |
| 5 | 5 | -1 | 3.59 |

## 5.13.6.1 Parameters Overview

The following table shows mapping parameters between unified cluster procedures and original procedures.

| Algorithm | new name | Origin name |
|---|---|---|
| AHC | LINKAGE | CLUSTER_METHOD |
| | N_CLUSTERS | CLUSTER_NUM |
| | DISTANCE_LEVEL | DISTANCE_FUNC |
| | NORMALIZATION | NORMALIZE_TYPE |
| DBSCAN | DISTANCE_LEVEL | DISTANCE_METHOD |
| | INIT_PARAM | INIT_MODE |
| | ALGORITHM | METHOD |
| | MINKOWSKI_POWER | MINKOW_P |
| | EPS | RADIUS |
| GMM | MAX_ITER | MAX_ITERATION |
| KMEDIANS | TOL | EXIT_THRESHOLD |
| | N_CLUSTERS | GROUP_NUMBER |
| | INIT | INIT_TYPE |
| | MAX_ITER | MAX_ITERATION |
| KMEDOIDS | TOL | EXIT_THRESHOLD |
| | N_CLUSTERS | GROUP_NUMBER |
| | INIT | INIT_TYPE |
| | MAX_ITER | MAX_ITERATION |
| KMEANS | TOL | EXIT_THRESHOLD |

| Algorithm | new name | Origin name |
|---|---|---|
| | N_CLUSTERS | GROUP_NUMBER |
| | N_CLUSTERS_MAX | GROUP_NUMBER_MAX |
| | N_CLUSTERS_MIN | GROUP_NUMBER_MIN |
| | INIT | INIT_TYPE |
| | MAX_ITER | MAX_ITERATION |
| AKMEANS | TOL | EXIT_THRESHOLD |
| | N_CLUSTERS | GROUP_NUMBER |
| | N_CLUSTERS_MAX | GROUP_NUMBER_MAX |
| | N_CLUSTERS_MIN | GROUP_NUMBER_MIN |
| | INIT | INIT_TYPE |
| | MAX_ITER | MAX_ITERATION |
| SOM | MAX_ITER | MAX_ITERATION |
| AP | DISTANCE_LEVEL | DISTANCE_METHOD |
| | MAX_ITER | MAX_ITERATION |
| SP | N_CLUSTERS | GROUP_NUMBER |
| | MAX_ITER | MAX_ITERATION |
| | INIT | INIT_TYPE |
| | SPLIT_THRESHOLD | GAMMA |
| | TOL | EXIT_THRESHOLD |

## 5.13.6.2 Parameter Tables

This topic lists all parameters for unified clustering pivot.

Mandatory Parameters

None

Optional Parameters

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| PRECALCU-LATED_DISTANCE | INTEGER | 0 | State of pre-computed distance matrix as input data:<br>• 0: invalid<br>• 1: valid | |

## AHC

Mandatory Parameters

None

Optional Parameters

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| LINKAGE | INTEGER | 5 | Linkage type between two clusters.<br>• 1: nearest neighbor (single linkage)<br>• 2: furthest neighbor (complete linkage)<br>• 3: group average (UPGMA)<br>• 4: weighted average (WPGMA)<br>• 5: centroid clustering<br>• 6: median clustering<br>• 7: ward method<br>**Note**<br>For clustering method 5, 6, or 7, set the corresponding distance function (DISTANCE_LEVEL) to 8. | |
| N_CLUSTERS | INTEGER | 1 | Number of clusters after agglomerate hierarchical clustering algorithm.<br>Value range: between 1 and the initial number of input data. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| DISTANCE_DIMENSION | DOUBLE | 3 | Distance dimension can be set if DISTANCE_LEVEL is set to 3 (Minkowski Distance).<br>The value should be no less than 1. | Only valid when DISTANCE_LEVEL is 3. |
| DISTANCE_LEVEL | INTEGER | 8 | Measure of distance between two clusters.<br>• 1: Manhattan Distance<br>• 2: Euclidean Distance<br>• 3: Minkowski Distance<br>• 4: Chebyshev Distance<br>• 6: Cosine<br>• 7: Pearson Correlation<br>• 8: Squared Euclidean Distance<br>• 9: Jaccard Distance<br>• 10: Gower Distance<br>**Note**<br>• For Jaccard Distance, non-zero input data will be treated as 1, and zero input data will be treated as 0.<br>Jaccard Distance = $(M_{01} + M_{10}) / (M_{11} + M_{01} + M_{10})$<br>• Only Gower Distance supports category attributes. | When LINKAGE is 5, 6, or 7, this parameter must be set to 8. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| NORMALIZATION | INTEGER | 0 | Normalization type:<br>• 0: does nothing<br>• 1: Z score standardize<br>• 2: transforms to new range: -1 to 1<br>• 3: transforms to new range: 0 to 1 | |
| THREAD_RATIO | DOUBLE | -1.0 | The ratio of available threads for multi-thread tasks.<br>• 0: single thread<br>• 0–1: uses the specified percentage of available threads. PAL uses all available threads if the number is 1.0.<br>• -1: uses all of available threads. | |

## DBSCAN

Mandatory Parameters

| Name | Data Type | Description | Dependency |
|---|---|---|---|
| AUTO_PARAM | VARCHAR or NVARCHAR | Specifies whether the MINPTS and EPS parameters are determined automatically or by user input.<br>• true: automatically determines the parameters.<br>• false: uses parameter values provided by you. | |
| MINPTS | INTEGER | Specifies the minimum number of points required to form a cluster. | Only valid when AUTO_PARAM is false. |
| EPS | Double | Specifies the scan radius (eps). | Only valid when AUTO_PARAM is false. |

Optional Parameters

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| CATEGORY_WEIGHTS | DOUBLE | 0.707 | Represents the weight of categorical variable (γ). | |
| DISTANCE_LEVEL | INTEGER | 2 | Specifies the method to compute the distance between two points.<br>• 1: Manhattan<br>• 2: Euclidean<br>• 3: Minkowski<br>• 4: Chebyshev<br>• 5: Standardized Euclidean<br>• 6: Cosine | |
| ALGORITHM | INTEGER | 1 | Searching method:<br>• 0: Brute force searching<br>• 1: KD-tree searching | |
| MINKOWSKI_POWER | INTEGER | 3 | Specifies the power of the Minkowski method. | Only valid when DISTANCE_LEVEL is 3. |
| SAVE_MODEL | INTEGER | 1 | Indicates whether model should be saved:<br>• 0: not save<br>• 1: save | |
| STRING_VARIABLE | NVARCHAR | No default value | Indicates a string column storing not categorical data. Levenshtein distance is used to calculate similarity between two strings. Ignored if it is not a string column. | If DISTANCE_LEVEL is set to 6 (COSINE), this parameter is not supported. |
| VARIABLE_WEIGHT | NVARCHAR for variable name and DOUBLE for weight value | No default value | Specifies the weight of a variable participating in distance calculation.<br>The value must be equal to or greater than 0. Defaults to 1 for variables not specified. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| THREAD_RATIO | DOUBLE | -1.0 | The ratio of available threads for multi-thread tasks:<br>• 0: single thread<br>• 0–1: uses the specified percentage of available threads. PAL uses all available threads if the number is 1.0.<br>• -1: uses all of available threads. | |

## GMM

Mandatory Parameters

| Name | Data Type | Description | Dependency |
| --- | --- | --- | --- |
| INIT_MODE | INTEGER | Specifies the initialization mode.<br>• 0: sets the number of the Gaussian distributions of GMM in the initialize parameter table. The initial centers will be given by the farthest-first traversal algorithm.<br>• 1: specifies the ID of the initial centers in the initialize parameter table. The number of the components in GMM will be equal to the number of different initial centers that are given.<br>• 2: sets the number of the Gaussian distributions of GMM in the initialize parameter table. The responsibilities matrix (r_ik) is randomly initialized, which means the initial centers are the means of all the data that are randomly weighted.<br>• 3: sets the number of the Gaussian distributions of GMM in the initialize parameter table. The initial centers are given using the k-means++ approach. | |

| Name | Data Type | Description | Dependency |
| --- | --- | --- | --- |
| INITIALIZE_PARAMETER | INTEGER, VARCHAR, or NVARCHAR | Specifies the centers of data or the cluster number;<br>• 2nd column: number of data;<br>• 4th column: specifies the cluster number or the center of data. | When INIT_MODE is set to 0, 2, or 3, then this value is the number of clusters in GMM. In this case, only the first value of INITIALIZE_PARAMETER is used.<br>When INIT_MODE is set to 1, then values of INITIALIZE_PARAMETER are the "IDs" of the specified initial centers in the data table. Each value represents an initial center in the data table. The number of clusters is equal to the number of the given initial centers. For example, setting initial centers 1, 2, 10 means selecting the rows in the input data table whose value of the ID column is 1, 2, or 10, and the number of clusters will be 3. |

Optional Parameters

| Name | Data Type | Default Value | Description |
| --- | --- | --- | --- |
| CATEGORY_WEIGHT | DOUBLE | 0.707 | Represents the weight of category attributes ($\gamma$). |
| COVARIANCE_TYPE | INTEGER | 0 | Indicates the type of covariance matrices in the model:<br>• 0: uses full covariance matrices.<br>• 1: uses diagonal covariance matrices.<br>• 2: uses diagonal covariance matrices with all equal diagonal entries. |
| ERROR_TOL | DOUBLE | 1.00E-05 | Specifies the error tolerance, which is the stop condition. |
| MAX_ITER | INTEGER | 100 | Specifies the maximum iteration number of the EM algorithm. |

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| REGULARIZATION | DOUBLE | 1.00E-06 | Regularization to be added to the diagonal of covariance matrices to ensure positive-definite. |
| SEED | INTEGER | 0 | Specifies the seed for random number generator:<br>• 0: uses the current time as the seed.<br>• Others: uses the specified value as the seed. |
| SHARED_COVARIANCE | INTEGER | 0 | Indicates whether all the clusters share the same covariance matrix:<br>• 0: clusters have different covariance matrices.<br>• 1: clusters share a same covariance matrix. |
| THREAD_RATIO | DOUBLE | -1.0 | The ratio of available threads for multi-thread tasks:<br>• 0: single thread<br>• 0–1: uses the specified percentage of available threads. PAL uses all available threads if the number is 1.0.<br>• -1: uses all of available threads. |

## SOM

Mandatory Parameters

None

Optional Parameters

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| ALPHA | DOUBLE | 0.5 | Specifies the learning rate. |

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| BATCH_SOM | INTEGER | 0 | Indicates whether batch SOM is carried out:<br>• 0: classical SOM<br>• 1: batch SOM<br>For batch SOM, KERNEL_FUNCTION is always Gaussian, and the LEARNING_RATE factors take no effect. |
| CONVERGENCE_CRITERION | DOUBLE | 1.00E-06 | If the largest difference of the successive maps is less than this value, the calculation is regarded as convergence, and SOM is completed consequently. |
| HEIGHT_OF_MAP | INTEGER | 10 | Indicates the height of the map. |
| KERNEL_FUNCTION | INTEGER | 1 | Represents the neighborhood kernel function:<br>• 1: Gaussian<br>• 2: Bubble/Flat |
| LEARNING_RATE | INTEGER | 1 | Indicates the decay function for learning rate:<br>• 1: exponential<br>• 2: linear |
| MAX_ITER | INTEGER | 1000 plus 500 times the number of neurons in the lattice | Maximum number of iterations.<br>**Note**<br>The training might not converge if this value is too small, for example, less than 1000. |
| NORMALIZATION | INTEGER | 0 | Normalization type:<br>• 0: no<br>• 1: transform to new range (0.0, 1.0).<br>• 2: Z-score normalization |
| RADIUS | DOUBLE | The bigger value of HEIGHT_OF_MAP and WIDTH_OF_MAP | Specifies the scan radius. |

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| RANDOM_SEED | INTEGER | -1 | • -1: random<br>• 0: sets every weight to zero.<br>• Other value: uses this value as seed. |
| SHAPE_OF_GRID | INTEGER | 2 | Indicates the shape of the grid:<br>• 1: rectangle<br>• 2: hexagon |
| WIDTH_OF_MAP | INTEGER | 10 | Indicates the width of the map. |

## Accelerated K-Means

Mandatory Parameters

None

Optional Parameters

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| CATEGORY_WEIGHTS | DOUBLE | 0.707 | Represents the weight of category attributes (γ). | |
| DISTANCE_LEVEL | INTEGER | 2 | Computes the distance between the item and the cluster center:<br>• 1: Manhattan distance<br>• 2: Euclidean distance<br>• 3: Minkowski distance<br>• 4: Chebyshev distance<br>• 6: Cosine distance. | |
| TOL | DOUBLE | 1.00E-06 | Threshold (actual value) for exiting the iterations. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| N_CLUSTERS | INTEGER | No default value | Number of groups (k).<br>**Note**<br>If this parameter is not specified, you must specify the range of k using N_CLUSTERS_MIN and N_CLUSTERS_MAX. Then the algorithm will iterate through the range and return the k with the highest slight silhouette. | |
| N_CLUSTERS_MAX | INTEGER | No default value | Upper boundary of the range that k falls in.<br>**Note**<br>You must specify either an exact value or a range for k. If both are specified, the exact value will be used. | |
| N_CLUSTERS_MIN | INTEGER | No default value | Lower boundary of the range that k falls in.<br>**Note**<br>You must specify either an exact value or a range for k. If both are specified, the exact value will be used. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| INIT | INTEGER | 4 | Controls how the initial centers are selected:<br>• 1: first k observations<br>• 2: random with replacement<br>• 3: random without replacement<br>• 4: patent of selecting the init center (US 6,882,998 B1) | |
| MAX_ITER | INTEGER | 100 | Maximum iterations. | |
| MINKOWSKI_POWER | DOUBLE | 3.0 | When you use the Minkowski distance, this parameter controls the value of power. | Only valid when DISTANCE_LEVEL is 3. |
| NORMALIZATION | INTEGER | 0 | Normalization type:<br>• 0: no<br>• 1: yes. For each point `X (x1,x2,...,xn)`, the normalized value will be `X'(x1/S,x2/S,...,xn/S)`, where `S = \|x1\|+\|x2\|+...\|xn\|`.<br>• 2: for each column `C`, get the min and max value of `C`, and then `C[i] = (C[i]-min)/(max-min)`. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| THREAD_RATIO | DOUBLE | -1.0 | The ratio of available threads for multi-thread tasks:<br>• 0: single thread<br>• 0–1: uses the specified percentage of available threads. PAL uses all available threads if the number is 1.0.<br>• -1: uses all of available threads. | |

## K-Means

Mandatory Parameters

None

Optional Parameters

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| CATEGORY_WEIGHTS | DOUBLE | 0.707 | Represents the weight of category attributes (γ). | |
| DISTANCE_LEVEL | INTEGER | 2 | Computes the distance between the item and the cluster center:<br>• 1: Manhattan distance<br>• 2: Euclidean distance<br>• 3: Minkowski distance<br>• 4: Chebyshev distance<br>• 6: Cosine distance | |
| TOL | DOUBLE | 1.00E-06 | Threshold (actual value) for exiting the iterations. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| N_CLUSTERS | INTEGER | No default value | Number of groups (k).<br>**Note**<br>If this parameter is not specified, you must specify the range of k using N_CLUSTERS_MIN and N_CLUSTERS_MAX. Then the algorithm will iterate through the range and return the k with the highest slight silhouette. | |
| N_CLUSTERS_MAX | INTEGER | No default value | Upper boundary of the range that k falls in.<br>**Note**<br>You must specify either an exact value or a range for k. If both are specified, the exact value will be used. | |
| N_CLUSTERS_MIN | INTEGER | No default value | Lower boundary of the range that k falls in.<br>**Note**<br>You must specify either an exact value or a range for k. If both are specified, the exact value will be used. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| INIT | INTEGER | 4 | Controls how the initial centers are selected:<br>• 1: first k observations<br>• 2: random with replacement<br>• 3: random without replacement<br>• 4: patent of selecting the init center (US 6,882,998 B1) | |
| MAX_ITER | INTEGER | 100 | Maximum iterations. | |
| MINKOWSKI_POWER | DOUBLE | 3.0 | When you use the Minkowski distance, this parameter controls the value of power. | Only valid when DISTANCE_LEVEL is 3. |
| NORMALIZATION | INTEGER | 0 | Normalization type:<br>• 0: no<br>• 1: yes. For each point X (x1,x2,...,xn), the normalized value will be X'(x1/S,x2/S,...,xn/S), where S = \|x1\|+\|x2\|+...\|xn\|.<br>• 2: for each column C, get the min and max value of C, and then C[i] = (C[i]-min)/(max-min). | |
| THREAD_RATIO | DOUBLE | -1.0 | The ratio of available threads for multi-thread tasks:<br>• 0: single thread<br>• 0–1: uses the specified percentage of available threads. PAL uses all available threads if the number is 1.0.<br>• -1: uses all of available threads. | |

## K-Medians

Mandatory Parameters

| Name | Data Type | Description |
|---|---|---|
| N_CLUSTERS | INTEGER | Number of groups (k). The value range is from 2 to the number of training records. |

Optional Parameters

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| CATEGORY_WEIGHTS | DOUBLE | 0.707 | Represents the weight of category attributes (γ). | |
| DISTANCE_LEVEL | INTEGER | 2 | Computes the distance between the item and the cluster center:<br>• 1: Manhattan distance<br>• 2: Euclidean distance<br>• 3: Minkowski distance<br>• 4: Chebyshev distance<br>• 6: Cosine distance | |
| TOL | DOUBLE | 1.00E-06 | Threshold (actual value) for exiting the iterations. | |
| INIT | INTEGER | 4 | Controls how the initial centers are selected:<br>• 1: first k observations<br>• 2: random with replacement<br>• 3: random without replacement<br>• 4: patent of selecting the initial center (US 6,882,998 B1) | |
| MAX_ITER | INTEGER | 100 | Maximum iterations. | |
| MINKOWSKI_POWER | DOUBLE | 3.0 | When you use the Minkowski distance, this parameter controls the value of power. | Only valid when DISTANCE_LEVEL is 3. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| NORMALIZATION | INTEGER | 0 | Normalization type:<br>• 0: no<br>• 1: yes. For each point X (x1,x2,...,xn), the normalized value will be X'(x1/S,x2/S,...,xn/S), where S = \|x1\|+\|x2\|+...\|xn\|.<br>• 2: for each column C, get the min and max value of C, and then C[i] = (C[i]-min)/(max-min). | |
| THREAD_RATIO | DOUBLE | -1.0 | The ratio of available threads for multi-thread tasks:<br>• 0: single thread<br>• 0–1: uses the specified percentage of available threads. PAL uses all available threads if the number is 1.0.<br>• -1: uses all of available threads. | |

## K-Medoids

Mandatory Parameters

| Name | Data Type | Description |
|---|---|---|
| N_CLUSTERS | INTEGER | Number of groups (k). The value range is from 2 to the number of training records. |

Optional Parameters

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| CATEGORY_WEIGHTS | DOUBLE | 0.707 | Represents the weight of category attributes (γ). | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| DISTANCE_LEVEL | INTEGER | 2 | Computes the distance between the item and the cluster center:<br>• 1: Manhattan distance<br>• 2: Euclidean distance<br>• 3: Minkowski distance<br>• 4: Chebyshev distance<br>• 5: Jaccard distance<br>• 6: Cosine distance | When Jaccard distance is used, all columns must be categorical. |
| TOL | DOUBLE | 1.00E-06 | Threshold (actual value) for exiting the iterations. | |
| INIT | INTEGER | 4 | Controls how the initial centers are selected:<br>• 1: first k observations<br>• 2: random with replacement<br>• 3: random without replacement<br>• 4: patent of selecting the init center (US 6,882,998 B1) | |
| MAX_ITER | INTEGER | 100 | Maximum iterations. | |
| MINKOWSKI_POWER | DOUBLE | 3.0 | When you use the Minkowski distance, this parameter controls the value of power. | Only valid when DISTANCE_LEVEL is 3. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| NORMALIZATION | INTEGER | 0 | Normalization type:<br>• 0: no<br>• 1: yes. For each point X (x1,x2,...,xn), the normalized value will be X'(x1/S,x2/S,...,xn/S), where S = \|x1\|+\|x2\|+...\|xn\|.<br>• 2: for each column C, get the min and max value of C, and then C[i] = (C[i]-min)/(max-min). | |
| THREAD_RATIO | DOUBLE | -1.0 | The ratio of available threads for multi-thread tasks:<br>• 0: single thread<br>• 0–1: uses the specified percentage of available threads. PAL uses all available threads if the number is 1.0.<br>• -1: uses all of available threads. | |

## AP

Mandatory Parameters

| Name | Data Type | Description |
|---|---|---|
| DISTANCE_LEVEL | INTEGER | Specifies the method to compute the distance between two points.<br>• 1: Manhattan<br>• 2: Euclidean<br>• 3: Minkowski<br>• 4: Chebyshev<br>• 5: Standardized Euclidean<br>• 6: Cosine |

| Name | Data Type | Description |
| --- | --- | --- |
| CLUSTER_NUMBER | INTEGER | Number of cluster.<br>• 0: does not adjust AP Cluster result;<br>• Non-zero INTEGER: if AP cluster number is greater than CLUSTER_NUMBER, PAL will merge the result to set the cluster number to the value specified for CLUSTER_NUMBER. |

Optional Parameters

| Name | Data Type | Default Value | Description | Dependency |
| --- | --- | --- | --- | --- |
| INITIALIZE_SEED | NVARCHAR | No default value | Specifies the record id. | |
| MAX_ITER | INTEGER | 500 | Maximum number of iterations. | |
| CON_INTERATION | INTEGER | 100 | When the clusters keep a steady one for the specified times, the algorithm ends. | |
| DAMP | DOUBLE | 0.9 | Controls the updating velocity.<br>Value range: 0 < DAMP < 1 | |
| PREFERENCE | DOUBLE | 0.5 | Determines the preference.<br>Value range: 0 <= PREFERENCE <= 1 | |
| SEED_RATION | DOUBLE | 1 | Select a portion of (SEED_RATION * data_number) the input data as seed, where data_number is the row_size of the input data.<br>Value range: 0< SEED_RATION <= 1 | Only valid when the INITIALIZE_SEED is not define. If SEED_RATIO is 1, all the input data will be the seed. |
| TIMES | INTEGER | 1 | The sampling times. | Only valid when the INITIALIZE_SEED is not defined and SEED_RATIO is less than 1. |
| MINKOW_P | INTEGER | 3 | The power of the Minkowski method. | Only valid when DISTANCE_LEVEL is 3. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| THREAD_RATIO | DOUBLE | -1.0 | The ratio of available threads for multi-thread tasks:<br>• 0: single thread<br>• 0–1: uses the specified percentage of available threads. PAL uses all available threads if the number is 1.0.<br>• -1: uses all of available threads. | |

## SP

Mandatory Parameters

| Name | Data Type | Description |
|---|---|---|
| N_CLUSTERS | INTEGER | Number of groups. The value range is from 2 to the number of training records. |

Optional Parameters

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| THREAD_RATIO | DOUBLE | -1.0 | Specifies the ratio of total number of threads that can be used by this function.<br>• 0: single thread<br>• 0–1: uses the specified percentage of available threads. PAL uses all available threads if the number is 1.0.<br>• -1: uses all of available threads. | |
| N_COMPONENTS | INTEGER | GROUP_NUMBER | Number of eigenvectors to use for the spectral embedding. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| SPLIT_THRESHOLD | DOUBLE | 1.0 | Kernel coefficient of RBF(radia basis function) used in constructing affinity matrix construction with distance d: exp(-gamma*d^2). | |
| AFFINITY | INTEGER | 2 | How to construct the affinity matrix: <ul><li>0: k-nearest neighbor graph</li><li>1: mutual k-nearest neighbor graph</li><li>2: full connected graph</li></ul> | |
| N_NEIGHBOURS | INTEGER | 10 | Number of neighbors to use while constructing the affinity matrix using nearest neighbors method. | Only valid when AFFINITY is 0 or 1. |
| CUT_METHOD | INTEGER | 0 | How to cut the graph:<ul><li>0: RatioCut</li><li>1: NCut</li> | |
| EIGEN_TOL | DOUBLE | 1e-10 | Stopping criterion for eigendecomposition of the Laplacian matrix. | |
| KRYLOV_DIMENSION | INTEGER | N_CONPONENTS*2 | Parameter that controls the convergence speed of the algorithm. Typically a larger KRYLOY_DIMENSION means faster convergence, but it may also result in greater memory use and more matrix operations in each iteration.<br>**Note**<br>This parameter must satifiy N_COMPONENTS< KRYLOY_DIMENSION≤ the number of training records. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| DISTANCE_LEVEL | INTEGER | 2 | Computes the distance between the item and the cluster center:<ul><li>1: Manhattan distance</li><li>2: Euclidean distance</li><li>3: Minkowski distance</li><li>4: Chebyshev distance</li><li>6: Cosine distance</li></ul> | |
| CATEGORY_WEIGHTS | DOUBLE | 0.707 | Represents the weight of category attributes. | |
| MINKOWSKI_POWER | DOUBLE | 3.0 | When you use the Minkowski distance, this parameter controls the value of power. | Only valid when DISTANCE_LEVEL is 3. |
| MAX_ITER | INTEGER | 100 | Maximum iterations for K-Means. | |
| INIT | INTEGER | 4 | INIT_TYPE for K-Means. | |
| TOL | DOUBLE | 1e-6 | Threshold (actual value) for exiting the iterations of K-Means. | |

## 5.13.6.3 Precomputed Distance Matrix as Input Data

You can use some clustering algorithms in the unified clustering with pivot by using precomputed distance matrix as pivoted input data.

To perform unified clustering with pivoted precomputed distance matrix, the input data must be either upper or lower triangular, which means that the expanded shape needs to be N samples * N samples and the distance value of the pair of (i,j) or (j,i) must be unique.

Currently, unified clustering with pivoted with pre-computed distance matrix supports K-Medoids, AHC and SP algorithms. Please note:

1. The order needs to be (left, right) or (right, left) + distance in the metadata table.
2. For SP, 0 value means the pair of points is disconnected, but for other algorithms, it means very close.

### Clustering with Pivot

#### Parameter Table

Mandatory Parameters

None.

Optional Parameters

| Name | Data Type | Defalut Value | Description | Dependency |
|---|---|---|---|---|
| THREAD_RATIO | DOUBLE | -1.0 | The ratio of available threads for multi-thread tasks:<br>• 0: single thread;<br>• 0~1: percentage;<br>• others: heuristically determines the number of threads to use. | |
| PRECALCULATED_DISTANCE | INTEGER | 0 | State of precomputed distance matrix as input data:<br>• 0: invalid;<br>• 1: valid; | |

## KMEDOIDS

Mandatory Parameters

| Name | Data Type | Description |
|---|---|---|
| N_CLUSTERS | INTEGER | Number of groups (k). The value range is from 2 to the number of training records. |

Optional Parameters

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| TOL | DOUBLE | 1.00E-06 | Threshold (actual value) for exiting the iterations. | |
| INIT | INTEGER | 1 | Controls how the initial centers are selected:<br>• 1: First k observations;<br>• 2: Random with replacement;<br>• 3: Random without replacement | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| RANDOM_SEED | INTEGER | 0 | Indicates the seed used to initialize the random number generator. It can be set to 0 or a positive value.<br>• 0: Uses the system time;<br>• Nonzero: Uses the specified seed. | |
| MAX_ITER | INTEGER | 100 | Maximum iterations. | |

### SP

Mandatory Parameters

| Name | Data Type | Description |
|---|---|---|
| N_CLUSTERS | INTEGER | Number of groups. The value range is from 2 to the number of training records. |

Optional Parameters

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| N_COMPONENTS | INTEGER | GROUP_NUMBER | Number of eigenvectors to use for the spectral embedding. | |
| CUT_METHOD | INTEGER | 0 | How to cut the graph:<ul><li>0: RatioCut</li><li>1: NCut</li> | |
| EIGEN_TOL | DOUBLE | 1e-10 | Stopping criterion for eigendecomposition of the Laplacian matrix. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| KRYLOV_DIMENSION | INTEGER | N_CONPONENTS*2 | Parameter that controls the convergence speed of the algorithm. Typically a larger KRYLOY_DIMMENSION means faster convergence, but it may also result in greater memory use and more matrix operations in each iteration.<br>**Note**<br>This parameter must satifiy N_COMPONENTS< KRYLOY_DIMENSION≤ the number of training records. | |
| MAX_ITER | INTEGER | 100 | Maximum iterations for K-Means. | |
| INIT | INTEGER | 4 | INIT_TYPE for K-Means. | |
| TOL | DOUBLE | 1e-6 | Threshold (actual value) for exiting the iterations of K-Means. | |

### AHC

Mandatory Parameters

None.

Optional Parameters

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| N_CLUSTERS | INTEGER | 1 | Number of clusters after agglomerate hierarchical clustering algorithm. <br>Value range: between 1 and the initial number of input data.</br> | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| LINKAGE | INTEGER | 5 | Linkage type between two clusters.<ul><li>5: Centroid Clustering</li><li>6: Median Clustering</li><li>7: Ward Method</li></ul> | |

### Examples

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.
  Example 1: K-Medoids

```
SET SCHEMA DM_PAL;
DROP TABLE META_DATA_TABLE;
CREATE COLUMN TABLE META_DATA_TABLE (
"VARIABLE_NAME" VARCHAR (100),
"VARIABLE_VALUE" VARCHAR (100)
);
--Define point name & type(CONTINUOUS or CATEGORICAL); distance(must be
CONTINUOUS)
INSERT INTO META_DATA_TABLE VALUES ('LEFT_POINT', 'CATEGORICAL');
INSERT INTO META_DATA_TABLE VALUES ('RIGHT_POINT', 'CATEGORICAL');
INSERT INTO META_DATA_TABLE VALUES ('DISTANCE', 'CONTINUOUS');
DROP TABLE MY_DATA_TABLE;
CREATE COLUMN TABLE MY_DATA_TABLE (
"ID" VARCHAR(100),
"NAME" VARCHAR (100),
"VALUE" VARCHAR (100),
"PURPOSE" INTEGER
);
INSERT INTO MY_DATA_TABLE VALUES ('A0', 'LEFT_POINT','0',1);
INSERT INTO MY_DATA_TABLE VALUES ('A0', 'RIGHT_POINT','1',1);
INSERT INTO MY_DATA_TABLE VALUES ('A0', 'DISTANCE','0.707',1);
INSERT INTO MY_DATA_TABLE VALUES ('A1', 'LEFT_POINT','0',1);
INSERT INTO MY_DATA_TABLE VALUES ('A1', 'RIGHT_POINT','2',1);
INSERT INTO MY_DATA_TABLE VALUES ('A1', 'DISTANCE','12.747',1);
INSERT INTO MY_DATA_TABLE VALUES ('A2', 'LEFT_POINT','0',1);
INSERT INTO MY_DATA_TABLE VALUES ('A2', 'RIGHT_POINT','3',1);
INSERT INTO MY_DATA_TABLE VALUES ('A2', 'DISTANCE','12.727',1);
INSERT INTO MY_DATA_TABLE VALUES ('A3', 'LEFT_POINT','1',1);
INSERT INTO MY_DATA_TABLE VALUES ('A3', 'RIGHT_POINT','2',1);
INSERT INTO MY_DATA_TABLE VALUES ('A3', 'DISTANCE','12.727',1);
INSERT INTO MY_DATA_TABLE VALUES ('A4', 'LEFT_POINT','1',1);
INSERT INTO MY_DATA_TABLE VALUES ('A4', 'RIGHT_POINT','3',1);
INSERT INTO MY_DATA_TABLE VALUES ('A4', 'DISTANCE','12.747',1);
INSERT INTO MY_DATA_TABLE VALUES ('A5', 'LEFT_POINT','2',1);
INSERT INTO MY_DATA_TABLE VALUES ('A5', 'RIGHT_POINT','3',1);
INSERT INTO MY_DATA_TABLE VALUES ('A5', 'DISTANCE','0.707',1);
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL (
"NAME" VARCHAR (50),
"INT_VALUE" INTEGER,
"DOUBLE_VALUE" DOUBLE,
"STRING_VALUE" VARCHAR (100)
);
INSERT INTO PAL_PARAMETER_TBL VALUES ('FUNCTION', NULL, NULL, 'KMEDOIDS');
INSERT INTO PAL_PARAMETER_TBL VALUES ('N_CLUSTERS', 2, NULL, NULL);
```

```
INSERT INTO PAL_PARAMETER_TBL VALUES ('PRECALCULATED_DISTANCE', 1, NULL, NULL);
DROP TABLE PAL_CLUSTERING_MODEL_TBL;
CREATE COLUMN TABLE PAL_CLUSTERING_MODEL_TBL (
"ROW_INDEX" INTEGER,
"PART_INDEX" INTEGER,
"MODEL_CONTENT" VARCHAR (5000)
);
DO BEGIN
lt_meta = SELECT * FROM META_DATA_TABLE;
lt_data = SELECT * FROM MY_DATA_TABLE;
lt_param = SELECT * FROM PAL_PARAMETER_TBL;
CALL "_SYS_AFL"."PAL_UNIFIED_CLUSTERING_PIVOT"(:lt_meta, :lt_data, :lt_param,
lt_result, lt_centers, lt_model, lt_stat, lt_optimal_parameter, lt_place1,
lt_place2);
INSERT INTO PAL_CLUSTERING_MODEL_TBL SELECT * FROM :lt_model;
END;
```

**Expected Result**:

RESULT:

Result 1 × Result 2 × Result 3 × Result 4 × Messages × History

Rows (4)

| | ID | CLUSTER_ID | DISTANCE | SLIGHT_SILHOUETTE |
|---|---|---|---|---|
| 1 | 3 | 0 | 0 | *NULL* |
| 2 | 1 | 1 | 0 | *NULL* |
| 3 | 2 | 0 | 0.707 | *NULL* |
| 4 | 0 | 1 | 0.707 | *NULL* |

CENTERS:

Result 1 × Result 2 × Result 3 × Result 4 × Messages × History

Rows (2)

| | CLUSTER_ID | VARIABLE_NAME | VALUE |
|---|---|---|---|
| 1 | 0 | LEFT_POINT | 3 |
| 2 | 1 | LEFT_POINT | 1 |

MODEL:

Result 1 × Result 2 × Result 3 × Result 4 × Messages × History

Rows (3)

| | ROW_INDEX | PART_INDEX | MODEL_CONTENT |
|---|---|---|---|
| 1 | 0 | *NULL* | {"Algorithm":"KMEDOIDSWITHDIS","ColumnInfo":{"DataType":83},"Cluster":[{"ClusterID":0,"CenterCoor":"3"},{"ClusterID":1,"CenterCoor":"1"}]} |
| 2 | -1 | *NULL* | CLUSTER_ASSIGNMENT_PIVOT |
| 3 | -2 | *NULL* | {"feature":["LEFT_POINT","RIGHT_POINT","DISTANCE"],"featureNum":3,"index":[0,1,2],"type":["CATEGORICAL","CATEGORICAL","CONTINUOUS"]} |

Example 2: SP

```
SET SCHEMA DM_PAL;
DROP TABLE META_DATA_TABLE;
CREATE COLUMN TABLE META_DATA_TABLE (
"VARIABLE_NAME" VARCHAR (100),
"VARIABLE_VALUE" VARCHAR (100)
);
INSERT INTO META_DATA_TABLE VALUES ('LEFT_POINT', 'CATEGORICAL');
INSERT INTO META_DATA_TABLE VALUES ('RIGHT_POINT', 'CATEGORICAL');
INSERT INTO META_DATA_TABLE VALUES ('DISTANCE', 'CONTINUOUS');
DROP TABLE MY_DATA_TABLE;
CREATE COLUMN TABLE MY_DATA_TABLE (
"ID" VARCHAR(100),
"NAME" VARCHAR (100),
```

```
"VALUE" VARCHAR (100),
"PURPOSE" INTEGER
);
-- If the distance value == 0, there is no need to input this unconnected point.
INSERT INTO MY_DATA_TABLE VALUES ('A0', 'LEFT_POINT','0',1);
INSERT INTO MY_DATA_TABLE VALUES ('A0', 'RIGHT_POINT','1',1);
INSERT INTO MY_DATA_TABLE VALUES ('A0', 'DISTANCE','0.707',1);
INSERT INTO MY_DATA_TABLE VALUES ('A1', 'LEFT_POINT','0',1);
INSERT INTO MY_DATA_TABLE VALUES ('A1', 'RIGHT_POINT','2',1);
INSERT INTO MY_DATA_TABLE VALUES ('A1', 'DISTANCE','12.747',1);
INSERT INTO MY_DATA_TABLE VALUES ('A2', 'LEFT_POINT','0',1);
INSERT INTO MY_DATA_TABLE VALUES ('A2', 'RIGHT_POINT','3',1);
INSERT INTO MY_DATA_TABLE VALUES ('A2', 'DISTANCE','12.727',1);
INSERT INTO MY_DATA_TABLE VALUES ('A3', 'LEFT_POINT','1',1);
INSERT INTO MY_DATA_TABLE VALUES ('A3', 'RIGHT_POINT','2',1);
INSERT INTO MY_DATA_TABLE VALUES ('A3', 'DISTANCE','12.727',1);
INSERT INTO MY_DATA_TABLE VALUES ('A4', 'LEFT_POINT','1',1);
INSERT INTO MY_DATA_TABLE VALUES ('A4', 'RIGHT_POINT','3',1);
INSERT INTO MY_DATA_TABLE VALUES ('A4', 'DISTANCE','12.747',1);
INSERT INTO MY_DATA_TABLE VALUES ('A5', 'LEFT_POINT','2',1);
INSERT INTO MY_DATA_TABLE VALUES ('A5', 'RIGHT_POINT','3',1);
INSERT INTO MY_DATA_TABLE VALUES ('A5', 'DISTANCE','0.707',1);
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL (
"NAME" VARCHAR (50),
"INT_VALUE" INTEGER,
"DOUBLE_VALUE" DOUBLE,
"STRING_VALUE" VARCHAR (100)
);
INSERT INTO PAL_PARAMETER_TBL VALUES ('FUNCTION', NULL, NULL, 'SP');
INSERT INTO PAL_PARAMETER_TBL VALUES ('N_CLUSTERS', 2, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('PRECALCULATED_DISTANCE', 1, NULL, NULL);
DROP TABLE PAL_CLUSTERING_MODEL_TBL;
CREATE COLUMN TABLE PAL_CLUSTERING_MODEL_TBL (
"ROW_INDEX" INTEGER,
"PART_INDEX" INTEGER,
"MODEL_CONTENT" VARCHAR (5000)
);
DO BEGIN
lt_meta = SELECT * FROM META_DATA_TABLE;
lt_data = SELECT * FROM MY_DATA_TABLE;
lt_param = SELECT * FROM PAL_PARAMETER_TBL;
CALL "_SYS_AFL"."PAL_UNIFIED_CLUSTERING_PIVOT"(:lt_meta, :lt_data, :lt_param,
lt_result, lt_centers, lt_model, lt_stat, lt_optimal_parameter, lt_place1,
lt_place2);
INSERT INTO PAL_CLUSTERING_MODEL_TBL SELECT * FROM :lt_model;
select * from :lt_result;
END;
```

**Expected Result:**

RESULT:

Result 1 × Result 2 × Result 3 × Result 4 × Messages × History

Rows (4) SQL

| | ID | CLUSTER_ID | DISTANCE | SLIGHT_SILHOUETTE |
|---|---|---|---|---|
| 1 | 3 | 0 | NULL | NULL |
| 2 | 1 | 0 | NULL | NULL |
| 3 | 2 | 1 | NULL | NULL |
| 4 | 0 | 1 | NULL | NULL |

## Clustering Assignment with Pivot

To do clustering assignment with pivot, the input structure needs to be the same as the clustering input structure and the model needs to be generated by the distance matrix procedure.

**Parameter Table**

Mandatory Parameters

None.

Optional Parameters

| Name | Data Type | Defalut Value | Description | Dependency |
|---|---|---|---|---|
| THREAD_RATIO | DOUBLE | -1.0 | The ratio of available threads for multi-thread tasks:<br>• 0: single thread;<br>• 0~1: percentage;<br>• others: heuristically determines the number of threads to use. | |

**Examples**

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
DROP TABLE MY_DATA_ASSIGNMENT_TABLE;
CREATE COLUMN TABLE MY_DATA_ASSIGNMENT_TABLE (
"DATA_ID" VARCHAR(100),
"NAME" VARCHAR (100),
"VALUE" VARCHAR (100)
);
-- 'DATA_ID' column type needs to be the same as the 'ID' column of clustering
input data.
--At least, The input data should contain the centers point.
--For this example, each assignment point(4, 5 point) should contain '3' and '1'
points.
INSERT INTO MY_DATA_ASSIGNMENT_TABLE VALUES ('A1', 'LEFT_POINT','5');
INSERT INTO MY_DATA_ASSIGNMENT_TABLE VALUES ('A1', 'RIGHT_POINT','2');
INSERT INTO MY_DATA_ASSIGNMENT_TABLE VALUES ('A1', 'DISTANCE','1.444');
INSERT INTO MY_DATA_ASSIGNMENT_TABLE VALUES ('A2', 'LEFT_POINT','5');
INSERT INTO MY_DATA_ASSIGNMENT_TABLE VALUES ('A2', 'RIGHT_POINT','1');
INSERT INTO MY_DATA_ASSIGNMENT_TABLE VALUES ('A2', 'DISTANCE','2.444');
INSERT INTO MY_DATA_ASSIGNMENT_TABLE VALUES ('A3', 'LEFT_POINT','5');
INSERT INTO MY_DATA_ASSIGNMENT_TABLE VALUES ('A3', 'RIGHT_POINT','3');
INSERT INTO MY_DATA_ASSIGNMENT_TABLE VALUES ('A3', 'DISTANCE','1.444');
INSERT INTO MY_DATA_ASSIGNMENT_TABLE VALUES ('A4', 'LEFT_POINT','4');
INSERT INTO MY_DATA_ASSIGNMENT_TABLE VALUES ('A4', 'RIGHT_POINT','1');
INSERT INTO MY_DATA_ASSIGNMENT_TABLE VALUES ('A4', 'DISTANCE','0.555');
INSERT INTO MY_DATA_ASSIGNMENT_TABLE VALUES ('A5', 'LEFT_POINT','4');
INSERT INTO MY_DATA_ASSIGNMENT_TABLE VALUES ('A5', 'RIGHT_POINT','3');
INSERT INTO MY_DATA_ASSIGNMENT_TABLE VALUES ('A5', 'DISTANCE','10.444');
DROP TABLE PAL_PARAMETER_TBL;
```

```
CREATE COLUMN TABLE PAL_PARAMETER_TBL (
"NAME" VARCHAR (50),
"INT_VALUE" INTEGER,
"DOUBLE_VALUE" DOUBLE,
"STRING_VALUE" VARCHAR (100)
);
CALL
_SYS_AFL.PAL_UNIFIED_CLUSTERING_ASSIGNMENT_PIVOT(MY_DATA_ASSIGNMENT_TABLE,PAL_CLU
STERING_MODEL_TBL, PAL_PARAMETER_TBL, ?,?);
```

**Expected Result**:

RESULT:

Result 1 × Result 2 × Messages × History

Rows (5) SQL

| | DATA_ID | CLUSTER_ID | DISTANCE |
|---|---|---|---|
| 1 | 3 | 0 | 0 |
| 2 | 1 | 1 | 0 |
| 3 | 4 | 1 | 0.555 |
| 4 | 2 | -1 | 1.7976931348623157e+308 |
| 5 | 5 | 0 | 1.444 |

## 5.13.7 Unified Massive Clustering

The unified massive clustering execution function is a unified interface to call clustering algorithms with unpivot data format.

The unified massive clustering algorithms include:

- AHC (Agglomerate Hierarchical Clustering)
- DBSCAN (Density-Based Spatial Clustering of Applications with Noise)
- GMM (Gaussian Mixture Model)
- KMEANS (K-Means)
- KMEDIANS (K-Medians)
- KMEDOIDS (K-Medoids)
- SOM (Self-Organizing Maps)
- AP (Affinity Propagation)
- SP (Spectral Clustering)

The keyword group name ('PAL_MASSIVE_PROCESSING_SPECIAL_GROUP_ID') is valid for all groups if the group name is not defined. For example, there are two groups ('A', 'B'),

```
INSERT INTO #PAL_PARAMETER_TBL VALUES
('PAL_MASSIVE_PROCESSING_SPECIAL_GROUP_ID', 'DISTANCE_LEVEL', 2, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES
('PAL_MASSIVE_PROCESSING_SPECIAL_GROUP_ID', 'ALGORITHM', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('B', 'ALGORITHM', 0, NULL, NULL);
```

Therefore, A: DISTANCE_LEVEL(2), ALGORITHM(1); B: ALGORITHM(0).

For more details on the feature of the unified massive clustering, see the corresponding topics (Compatibility, Parameters Overview...) in the chapter on unified clustering.

Moreover, it is clear that the models created from the unified massive clustering and the unified clustering are incompatible.

## Prerequisites

- All prerequisites for the unified clustering function apply in the unified massive clustering execution interface.

## _SYS_AFL.PAL_UNIFIED_MASSIVE_CLUSTERING

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA Table>` | IN |
| 2 | `<PARAMETER Table>` | IN |
| 3 | `<RESULT Table>` | OUT |
| 4 | `<CENTERS Table>` | OUT |
| 5 | `<MODEL Table>` | OUT |
| 6 | `<STATISTICS Table>` | OUT |
| 7 | `<OPTIMAL_PARAMETER Table>` | OUT |
| 8 | `<ERRORMSG Table>` | OUT |
| 9 | `<PLACE_HOLDER1 Table>` | OUT |
| 10 | `<PLACE_HOLDER2 Table>` | OUT |

### Input Table

| Table | Column | Data Type | Description |
|---|---|---|---|
| DATA | 1st column | INTEGER, VRACHAR or NVARCHAR | Group id. |
| | 2nd column | BIGINT, INTEGER, VRACHAR, or NVARCHAR | Record id. |
| | Feature columns | INTEGER, DOUBLE, DECIMAL (p,s), VARCHAR, or NVARCHAR | Attribute data. |
| PARAMETER | 1st column | INTEGER, VRACHAR or NVARCHAR | Group id. |
| | 2nd column | VARCHAR or NVARCHAR | Parameter name. |
| | 3rd column | INTEGER | Int value. |

| Table | Column | Data Type | Description |
|---|---|---|---|
| | 4th column | DOUBLE | Double value. |
| | 5th column | VARCHAR or NVARCHAR | String value. |

### Parameter Table

Mandatory Parameters

The following parameters are mandatory and must be given a value.

| Name | Data Type | Description |
|---|---|---|
| FUNCTION | VARCHAR or NVARCHAR | The algorithm name, including:<br>• AHC<br>• DBSCAN<br>• GMM<br>• KMEANS<br>• KMEDIANS<br>• KMEDOIDS<br>• SOM<br>• AP<br>• SP |

Optional Parameters

The parameter of the unified massive clustering execution function with pivot is the same as the unified clustering function. See Parameter Tables [page 1799] for details about parameters for a specific algorithm.

### Output Tables

| Table | Column | Column Name | Column Data Type |
|---|---|---|---|
| RESULT | 1st column | GROUP_ID | Group id column data type. |
| | 2nd column | Record id column name | Record id column data type |
| | 3rd column | CLUSTER_ID | INTEGER |
| | 4th column | DISTANCE | DOUBLE |
| | 5th column | SLIGHT_SILHOUETTE | DOUBLE |
| CENTERS | 1st column | GROUP_ID | Group id column data type. |
| | 2nd column | CLUSTER_ID | INTEGER |
| | 3rd column | VARIABLE_NAME | VARCHAR or NVARCHAR |
| | 4th column | VALUE | VARCHAR or NVARCHAR |
| MODEL | 1st column | GROUP_ID | Group id column data type. |
| | 2nd column | ROW_INDEX | INTEGER |
| | 3rd column | PART_INDEX | INTEGER |
| | 4th column | MODEL_CONTENT | NCLOB |

| Table | Column | Column Name | Column Data Type |
|---|---|---|---|
| STATISTICS | 1st column | GROUP_ID | Group id column data type. |
| | 2nd column | STATISTIC_NAME | VARHCAR or NVARCHAR |
| | 3rd column | STATISTIC_VALUE | VARHCAR or NVARCHAR |
| OPTIMAL_PARAMETER | 1st column | GROUP_ID | Group id column data type. |
| | 2nd column | NAME | VARHCAR or NVARCHAR |
| | 3rd column | INT_VALUE | INTEGER |
| | 4th column | DOUBLE_VALUE | DOUBLE |
| | 5th column | STRING_VALUE | VARHCAR or NVARCHAR |
| ERRORMSG | 1st column | GROUP_ID | Group id column data type. |
| | 2nd column | ERROR_TIMESTAMP | VARCHAR, NVARCHAR |
| | 3rd column | ERRORCODE | INTEGER |
| | 4th column | MASSAGE | VARCHAR, NVARCHAR |
| PLCAE_HOLDER1 | 1st column | GROUP_ID | Group id column data type. |
| | 2nd column | Record id column name | Record id column data type |
| | 3rd column | TYPE | INTEGER |
| PLCAE_HOLDER2 | 1st column | GROUP_ID | Group id column data type. |
| | 2nd column | OBJECT | VARCHAR, NVARCHAR |
| | 3rd column | KEY | VARCHAR, NVARCHAR |
| | 4th column | VALUE | VARCHAR, NVARCHAR |

### Example

Assume that:

- DM_PAL is a schema belonging to USER1;
- USER1 has been assigned the AFL__SYS_AFL_AFLPAL_EXECUTE or AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION role.

Example: Density-Based Spatial Clustering of Applications with Noise

```
SET SCHEMA DM_PAL;
----------------------------------Unified Massive Clustering
(DBSCAN)--------------------------
DROP TABLE MY_DATA_TABLE;
CREATE COLUMN TABLE MY_DATA_TABLE (
"GROUP_ID" VARCHAR(20),
"DATA_ID" INTEGER,
"V1" DOUBLE,
"V2" DOUBLE,
"V3" VARCHAR(10)
);
INSERT INTO MY_DATA_TABLE VALUES('1', 1, 0.10, 0.10,  'B');
INSERT INTO MY_DATA_TABLE VALUES('1', 2, 0.11, 0.10,  'A');
INSERT INTO MY_DATA_TABLE VALUES('1', 3, 0.10, 0.11,  'C');
INSERT INTO MY_DATA_TABLE VALUES('1', 4, 0.11, 0.11,  'B');
INSERT INTO MY_DATA_TABLE VALUES('1', 5, 0.12, 0.11,  'A');
INSERT INTO MY_DATA_TABLE VALUES('1', 6, 0.11, 0.12,  'E');
```

```
INSERT INTO MY_DATA_TABLE VALUES('1', 7, 0.12, 0.12,  'A');
INSERT INTO MY_DATA_TABLE VALUES('1', 8, 0.12, 0.13,  'C');
INSERT INTO MY_DATA_TABLE VALUES('1', 9, 0.13, 0.12,  'D');
INSERT INTO MY_DATA_TABLE VALUES('1', 10, 0.13, 0.13,  'D');
INSERT INTO MY_DATA_TABLE VALUES('1', 11, 0.13, 0.14,  'A');
INSERT INTO MY_DATA_TABLE VALUES('1', 12, 0.14, 0.13,  'C');
INSERT INTO MY_DATA_TABLE VALUES('1', 13, 10.10, 10.10,  'A');
INSERT INTO MY_DATA_TABLE VALUES('1', 14, 10.11, 10.10,  'F');
INSERT INTO MY_DATA_TABLE VALUES('1', 15, 10.10, 10.11,  'E');
INSERT INTO MY_DATA_TABLE VALUES('1', 16, 10.11, 10.11,  'E');
INSERT INTO MY_DATA_TABLE VALUES('1', 17, 10.11, 10.12,  'A');
INSERT INTO MY_DATA_TABLE VALUES('1', 18, 10.12, 10.11,  'B');
INSERT INTO MY_DATA_TABLE VALUES('1', 19, 10.12, 10.12,  'B');
INSERT INTO MY_DATA_TABLE VALUES('1', 20, 10.12, 10.13,  'D');
INSERT INTO MY_DATA_TABLE VALUES('1', 21, 10.13, 10.12,  'F');
INSERT INTO MY_DATA_TABLE VALUES('1', 22, 10.13, 10.13,  'A');
INSERT INTO MY_DATA_TABLE VALUES('1', 23, 10.13, 10.14,  'A');
INSERT INTO MY_DATA_TABLE VALUES('1', 24, 10.14, 10.13,  'D');
INSERT INTO MY_DATA_TABLE VALUES('1', 25, 4.10, 4.10,  'A');
INSERT INTO MY_DATA_TABLE VALUES('1', 26, 7.11, 7.10,  'C');
INSERT INTO MY_DATA_TABLE VALUES('1', 27, -3.10, -3.11,  'C');
INSERT INTO MY_DATA_TABLE VALUES('1', 28, 16.11, 16.11,  'A');
INSERT INTO MY_DATA_TABLE VALUES('1', 29, 20.11, 20.12,  'C');
INSERT INTO MY_DATA_TABLE VALUES('1', 30, 15.12, 15.11,  'A');
INSERT INTO MY_DATA_TABLE VALUES('2', 1, 0.10, 0.10,  'B');
INSERT INTO MY_DATA_TABLE VALUES('2', 2, 0.11, 0.10,  'A');
INSERT INTO MY_DATA_TABLE VALUES('2', 3, 0.10, 0.11,  'C');
INSERT INTO MY_DATA_TABLE VALUES('2', 4, 0.11, 0.11,  'B');
INSERT INTO MY_DATA_TABLE VALUES('2', 5, 0.12, 0.11,  'A');
INSERT INTO MY_DATA_TABLE VALUES('2', 6, 0.11, 0.12,  'E');
INSERT INTO MY_DATA_TABLE VALUES('2', 7, 0.12, 0.12,  'A');
INSERT INTO MY_DATA_TABLE VALUES('2', 8, 0.12, 0.13,  'C');
INSERT INTO MY_DATA_TABLE VALUES('2', 9, 0.13, 0.12,  'D');
INSERT INTO MY_DATA_TABLE VALUES('2', 10, 0.13, 0.13,  'D');
INSERT INTO MY_DATA_TABLE VALUES('2', 11, 0.13, 0.14,  'A');
INSERT INTO MY_DATA_TABLE VALUES('2', 12, 0.14, 0.13,  'C');
INSERT INTO MY_DATA_TABLE VALUES('2', 13, 10.10, 10.10,  'A');
INSERT INTO MY_DATA_TABLE VALUES('2', 14, 10.11, 10.10,  'F');
INSERT INTO MY_DATA_TABLE VALUES('2', 15, 10.10, 10.11,  'E');
INSERT INTO MY_DATA_TABLE VALUES('2', 16, 10.11, 10.11,  'E');
INSERT INTO MY_DATA_TABLE VALUES('2', 17, 10.11, 10.12,  'A');
INSERT INTO MY_DATA_TABLE VALUES('2', 18, 10.12, 10.11,  'B');
INSERT INTO MY_DATA_TABLE VALUES('2', 19, 10.12, 10.12,  'B');
INSERT INTO MY_DATA_TABLE VALUES('2', 20, 10.12, 10.13,  'D');
INSERT INTO MY_DATA_TABLE VALUES('2', 21, 10.13, 10.12,  'F');
INSERT INTO MY_DATA_TABLE VALUES('2', 22, 10.13, 10.13,  'A');
INSERT INTO MY_DATA_TABLE VALUES('2', 23, 10.13, 10.14,  'A');
INSERT INTO MY_DATA_TABLE VALUES('2', 24, 10.14, 10.13,  'D');
INSERT INTO MY_DATA_TABLE VALUES('2', 25, 4.10, 4.10,  'A');
INSERT INTO MY_DATA_TABLE VALUES('2', 26, 7.11, 7.10,  'C');
INSERT INTO MY_DATA_TABLE VALUES('2', 27, -3.10, -3.11,  'C');
INSERT INTO MY_DATA_TABLE VALUES('2', 28, 16.11, 16.11,  'A');
INSERT INTO MY_DATA_TABLE VALUES('2', 29, 20.11, 20.12,  'C');
INSERT INTO MY_DATA_TABLE VALUES('2', 30, 15.12, 15.11,  'A');
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL (
"GROUP_ID" VARCHAR(50),
"NAME" VARCHAR (50),
"INT_VALUE" INTEGER,
"DOUBLE_VALUE" DOUBLE,
"STRING_VALUE" VARCHAR (100)
);
INSERT INTO PAL_PARAMETER_TBL VALUES ('PAL_MASSIVE_PROCESSING_SPECIAL_GROUP_ID',
'FUNCTION', NULL, NULL, 'DBSCAN');
INSERT INTO PAL_PARAMETER_TBL VALUES ('PAL_MASSIVE_PROCESSING_SPECIAL_GROUP_ID',
'AUTO_PARAM', NULL, NULL, 'true');
INSERT INTO PAL_PARAMETER_TBL VALUES ('PAL_MASSIVE_PROCESSING_SPECIAL_GROUP_ID',
'METRIC', 1, NULL, NULL);
```

```
INSERT INTO PAL_PARAMETER_TBL VALUES ('PAL_MASSIVE_PROCESSING_SPECIAL_GROUP_ID',
'THREAD_RATIO', NULL, 1.0, NULL);
--CALL _SYS_AFL.PAL_UNIFIED_MASSIVE_CLUSTERING(MY_DATA_TABLE,
PAL_PARAMETER_TBL, ?, ?,?,?,?,?,?,?);
DROP TABLE PAL_DATA_RESULT_TBL;
CREATE COLUMN TABLE PAL_DATA_RESULT_TBL(
     "GROUP_ID" VARCHAR(20),
    "ID" NVARCHAR(100),
    "CLUSTER_ID" INTEGER,
    "DISTANCE" DOUBLE,
    "SLIGHT_SILHOUETTE" DOUBLE
);
DROP TABLE PAL_DATA_CENTER_TBL;
CREATE COLUMN TABLE PAL_DATA_CENTER_TBL(
     "GROUP_ID" VARCHAR(20),
   "CLUSTER_ID" INTEGER,
   "VARIABLE_NAME" NVARCHAR(100),
   "VALUE" NVARCHAR(100)
);
DROP TABLE PAL_DATA_MODEL_TBL;
CREATE COLUMN TABLE PAL_DATA_MODEL_TBL(
     "GROUP_ID" VARCHAR(20),
   "ROW_INDEX" INTEGER,
   "PART_INDEX" INTEGER,
   "MODEL_CONTENT" VARCHAR (5000)
);
DROP TABLE PAL_DATA_STAT_TBL;
CREATE COLUMN TABLE PAL_DATA_STAT_TBL(
     "GROUP_ID" VARCHAR(20),
   "STATISTIC_NAME" VARCHAR (100),
   "STATISTIC_VALUE" VARCHAR (1000)
);
DROP TABLE PAL_DATA_OPT_PARA_TBL;
CREATE COLUMN TABLE PAL_DATA_OPT_PARA_TBL (
"GROUP_ID" VARCHAR(20),
"NAME" VARCHAR (50),
"INT_VALUE" INTEGER,
"DOUBLE_VALUE" DOUBLE,
"STRING_VALUE" VARCHAR (100)
);
DO BEGIN
  data_tab = SELECT * FROM MY_DATA_TABLE;
  control_tab = SELECT * FROM PAL_PARAMETER_TBL;
  CALL _SYS_AFL.PAL_UNIFIED_MASSIVE_CLUSTERING(:data_tab, :control_tab,
result_tab, center_tab, model_tab, statistic_tab, opt_para_tab, errormsg,
place_holder_1_tab, place_holder_2_tab);
  INSERT into PAL_DATA_RESULT_TBL SELECT * FROM :result_tab;
  INSERT into PAL_DATA_CENTER_TBL SELECT * FROM :center_tab;
  INSERT into PAL_DATA_MODEL_TBL SELECT * FROM :model_tab;
  INSERT into PAL_DATA_STAT_TBL SELECT * FROM :statistic_tab;
  INSERT into PAL_DATA_OPT_PARA_TBL SELECT * FROM :opt_para_tab;
  select * from :errormsg;
END;
select * from PAL_DATA_RESULT_TBL;
select * from PAL_DATA_CENTER_TBL;
select * from PAL_DATA_MODEL_TBL;
select * from PAL_DATA_STAT_TBL;
select * from PAL_DATA_OPT_PARA_TBL;
```

**Expected Result**:

RESULT:

Rows (60)

| | GROUP_ID | ID | CLUSTER_ID | DISTANCE | SLIGHT_SILHOUETTE |
|---|---|---|---|---|---|
| 22 | 1 | 7 | 1 | *NULL* | *NULL* |
| 23 | 1 | 8 | 1 | *NULL* | *NULL* |
| 24 | 1 | 9 | 1 | *NULL* | *NULL* |
| 25 | 1 | 10 | 1 | *NULL* | *NULL* |
| 26 | 1 | 11 | 1 | *NULL* | *NULL* |
| 27 | 1 | 12 | 1 | *NULL* | *NULL* |
| 28 | 1 | 13 | 0 | *NULL* | *NULL* |
| 29 | 1 | 14 | 0 | *NULL* | *NULL* |
| 30 | 1 | 15 | 0 | *NULL* | *NULL* |
| 31 | 2 | 16 | 0 | *NULL* | *NULL* |
| 32 | 2 | 17 | 0 | *NULL* | *NULL* |
| 33 | 2 | 18 | 0 | *NULL* | *NULL* |
| 34 | 2 | 19 | 0 | *NULL* | *NULL* |
| 35 | 2 | 20 | 0 | *NULL* | *NULL* |
| 36 | 2 | 21 | 0 | *NULL* | *NULL* |
| 37 | 2 | 22 | 0 | *NULL* | *NULL* |
| 38 | 2 | 23 | 0 | *NULL* | *NULL* |
| 39 | 2 | 24 | 0 | *NULL* | *NULL* |
| 40 | 2 | 25 | -1 | *NULL* | *NULL* |
| 41 | 2 | 26 | -1 | *NULL* | *NULL* |
| 42 | 2 | 27 | -1 | *NULL* | *NULL* |
| 43 | 2 | 28 | -1 | *NULL* | *NULL* |
| 44 | 2 | 29 | -1 | *NULL* | *NULL* |
| 45 | 2 | 30 | -1 | *NULL* | *NULL* |
| 46 | 2 | 15 | 0 | *NULL* | *NULL* |
| 47 | 2 | 14 | 0 | *NULL* | *NULL* |
| 48 | 2 | 13 | 0 | *NULL* | *NULL* |
| 49 | 2 | 12 | 1 | *NULL* | *NULL* |
| 50 | 2 | 11 | 1 | *NULL* | *NULL* |
| 51 | 2 | 10 | 1 | *NULL* | *NULL* |

MODEL:

Rows (2)

| | GROUP_ID | ROW_INDEX | PART_INDEX | |
|---|---|---|---|---|
| 1 | 1 | 0 | *NULL* | {"Algorithm":"DBSCAN","Category":[{"CategoryCol":3 |
| 2 | 2 | 0 | *NULL* | {"Algorithm":"DBSCAN","Category":[{"CategoryCol":3 |

## _SYS_AFL.PAL_UNIFIED_MASSIVE_CLUSTERING_ASSIGNMENT

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA Table>` | IN |
| 2 | `<MODEL Table>` | IN |
| 3 | `<PARAMETER Table>` | IN |
| 4 | `<ASSIGNMENT Table>` | OUT |
| 5 | `<ERRORMSG Table>` | OUT |
| 6 | `<PLACE_HOLDER1 Table>` | OUT |

### Input Table

| Table | Column | Data Type | Description |
|---|---|---|---|
| DATA | 1st column | INTEGER, VRACHAR, or NVARCHAR | Group id. |
| | 2nd column | BIGINT, INTEGER, VRACHAR or NVARCHAR | Record id. |
| | Feature columns | INTEGER, DOUBLE, DECIMAL (p,s), VARCHAR, or NVARCHAR | Attribute data |
| MODEL | 1st column | INTEGER, VRACHAR or NVARCHAR | Group id. |
| | 2nd column | INTEGER | Row index. |
| | 3rd column | INTEGER | Part index. |
| | 4th column | NCLOB | Model content. |
| PARAMETER | 1st column | INTEGER, VRACHAR, or NVARCHAR | Group id. |
| | 2nd column | VARCHAR or NVARCHAR | Parameter name. |
| | 3rd column | INTEGER | Int value. |
| | 4th column | DOUBLE | Double value. |
| | 5th column | VARCHAR or NVARCHAR | String value. |

### Parameter Table

The parameter of the unified massive clustering assignment execution function is the same as the unified clustering assignment function. See Parameter Tables [page 1799] for details about parameters for a specific algorithm.

### Output Tables

| Table | Column | Column Name | Column Data Type |
|---|---|---|---|
| ASSIGNMENT | 1st column | GROUP_ID | Group id column data type. |
| | 2nd column | Record id column name | Record id column data type |
| | 3rd column | CLUSTER_ID | INTEGER |
| | 4th column | DISTANCE | DOUBLE |
| ERRORMSG | 1st column | GROUP_ID | Group id column data type. |
| | 2nd column | ERROR_TIMESTAMP | VARCHAR, NVARCHAR |
| | 3rd column | ERRORCODE | INTEGER |
| | 4th column | MASSAGE | VARCHAR, NVARCHAR |
| PLCAE_HOLDER1 | 1st column | GROUP_ID | Group id column data type. |
| | 2nd column | OBJECT | VARCHAR, NVARCHAR |
| | 3rd column | KEY | VARCHAR, NVARCHAR |
| | 4th column | VALUE | VARCHAR, NVARCHAR |

### Example

Assume that:

- DM_PAL is a schema belonging to USER1;
- USER1 has been assigned the AFL__SYS_AFL_AFLPAL_EXECUTE or AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION role.

### Example: Clustering Assignment

```
-------------------------------------------Unified Massive Clustering(Clustering
Assignment)-------------------------------
DROP TABLE MY_DATA_ASSIGNMENT_TABLE;
CREATE COLUMN TABLE MY_DATA_ASSIGNMENT_TABLE (
    "GROUP_ID" VARCHAR(20),
    "DATA_ID" INTEGER,
    "V1" DOUBLE,
    "V2" DOUBLE,
    "V3" VARCHAR(10)
);
INSERT INTO MY_DATA_ASSIGNMENT_TABLE VALUES ('1', 1, 0.10, 0.10, 'B');
INSERT INTO MY_DATA_ASSIGNMENT_TABLE VALUES ('1', 2, 0.10, -2.10, 'A');
INSERT INTO MY_DATA_ASSIGNMENT_TABLE VALUES ('1', 3, 3.10, 0.10, 'B');
INSERT INTO MY_DATA_ASSIGNMENT_TABLE VALUES ('1', 4, 10.10, 10.10, 'D');
INSERT INTO MY_DATA_ASSIGNMENT_TABLE VALUES ('1', 5, 3.10, -0.50, 'C');
INSERT INTO MY_DATA_ASSIGNMENT_TABLE VALUES ('2', 1, 0.10, 0.10, 'B');
INSERT INTO MY_DATA_ASSIGNMENT_TABLE VALUES ('2', 2, 0.10, -2.10, 'A');
INSERT INTO MY_DATA_ASSIGNMENT_TABLE VALUES ('2', 3, 3.10, 0.10, 'B');
INSERT INTO MY_DATA_ASSIGNMENT_TABLE VALUES ('2', 4, 10.10, 10.10, 'D');
INSERT INTO MY_DATA_ASSIGNMENT_TABLE VALUES ('2', 5, 3.10, -0.50, 'C');
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL (
"GROUP_ID" VARCHAR(50),
"NAME" VARCHAR (50),
"INT_VALUE" INTEGER,
"DOUBLE_VALUE" DOUBLE,
"STRING_VALUE" VARCHAR (100)
);
```

```
CALL
_SYS_AFL.PAL_UNIFIED_MASSIVE_CLUSTERING_ASSIGNMENT(MY_DATA_ASSIGNMENT_TABLE,
PAL_DATA_MODEL_TBL, PAL_PARAMETER_TBL, ?, ?, ?);
```

**Expected Result**:

ASSIGNMENT:

Rows (10)

| | GROUP_ID | DATA_ID | CLUSTER_ID | DISTANCE |
|---|---|---|---|---|
| 1 | 2 | 1 | 1 | 0 |
| 2 | 2 | 2 | 1 | 2.21 |
| 3 | 2 | 3 | -1 | 3 |
| 4 | 2 | 4 | 0 | 0.05000000000000071 |
| 5 | 2 | 5 | -1 | 3.59 |
| 6 | 1 | 1 | 1 | 0 |
| 7 | 1 | 2 | 1 | 2.21 |
| 8 | 1 | 3 | -1 | 3 |
| 9 | 1 | 4 | 0 | 0.05000000000000071 |
| 10 | 1 | 5 | -1 | 3.59 |

## 5.13.8 Unified Massive Clustering with Pivoted Input Data

The unified massive clustering with pivoted execution function is a unified interface to call clustering algorithms with pivot data format.

The unified massive clustering with pivoted algorithms include:

- AHC (Agglomerate Hierarchical Clustering)
- DBSCAN (Density-Based Spatial Clustering of Applications with Noise)
- GMM (Gaussian Mixture Model)
- KMEANS (K-Means)
- KMEDIANS (K-Medians)
- KMEDOIDS (K-Medoids)
- SOM (Self-Organizing Maps)
- AP (Affinity Propagation)
- SP (Spectral Clustering)

The keyword group name ('PAL_MASSIVE_PROCESSING_SPECIAL_GROUP_ID') is valid for all groups if the group name is not defined. For example, there are two groups ('A', 'B'),

```
INSERT INTO #PAL_PARAMETER_TBL VALUES
('PAL_MASSIVE_PROCESSING_SPECIAL_GROUP_ID', 'DISTANCE_LEVEL', 2, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES
('PAL_MASSIVE_PROCESSING_SPECIAL_GROUP_ID', 'ALGORITHM', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('B', 'ALGORITHM', 0, NULL, NULL);
```

Therefore, A: DISTANCE_LEVEL(2), ALGORITHM(1); B: ALGORITHM(0).

For more details on the feature of the unified massive clustering with pivot, see the corresponding topics (Parameters Overview) in the chapter on unified clustering with pivot.

Moreover, it is clear that the models created from the unified massive clustering with pivot and the unified clustering with pivot are incompatible.

## Prerequisites

- All prerequisites for the unified clustering with pivot function apply in the unified massive clustering with pivot execution interface.

## _SYS_AFL.PAL_UNIFIED_MASSIVE_CLUSTERING_PIVOT

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<METADATA Table>` | IN |
| 2 | `<DATA Table>` | IN |
| 3 | `<PARAMETER Table>` | IN |
| 4 | `<RESULT Table>` | OUT |
| 5 | `<CENTERS Table>` | OUT |
| 6 | `<MODEL Table>` | OUT |
| 7 | `<STATISTICS Table>` | OUT |
| 8 | `<OPTIMAL_PARAMETER Table>` | OUT |
| 9 | `<ERRORMSG Table>` | OUT |
| 10 | `<PLACE_HOLDER1 Table>` | OUT |
| 11 | `<PLACE_HOLDER2 Table>` | OUT |

### Input Table

| Table | Column | Data Type | Description |
|---|---|---|---|
| METADATA | 1st column | VARCHAR or NVARCHAR | The name of the variable. |
| | 2nd column | VARCHAR or NVARCHAR | The type of the variable:<br>• CONTINUOUS<br>• CATEGORICAL |
| DATA | 1st column | INTEGER, VRACHAR, or NVARCHAR | Group id. |

| Table | Column | Data Type | Description |
|---|---|---|---|
| | 2nd column | BIGINT, INTEGER, VRACHAR, or NVARCHAR | Record id. |
| | 3rd column | VARCHAR or NVARCHAR | Variable name. |
| | 4th column | VARCHAR or NVARCHAR | Variable value. |
| | 5th column | INTEGER | Self-defined data partition:<br>• 1: Clustering<br>• 2: Testing<br>So far, only support clustering. |
| PARAMETER | 1st column | INTEGER, VRACHAR, or NVARCHAR | Group id. |
| | 2nd column | VARCHAR or NVARCHAR | Parameter name. |
| | 3rd column | INTEGER | Int value. |
| | 4th column | DOUBLE | Double value. |
| | 5th column | VARCHAR or NVARCHAR | String value. |

### Parameter Table

Mandatory Parameters

The following parameters are mandatory and must be given a value.

| Name | Data Type | Description |
|---|---|---|
| FUNCTION | VARCHAR or NVARCHAR | The algorithm name, including:<br>• AHC<br>• DBSCAN<br>• GMM<br>• KMEANS<br>• KMEDIANS<br>• KMEDOIDS<br>• SOM<br>• AP<br>• SP |

Optional Parameters

The parameter of the unified massive clustering with pivot execution function is the same as the unified clustering with pivot function. See Parameter Tables [page 1843] for details about parameters for a specific algorithm.

### Output Tables

| Table | Column | Column Name | Column Data Type |
|---|---|---|---|
| RESULT | 1st column | GROUP_ID | Group id column data type. |

| Table | Column | Column Name | Column Data Type |
|---|---|---|---|
| | 2nd column | Record id column name | Record id column data type |
| | 3rd column | CLUSTER_ID | INTEGER |
| | 4th column | DISTANCE | DOUBLE |
| | 5th column | SLIGHT_SILHOUETTE | DOUBLE |
| CENTERS | 1st column | GROUP_ID | Group id column data type. |
| | 2nd column | CLUSTER_ID | INTEGER |
| | 3rd column | VARIABLE_NAME | VARCHAR or NVARCHAR |
| | 4th column | VALUE | VARCHAR or NVARCHAR |
| MODEL | 1st column | GROUP_ID | Group id column data type. |
| | 2nd column | ROW_INDEX | INTEGER |
| | 3rd column | PART_INDEX | INTEGER |
| | 4th column | MODEL_CONTENT | NCLOB |
| STATISTICS | 1st column | GROUP_ID | Group id column data type. |
| | 2nd column | STATISTIC_NAME | VARHCAR or NVARCHAR |
| | 3rd column | STATISTIC_VALUE | VARHCAR or NVARCHAR |
| OPTIMAL_PARAMETER | 1st column | GROUP_ID | Group id column data type. |
| | 2nd column | NAME | VARHCAR or NVARCHAR |
| | 3rd column | INT_VALUE | INTEGER |
| | 4th column | DOUBLE_VALUE | DOUBLE |
| | 5th column | STRING_VALUE | VARHCAR or NVARCHAR |
| ERRORMSG | 1st column | GROUP_ID | Group id column data type. |
| | 2nd column | ERROR_TIMESTAMP | VARCHAR, NVARCHAR |
| | 3rd column | ERRORCODE | INTEGER |
| | 4th column | MASSAGE | VARCHAR, NVARCHAR |
| PLCAE_HOLDER1 | 1st column | GROUP_ID | Group id column data type. |
| | 2nd column | Record id column name | Record id column data type |
| | 3rd column | TYPE | INTEGER |
| PLCAE_HOLDER2 | 1st column | GROUP_ID | Group id column data type. |
| | 2nd column | OBJECT | VARCHAR, NVARCHAR |
| | 3rd column | KEY | VARCHAR, NVARCHAR |
| | 4th column | VALUE | VARCHAR, NVARCHAR |

### Example

Assume that:

- DM_PAL is a schema belonging to USER1;

- USER1 has been assigned the AFL__SYS_AFL_AFLPAL_EXECUTE or AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION role.

**Example: Density-Based Spatial Clustering of Applications with Noise**

```
SET SCHEMA DM_PAL;
-----------------------------------Unified Massive Clustering Pivot
(DBSCAN)--------------------------
DROP TABLE META_DATA_TABLE;
CREATE COLUMN TABLE META_DATA_TABLE (
"NAME" VARCHAR (100),
"VALUE" VARCHAR (100)
);
INSERT INTO META_DATA_TABLE VALUES ('V1', 'CONTINUOUS');
INSERT INTO META_DATA_TABLE VALUES ('V2', 'CONTINUOUS');
INSERT INTO META_DATA_TABLE VALUES ('V3', 'CATEGORICAL');
--Data_table
DROP TABLE MY_DATA_TABLE;
CREATE COLUMN TABLE MY_DATA_TABLE (
"GROUP_ID" INTEGER,
"ID" INTEGER,
"NAME" VARCHAR (100),
"VALUE" VARCHAR (100),
"PURPOSE" INTEGER
);
INSERT INTO MY_DATA_TABLE VALUES (1, 1, 'V1','0.10',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 1, 'V2','0.10',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 1, 'V3','B',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 2, 'V1','0.11',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 2, 'V2','0.10',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 2, 'V3','A',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 3, 'V1','0.10',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 3, 'V2','0.11',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 3, 'V3','C',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 4, 'V1','0.11',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 4, 'V2','0.11',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 4, 'V3','B',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 5, 'V1','0.12',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 5, 'V2','0.11',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 5, 'V3','A',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 6, 'V1','0.11',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 6, 'V2','0.12',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 6, 'V3','E',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 7, 'V1','0.12',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 7, 'V2','0.12',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 7, 'V3','A',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 8, 'V1','0.12',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 8, 'V2','0.13',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 8, 'V3','C',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 9, 'V1','0.13',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 9, 'V2','0.12',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 9, 'V3','D',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 10, 'V1','0.13',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 10, 'V2','0.13',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 10, 'V3','D',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 11, 'V1','0.13',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 11, 'V2','0.14',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 11, 'V3','A',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 12, 'V1','0.14',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 12, 'V2','0.13',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 12, 'V3','C',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 13, 'V1','10.10',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 13, 'V2','10.10',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 13, 'V3','A',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 14, 'V1','10.11',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 14, 'V2','10.10',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 14, 'V3','F',1);
```

```
INSERT INTO MY_DATA_TABLE VALUES (1, 15, 'V1','10.10',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 15, 'V2','10.11',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 15, 'V3','E',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 16, 'V1','10.11',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 16, 'V2','10.11',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 16, 'V3','E',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 17, 'V1','10.11',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 17, 'V2','10.12',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 17, 'V3','A',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 18, 'V1','10.12',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 18, 'V2','10.11',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 18, 'V3','B',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 19, 'V1','10.12',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 19, 'V2','10.12',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 19, 'V3','B',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 20, 'V1','10.12',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 20, 'V2','10.13',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 20, 'V3','D',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 21, 'V1','10.13',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 21, 'V2','10.12',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 21, 'V3','F',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 22, 'V1','10.13',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 22, 'V2','10.13',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 22, 'V3','F',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 23, 'V1','10.13',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 23, 'V2','10.14',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 23, 'V3','A',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 24, 'V1','10.14',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 24, 'V2','10.13',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 24, 'V3','D',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 25, 'V1','4.10',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 25, 'V2','4.10',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 25, 'V3','A',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 26, 'V1','7.11',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 26, 'V2','7.10',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 26, 'V3','C',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 27, 'V1','-3.10',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 27, 'V2','-3.11',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 27, 'V3','C',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 28, 'V1','16.11',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 28, 'V2','16.11',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 28, 'V3','A',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 29, 'V1','20.11',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 29, 'V2','20.12',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 29, 'V3','C',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 30, 'V1','15.12',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 30, 'V2','15.11',1);
INSERT INTO MY_DATA_TABLE VALUES (1, 30, 'V3','A',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 1, 'V1','0.10',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 1, 'V2','0.10',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 1, 'V3','B',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 2, 'V1','0.11',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 2, 'V2','0.10',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 2, 'V3','A',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 3, 'V1','0.10',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 3, 'V2','0.11',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 3, 'V3','C',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 4, 'V1','0.11',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 4, 'V2','0.11',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 4, 'V3','B',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 5, 'V1','0.12',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 5, 'V2','0.11',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 5, 'V3','A',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 6, 'V1','0.11',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 6, 'V2','0.12',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 6, 'V3','E',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 7, 'V1','0.12',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 7, 'V2','0.12',1);
```

```
INSERT INTO MY_DATA_TABLE VALUES (2, 7, 'V3','A',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 8, 'V1','0.12',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 8, 'V2','0.13',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 8, 'V3','C',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 9, 'V1','0.13',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 9, 'V2','0.12',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 9, 'V3','D',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 10, 'V1','0.13',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 10, 'V2','0.13',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 10, 'V3','D',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 11, 'V1','0.13',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 11, 'V2','0.14',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 11, 'V3','A',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 12, 'V1','0.14',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 12, 'V2','0.13',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 12, 'V3','C',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 13, 'V1','10.10',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 13, 'V2','10.10',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 13, 'V3','A',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 14, 'V1','10.11',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 14, 'V2','10.10',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 14, 'V3','F',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 15, 'V1','10.10',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 15, 'V2','10.11',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 15, 'V3','E',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 16, 'V1','10.11',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 16, 'V2','10.11',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 16, 'V3','E',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 17, 'V1','10.11',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 17, 'V2','10.12',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 17, 'V3','A',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 18, 'V1','10.12',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 18, 'V2','10.11',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 18, 'V3','B',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 19, 'V1','10.12',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 19, 'V2','10.12',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 19, 'V3','B',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 20, 'V1','10.12',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 20, 'V2','10.13',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 20, 'V3','D',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 21, 'V1','10.13',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 21, 'V2','10.12',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 21, 'V3','F',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 22, 'V1','10.13',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 22, 'V2','10.13',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 22, 'V3','F',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 23, 'V1','10.13',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 23, 'V2','10.14',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 23, 'V3','A',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 24, 'V1','10.14',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 24, 'V2','10.13',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 24, 'V3','D',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 25, 'V1','4.10',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 25, 'V2','4.10',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 25, 'V3','A',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 26, 'V1','7.11',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 26, 'V2','7.10',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 26, 'V3','C',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 27, 'V1','-3.10',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 27, 'V2','-3.11',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 27, 'V3','C',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 28, 'V1','16.11',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 28, 'V2','16.11',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 28, 'V3','A',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 29, 'V1','20.11',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 29, 'V2','20.12',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 29, 'V3','C',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 30, 'V1','15.12',1);
```

```
INSERT INTO MY_DATA_TABLE VALUES (2, 30, 'V2','15.11',1);
INSERT INTO MY_DATA_TABLE VALUES (2, 30, 'V3','A',1);
--parameter table
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL (
"GROUP_ID" INTEGER,
"NAME" VARCHAR (50),
"INT_VALUE" INTEGER,
"DOUBLE_VALUE" DOUBLE,
"STRING_VALUE" VARCHAR (100)
);
INSERT INTO PAL_PARAMETER_TBL VALUES (1, 'FUNCTION', NULL, NULL, 'DBSCAN');
INSERT INTO PAL_PARAMETER_TBL VALUES (1, 'AUTO_PARAM', NULL, NULL, 'true');
INSERT INTO PAL_PARAMETER_TBL VALUES (1, 'METRIC', 1, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES (1, 'THREAD_RATIO', NULL, 1.0, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES (2, 'FUNCTION', NULL, NULL, 'DBSCAN');
INSERT INTO PAL_PARAMETER_TBL VALUES (2, 'AUTO_PARAM', NULL, NULL, 'true');
INSERT INTO PAL_PARAMETER_TBL VALUES (2, 'METRIC', 1, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES (2, 'THREAD_RATIO', NULL, 1.0, NULL);
--CALL
_SYS_AFL.PAL_UNIFIED_MASSIVE_CLUSTERING_PIVOT(META_DATA_TABLE,MY_DATA_TABLE,
PAL_PARAMETER_TBL, ?,?,?,?,?,?,?,?);
DROP TABLE PAL_DATA_RESULT_TBL;
CREATE COLUMN TABLE PAL_DATA_RESULT_TBL(
     "GROUP_ID" INTEGER,
   "ID" NVARCHAR(100),
   "CLUSTER_ID" INTEGER,
   "DISTANCE" DOUBLE,
   "SLIGHT_SILHOUETTE" DOUBLE
);
DROP TABLE PAL_DATA_CENTER_TBL;
CREATE COLUMN TABLE PAL_DATA_CENTER_TBL(
     "GROUP_ID" INTEGER,
   "CLUSTER_ID" INTEGER,
   "VARIABLE_NAME" NVARCHAR(100),
   "VALUE" NVARCHAR(100)
);
DROP TABLE PAL_DATA_MODEL_TBL;
CREATE COLUMN TABLE PAL_DATA_MODEL_TBL(
     "GROUP_ID" INTEGER,
   "ROW_INDEX" INTEGER,
   "PART_INDEX" INTEGER,
   "MODEL_CONTENT" VARCHAR (5000)
);
DROP TABLE PAL_DATA_STAT_TBL;
CREATE COLUMN TABLE PAL_DATA_STAT_TBL(
     "GROUP_ID" INTEGER,
   "STATISTIC_NAME" VARCHAR (100),
   "STATISTIC_VALUE" VARCHAR (1000)
);
DROP TABLE PAL_DATA_OPT_PARA_TBL;
CREATE COLUMN TABLE PAL_DATA_OPT_PARA_TBL (
"GROUP_ID" INTEGER,
"NAME" VARCHAR (50),
"INT_VALUE" INTEGER,
"DOUBLE_VALUE" DOUBLE,
"STRING_VALUE" VARCHAR (100)
);
DO BEGIN
  meta_data_tab = SELECT * FROM META_DATA_TABLE;
  data_tab = SELECT * FROM MY_DATA_TABLE;
  control_tab = SELECT * FROM PAL_PARAMETER_TBL;
  CALL
_SYS_AFL.PAL_UNIFIED_MASSIVE_CLUSTERING_PIVOT(:meta_data_tab, :data_tab, :control
_tab, result_tab, center_tab, model_tab, statistic_tab, opt_para_tab, errormsg,
place_holder_1_tab, place_holder_2_tab);
  INSERT into PAL_DATA_RESULT_TBL SELECT * FROM :result_tab;
  INSERT into PAL_DATA_CENTER_TBL SELECT * FROM :center_tab;
```

```
  INSERT into PAL_DATA_MODEL_TBL SELECT * FROM :model_tab;
  INSERT into PAL_DATA_STAT_TBL SELECT * FROM :statistic_tab;
  INSERT into PAL_DATA_OPT_PARA_TBL SELECT * FROM :opt_para_tab;
  select * from :errormsg;
END
select * from PAL_DATA_RESULT_TBL;
select * from PAL_DATA_CENTER_TBL;
select * from PAL_DATA_MODEL_TBL;
select * from PAL_DATA_STAT_TBL;
select * from PAL_DATA_OPT_PARA_TBL;
```

**Expected Result**:

RESULT:

Rows (60)

| | GROUP_ID | ID | CLUSTER_ID | DISTANCE | SLIGHT_SILHOUETTE |
|---|---|---|---|---|---|
| 21 | 1 | 21 | 1 | NULL | NULL |
| 22 | 1 | 22 | 1 | NULL | NULL |
| 23 | 1 | 23 | 1 | NULL | NULL |
| 24 | 1 | 24 | 1 | NULL | NULL |
| 25 | 1 | 25 | -1 | NULL | NULL |
| 26 | 1 | 26 | -1 | NULL | NULL |
| 27 | 1 | 27 | -1 | NULL | NULL |
| 28 | 1 | 28 | -1 | NULL | NULL |
| 29 | 1 | 29 | -1 | NULL | NULL |
| 30 | 1 | 30 | -1 | NULL | NULL |
| 31 | 2 | 1 | 0 | NULL | NULL |
| 32 | 2 | 2 | 0 | NULL | NULL |
| 33 | 2 | 3 | 0 | NULL | NULL |
| 34 | 2 | 4 | 0 | NULL | NULL |
| 35 | 2 | 5 | 0 | NULL | NULL |
| 36 | 2 | 6 | 0 | NULL | NULL |
| 37 | 2 | 7 | 0 | NULL | NULL |
| 38 | 2 | 8 | 0 | NULL | NULL |
| 39 | 2 | 9 | 0 | NULL | NULL |
| 40 | 2 | 10 | 0 | NULL | NULL |
| 41 | 2 | 11 | 0 | NULL | NULL |
| 42 | 2 | 12 | 0 | NULL | NULL |
| 43 | 2 | 13 | 1 | NULL | NULL |

MODEL:

Rows (6)

| | GROUP_ID | ROW_INDEX | PART_INDEX | |
|---|---|---|---|---|
| 1 | 1 | 0 | *NULL* | {"Algorithm":"DBSCAN","Category":[{"CategoryCol":3 |
| 2 | 1 | -1 | *NULL* | CLUSTER_ASSIGNMENT_PIVOT |
| 3 | 1 | -2 | *NULL* | {"feature":["V1","V2","V3"],"featureNum":3,"index":[0,1 |
| 4 | 2 | 0 | *NULL* | {"Algorithm":"DBSCAN","Category":[{"CategoryCol":3 |
| 5 | 2 | -1 | *NULL* | CLUSTER_ASSIGNMENT_PIVOT |
| 6 | 2 | -2 | *NULL* | {"feature":["V1","V2","V3"],"featureNum":3,"index":[0,1 |

## _SYS_AFL.PAL_UNIFIED_MASSIVE_CLUSTERING_ASSIGNMENT_PIVOT

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA Table>` | IN |
| 2 | `<MODEL Table>` | IN |
| 3 | `<PARAMETER Table>` | IN |
| 4 | `<ASSIGNMENT Table>` | OUT |
| 5 | `<ERRORMSG Table>` | OUT |
| 6 | `<PLACE_HOLDER1 Table>` | OUT |

### Input Table

| Table | Column | Data Type | Description |
|---|---|---|---|
| DATA | 1st column | INTEGER, VRACHAR, or NVARCHAR | Group id. |
| | 2nd column | BIGINT, INTEGER, VRACHAR, or NVARCHAR | Record id. |
| | 3rd column | VARCHAR or NVARCHAR | Variable name. |
| | 4th column | VARCHAR or NVARCHAR | Variable value. |
| MODEL | 1st column | INTEGER, VRACHAR, or NVARCHAR | Group id. |
| | 2nd column | INTEGER | Row index. |
| | 3rd column | INTEGER | Part index. |
| | 4th column | NCLOB | Model content. |
| PARAMETER | 1st column | INTEGER, VRACHAR, or NVARCHAR | Group id. |

| Table | Column | Data Type | Description |
|---|---|---|---|
| | 2nd column | VARCHAR or NVARCHAR | Parameter name. |
| | 3rd column | INTEGER | Int value. |
| | 4th column | DOUBLE | Double value. |
| | 5th column | VARCHAR or NVARCHAR | String value. |

### Parameter Table

The parameter of the unified massive clustering assignment with pivot execution function is the same as the unified clustering assignment with pivot function.

See Parameter Tables [page 1843] for details about parameters for a specific algorithm.

### Output Tables

| Table | Column | Column Name | Column Data Type |
|---|---|---|---|
| ASSIGNMENT | 1st column | GROUP_ID | Group id column data type. |
| | 2nd column | Record id column name | Record id column data type |
| | 3rd column | CLUSTER_ID | INTEGER |
| | 4th column | DISTANCE | DOUBLE |
| ERRORMSG | 1st column | GROUP_ID | Group id column data type. |
| | 2nd column | ERROR_TIMESTAMP | VARCHAR, NVARCHAR |
| | 3rd column | ERRORCODE | INTEGER |
| | 4th column | MASSAGE | VARCHAR, NVARCHAR |
| PLCAE_HOLDER1 | 1st column | GROUP_ID | Group id column data type. |
| | 2nd column | OBJECT | VARCHAR, NVARCHAR |
| | 3rd column | KEY | VARCHAR, NVARCHAR |
| | 4th column | VALUE | VARCHAR, NVARCHAR |

### Example

Assume that:

- DM_PAL is a schema belonging to USER1;
- USER1 has been assigned the AFL__SYS_AFL_AFLPAL_EXECUTE or AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION role.

Example: Clustering Assignment

```
-----------------------------------------Unified Massive assignment
Pivot(clustering assignment)-------------------------------
DROP TABLE MY_DATA_ASSIGNMENT_TABLE;
CREATE COLUMN TABLE MY_DATA_ASSIGNMENT_TABLE (
"GROUP_ID" INTEGER,
"DATA_ID" INTEGER,
"NAME" VARCHAR (100),
"VALUE" VARCHAR (100)
);
INSERT INTO MY_DATA_ASSIGNMENT_TABLE VALUES (1, 1, 'V1','0.10');
```

```
INSERT INTO MY_DATA_ASSIGNMENT_TABLE VALUES (1, 1, 'V2','0.10');
INSERT INTO MY_DATA_ASSIGNMENT_TABLE VALUES (1, 1, 'V3','B');
INSERT INTO MY_DATA_ASSIGNMENT_TABLE VALUES (1, 2, 'V1','0.10');
INSERT INTO MY_DATA_ASSIGNMENT_TABLE VALUES (1, 2, 'V2','-2.10');
INSERT INTO MY_DATA_ASSIGNMENT_TABLE VALUES (1, 2, 'V3','A');
INSERT INTO MY_DATA_ASSIGNMENT_TABLE VALUES (1, 3, 'V1','3.10');
INSERT INTO MY_DATA_ASSIGNMENT_TABLE VALUES (1, 3, 'V2','0.10');
INSERT INTO MY_DATA_ASSIGNMENT_TABLE VALUES (1, 3, 'V3','B');
INSERT INTO MY_DATA_ASSIGNMENT_TABLE VALUES (1, 4, 'V1','10.10');
INSERT INTO MY_DATA_ASSIGNMENT_TABLE VALUES (1, 4, 'V2','10.10');
INSERT INTO MY_DATA_ASSIGNMENT_TABLE VALUES (1, 4, 'V3','D');
INSERT INTO MY_DATA_ASSIGNMENT_TABLE VALUES (1, 5, 'V1','3.10');
INSERT INTO MY_DATA_ASSIGNMENT_TABLE VALUES (1, 5, 'V2','-0.50');
INSERT INTO MY_DATA_ASSIGNMENT_TABLE VALUES (1, 5, 'V3','C');
INSERT INTO MY_DATA_ASSIGNMENT_TABLE VALUES (2, 1, 'V1','0.10');
INSERT INTO MY_DATA_ASSIGNMENT_TABLE VALUES (2, 1, 'V2','0.10');
INSERT INTO MY_DATA_ASSIGNMENT_TABLE VALUES (2, 1, 'V3','B');
INSERT INTO MY_DATA_ASSIGNMENT_TABLE VALUES (2, 2, 'V1','0.10');
INSERT INTO MY_DATA_ASSIGNMENT_TABLE VALUES (2, 2, 'V2','-2.10');
INSERT INTO MY_DATA_ASSIGNMENT_TABLE VALUES (2, 2, 'V3','A');
INSERT INTO MY_DATA_ASSIGNMENT_TABLE VALUES (2, 3, 'V1','3.10');
INSERT INTO MY_DATA_ASSIGNMENT_TABLE VALUES (2, 3, 'V2','0.10');
INSERT INTO MY_DATA_ASSIGNMENT_TABLE VALUES (2, 3, 'V3','B');
INSERT INTO MY_DATA_ASSIGNMENT_TABLE VALUES (2, 4, 'V1','10.10');
INSERT INTO MY_DATA_ASSIGNMENT_TABLE VALUES (2, 4, 'V2','10.10');
INSERT INTO MY_DATA_ASSIGNMENT_TABLE VALUES (2, 4, 'V3','D');
INSERT INTO MY_DATA_ASSIGNMENT_TABLE VALUES (2, 5, 'V1','3.10');
INSERT INTO MY_DATA_ASSIGNMENT_TABLE VALUES (2, 5, 'V2','-0.50');
INSERT INTO MY_DATA_ASSIGNMENT_TABLE VALUES (2, 5, 'V3','C');
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL (
"GROUP_ID" INTEGER,
"NAME" VARCHAR (50),
"INT_VALUE" INTEGER,
"DOUBLE_VALUE" DOUBLE,
"STRING_VALUE" VARCHAR (100)
);
CALL
_SYS_AFL.PAL_UNIFIED_MASSIVE_CLUSTERING_ASSIGNMENT_PIVOT(MY_DATA_ASSIGNMENT_TABLE
, PAL_DATA_MODEL_TBL, PAL_PARAMETER_TBL, ?, ?, ?);
```

**Expected Result**:

ASSIGNMENT:

Rows (10) SQL

| | GROUP_ID | DATA_ID | CLUSTER_ID | DISTANCE |
|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 |
| 2 | 1 | 2 | 0 | 2.21 |
| 3 | 1 | 3 | -1 | 3 |
| 4 | 1 | 4 | 1 | 0.05000000000000071 |
| 5 | 1 | 5 | -1 | 3.59 |
| 6 | 2 | 1 | 0 | 0 |
| 7 | 2 | 2 | 0 | 2.21 |
| 8 | 2 | 3 | -1 | 3 |
| 9 | 2 | 4 | 1 | 0.05000000000000071 |
| 10 | 2 | 5 | -1 | 3.59 |

## 5.13.9 Unified Regression

Unified regression is a unified interface to call some regression algorithms with unpivot data format.

The regression algorithms include:

- GEO (Bi-Variate Geometric Regression)
- LOG (Bi-Variate Natural Logarithmic Regression)
- EXP (Exponential Regression)
- GLM (Generalised Linear Models)
- MLR (Multiple Linear Regression)
- POL(Polynomial Regression)
- DT (Decision Trees)
- HGBT (Hybrid Gradient Boosting Tree)
- MLP (Multilayer Perceptron)
- RDT (Random Decision Trees)
- SVM (Support Vector Machine)

For MLP, unified regression just can support single target column and it will throw an error if calling multiple target columns.

For DT, HGBT, RDT and SVM, unified regression supports continue target column and runs successful with regression algorithm only.

See the corresponding topic for details about a specific regression method.

### Prerequisites

- All prerequisites for the original corresponding regression function apply in the unified regression interface.

### _SYS_AFL.PAL_UNIFIED_REGRESSION

#### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA Table>` | IN |
| 2 | `<PARAMETER Table>` | IN |
| 3 | `<MODEL Table>` | OUT |
| 4 | `<STATISTICS Table>` | OUT |
| 5 | `<OPTIMAL_PARAMETER Table>` | OUT |
| 6 | `<PARTITION_TYPE Table>` | OUT |

| Position | Table Name | Parameter Type |
|---|---|---|
| 7 | `<PLACE_HOLDER_1 Table>` | OUT |
| 8 | `<PLACE_HOLDER_2 Table>` | OUT |

### Input Table

| Table | Column | Data Type | Description | Constraint |
|---|---|---|---|---|
| DATA | 1st column | INTEGER, VARCHAR, or NVARCHAR | Record ID.<br>If this column is absent, the KEY parameter must be set to 0. | |
| | Feature columns | INTEGER, DOUBLE, DECIMAL(p,s), VARCHAR, or NVARCHAR | Variable values. | |
| | LABEL column | INTEGER or DOUBLE | Dependent variable. | Usually this column is the last column except the following two scenarios:<br>• LABEL is set to be another column.<br>• The optional purpose column is needed, which means PARTITION_METHOD is 1. |
| | Purpose column (Optional) | INTEGER | An optional column used for user-defined data partition.<br>This column is needed if PARTITION_METHOD is set to 1. Otherwise, the column does not exist. The valid values are:<br>• 1: training<br>• 2: validation | |

Parameter Table

Mandatory Parameters

The following parameters are mandatory and must be given a value.

| Name | Data Type | Description |
| --- | --- | --- |
| FUNCTION | VARCHAR or NVARCHAR | The algorithm name, including:<br>• GEO;<br>• LOG;<br>• EXP;<br>• GLM;<br>• MLR;<br>• POL;<br>• DT;<br>• HGBT;<br>• MLP;<br>• RDT;<br>• SVM |

Optional Parameters

The following parameters are for partition processing only. See Parameter Tables [page 1910] for details about parameters for a specific algorithm.

| Name | Data Type | Default Value | Description | Dependency |
| --- | --- | --- | --- | --- |
| PARTITION_METHOD | INTEGER | 0 | Data set Parition via different algorithm:<br>• 0: Do not use partition;<br>• 1: Data set partition with specific flag;<br>• 2: Data set partition with random. | If the parameter is set to be 1, an additional purpose column in data table is needed. |
| PARTITION_TRAINING_PERCENT | DOUBLE | 0.8 | Specific scale value to data partition with random.<br>Value: 0-1 | Valid only for partition_method=2. |
| PARTITION_RANDOM_SEED | INTEGER | 0 | Indicates the seed used to initialize the random number generator.<br>• 0: Uses the system time;<br>• Not 0: Uses the specific seed. | Valid only for partition_method=2. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| OUTPUT_PARTI-TION_RESULT | INTEGER | 0 | Output partition result or not:<br>• 0: Do not output result;<br>• 1: Output result. | |

### Output Tables

| Table | Column | Column Name | Column Data Type |
|---|---|---|---|
| MODEL | 1st column | ROW_INDEX | INTEGER |
| | 2st column | PART_INDEX | INTEGER |
| | 3rd column | MODEL_CONTENT | VARCHAR, NVARCHAR, NCLOB |
| STATISTICS | 1st column | STAT_NAME | VARCHAR, NVARCHAR |
| | 2nd column | STAT_VALUE | VARCHAR, NVARCHAR |
| OPTIMAL_PARAMETER | 1st column | PARAM_NAME | VARCHAR, NVARCHAR |
| | 2nd column | INT_VALUE | INTEGER |
| | 3rd column | DOUBLE_VALUE | DOUBLE |
| | 4th column | STRING_VALUE | VARCHAR, NVARCHAR |
| PARTITION_TYPE | 1st column | ID column name (if there is no ID column, the name of first column of data table.) | ID column data type (if there is no ID column, the type of first column of data table.) |
| | 2nd column | TYPE | INTEGER |
| PLCAE_HOLDER_1 | 1st column | OBJECT | VARCHAR, NVARCHAR |
| | 2nd column | KEY | VARCHAR, NVARCHAR |
| | 3rd column | VALUE | VARCHAR, NVARCHAR |
| PLACE_HOLDER_2 | 1st column | OJBECT | VARCHAR, NVARCHAR |
| | 2nd column | KEY | VARCHAR, NVARCHAR |
| | 3rd column | VALUE | VARCHAR, NVARCHAR |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

Example:

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_DATA_TBL;
CREATE COLUMN TABLE PAL_DATA_TBL(
     "ID" INTEGER,
```

```
     "X1" DOUBLE,
     "X2" VARCHAR (100),
     "X3" INTEGER,
     "Y" DOUBLE
);
INSERT INTO PAL_DATA_TBL VALUES (0, 0.0, 'A', 1, -6.879);
INSERT INTO PAL_DATA_TBL VALUES (1, 0.50, 'A', 1, -3.449);
INSERT INTO PAL_DATA_TBL VALUES (2, 0.54, 'B', 1, 6.635);
INSERT INTO PAL_DATA_TBL VALUES (3, 1.04, 'B', 1, 11.844);
INSERT INTO PAL_DATA_TBL VALUES (4, 1.50, 'A', 1, 2.786);
INSERT INTO PAL_DATA_TBL VALUES (5, 0.04, 'B', 2, 2.389);
INSERT INTO PAL_DATA_TBL VALUES (6, 2.00, 'A', 2, -0.011);
INSERT INTO PAL_DATA_TBL VALUES (7, 2.04, 'B', 2, 8.839);
INSERT INTO PAL_DATA_TBL VALUES (8, 1.54, 'B', 1, 4.689);
INSERT INTO PAL_DATA_TBL VALUES (9, 1.00, 'A', 2, -5.507);
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO PAL_PARAMETER_TBL VALUES ('FUNCTION',null,null,'MLR');
INSERT INTO PAL_PARAMETER_TBL VALUES ('PARTITION_METHOD',2,null,null);
INSERT INTO PAL_PARAMETER_TBL VALUES ('PARTITION_TRAINING_PERCENT',
null,0.7,null);
INSERT INTO PAL_PARAMETER_TBL VALUES ('OUTPUT_PARTITION_RESULT', 1,null, null);
INSERT INTO PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', null,0.5,null);
INSERT INTO PAL_PARAMETER_TBL VALUES ('LABEL', null,null,'X3');
--CALL _SYS_AFL.PAL_UNIFIED_REGRESSION(PAL_DATA_TBL,
PAL_PARAMETER_TBL, ?, ?, ?, ?, ?, ?);
DROP TABLE PAL_DATA_MODEL_TBL;
CREATE COLUMN TABLE PAL_DATA_MODEL_TBL(
     "ROW_INDEX" INTEGER,
     "PART_INDEX" INTEGER,
     "MODEL_CONTENT" VARCHAR (5000)
);
DO BEGIN
  data_tab = SELECT * FROM PAL_DATA_TBL;
  control_tab = SELECT * FROM PAL_PARAMETER_TBL;
  CALL _SYS_AFL.PAL_UNIFIED_REGRESSION(:data_tab, :control_tab, model_tab,
stats_tab, optimal_parameter_tab, partition_result, place_holder_1_tab,
place_holder_2_tab);
  INSERT into PAL_DATA_MODEL_TBL SELECT * FROM :model_tab;
END
```

**Expected Result**:

MODEL:

| | ROW_INDEX | PART_INDEX | MODEL_CONTENT |
|---|---|---|---|
| 1 | 1 | ? | <PMML version="4.0" xmlns="http://www.dmg.org/PMML-4_0" xmlns:xsi="http://www.... |

STATISTIC:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | TEST_EVAR | -5.290135691785594 |
| 2 | TEST_MAE | 1.0691816553466666 |
| 3 | TEST_MAPE | 0.7173589913466666 |
| 4 | TEST_MAX_ERROR | 1.67586152754 |
| 5 | TEST_MSE | 1.4001176338811572 |
| 6 | TEST_R2 | -5.300529352465207 |
| 7 | TEST_RMSE | 1.1832656649633493 |
| 8 | TEST_WMAPE | 0.641508993208 |

OPTIMAL_PARAMETER:

PARTITION_TYPE:

| | ID | TYPE |
|---|---|---|
| 1 | 0 | 1 |
| 2 | 1 | 1 |
| 3 | 2 | 1 |
| 4 | 3 | 1 |
| 5 | 7 | 1 |
| 6 | 8 | 1 |
| 7 | 9 | 1 |
| 8 | 4 | 2 |
| 9 | 5 | 2 |
| 10 | 6 | 2 |

## _SYS_AFL.PAL_UNIFIED_REGRESSION_PREDICT

This function uses the trained model to make calculated predictions or predicted values.

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA Table>` | IN |
| 2 | `<MODEL Table>` | IN |
| 3 | `<PARAMETER Table>` | IN |
| 4 | `<RESULT Table>` | OUT |
| 5 | `<PLACE_HOLDER_1 Table>` | OUT |

### Input Table

| Table | Column | Data Type | Description |
|---|---|---|---|
| DATA | 1st column | INTEGER, VARCHAR, or NVARCHAR | Data ID. This column must be the first column in the table. |
| | Feature Columns | INTEGER, DOUBLE, DECIMAL(p,s), VARCHAR, or NVARCHAR | Independent variables. |

### Model Table

| Table | Column | Data Type | Description |
|---|---|---|---|
| MODEL | 1st column | INTEGER | |
| | 2nd column | INTEGER | |
| | 3rd column | VARCHAR, NVARCHAR, NCLOB | |

### Parameter Table

See more details in Parameter Tables [page 1910].

Mandatory parameters

| Name | Data Type | Description | Constraint |
|---|---|---|---|
| FUNCTION | VARCHAR or NVARCHAR | The algorithm name, including:<br>• GEO;<br>• LOG;<br>• EXP;<br>• GLM;<br>• MLR;<br>• POL;<br>• DT;<br>• HGBT;<br>• MLP;<br>• RDT;<br>• SVM | The parameter must be same with the FUCTION used to train the model. |

### Output Tables

| Table | Column | Column Name | Column Data | Description |
|---|---|---|---|---|
| RESULT | 1st column | ID column name | ID column data type | |
| | 2nd column | SCORE | DOUBLE | |
| | 3rd column | LOWER_BOUND | DOUBLE | |
| | 4th column | UPPER_BOUND | DOUBLE | |
| | 5th column | REASON | NCLOB | |

| Table | Column | Column Name | Column Data | Description |
|---|---|---|---|---|
| PLACE_HOLDER_1 | 1st column | OBJECT | VARCHAR, NVARCHAR | |
| | 2nd column | KEY | VARCHAR, NVARCHAR | |
| | 3rd column | VALUE | VARCHAR, NVARCHAR | |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_DATA_TBL;
CREATE COLUMN TABLE PAL_DATA_TBL(
     "ID" INTEGER,
     "X1" DOUBLE,
     "X2" VARCHAR (100),
     "X3" INTEGER,
     "Y" DOUBLE
);
INSERT INTO PAL_DATA_TBL VALUES (0, 0.0, 'A', 1, -6.879);
INSERT INTO PAL_DATA_TBL VALUES (1, 0.50, 'A', 1, -3.449);
INSERT INTO PAL_DATA_TBL VALUES (2, 0.54, 'B', 1, 6.635);
INSERT INTO PAL_DATA_TBL VALUES (3, 1.04, 'B', 1, 11.844);
INSERT INTO PAL_DATA_TBL VALUES (4, 1.50, 'A', 1, 2.786);
INSERT INTO PAL_DATA_TBL VALUES (5, 0.04, 'B', 2, 2.389);
INSERT INTO PAL_DATA_TBL VALUES (6, 2.00, 'A', 2, -0.011);
INSERT INTO PAL_DATA_TBL VALUES (7, 2.04, 'B', 2, 8.839);
INSERT INTO PAL_DATA_TBL VALUES (8, 1.54, 'B', 1, 4.689);
INSERT INTO PAL_DATA_TBL VALUES (9, 1.00, 'A', 2, -5.507);
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO PAL_PARAMETER_TBL VALUES ('FUNCTION',null,null,'MLR');
INSERT INTO PAL_PARAMETER_TBL VALUES ('PARTITION_METHOD',2,null,null);
INSERT INTO PAL_PARAMETER_TBL VALUES ('PARTITION_TRAINING_PERCENT',
null,0.7,null);
INSERT INTO PAL_PARAMETER_TBL VALUES ('OUTPUT_PARTITION_RESULT', 1,null, null);
INSERT INTO PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', null,0.5,null);
INSERT INTO PAL_PARAMETER_TBL VALUES ('LABEL', null,null,'X3');
--CALL _SYS_AFL.PAL_UNIFIED_REGRESSION(PAL_DATA_TBL,
PAL_PARAMETER_TBL, ?, ?, ?, ?, ?, ?);
DROP TABLE PAL_DATA_MODEL_TBL;
CREATE COLUMN TABLE PAL_DATA_MODEL_TBL(
     "ROW_INDEX" INTEGER,
     "PART_INDEX" INTEGER,
     "MODEL_CONTENT" VARCHAR (5000)
);
DO BEGIN
  data_tab = SELECT * FROM PAL_DATA_TBL;
  control_tab = SELECT * FROM PAL_PARAMETER_TBL;
```

```
  CALL _SYS_AFL.PAL_UNIFIED_REGRESSION(:data_tab, :control_tab, model_tab,
stats_tab, optimal_parameter_tab, partition_result, place_holder_1_tab,
place_holder_2_tab);
  INSERT into PAL_DATA_MODEL_TBL SELECT * FROM :model_tab;
END
-------------------------------Predict-------------------------------
DROP TABLE PAL_DATA_TBL_PRE;
CREATE COLUMN TABLE PAL_DATA_TBL_PRE(
     "ID" INTEGER,
     "X1" DOUBLE,
     "X2" VARCHAR (100),
     "Y" DOUBLE
);
INSERT INTO PAL_DATA_TBL_PRE VALUES (0, 1.69, 'B', 1.2);
INSERT INTO PAL_DATA_TBL_PRE VALUES (1, 0.054, 'B', 2.1);
INSERT INTO PAL_DATA_TBL_PRE VALUES (2, 0.123, 'A', 2.4);
INSERT INTO PAL_DATA_TBL_PRE VALUES (3, 1.980, 'A', 1.8);
INSERT INTO PAL_DATA_TBL_PRE VALUES (4, 0.563, 'A', 1.0);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FUNCTION',null,null,'MLR');
CALL _SYS_AFL.PAL_UNIFIED_REGRESSION_PREDICT(PAL_DATA_TBL_PRE,
PAL_DATA_MODEL_TBL, #PAL_PARAMETER_TBL, ?,?);
```

**Expected Result**:

RESULT:

| | ID | SCORE | UPPER_BOUND | LOWER_BOUND | REASON |
|---|---|---|---|---|---|
| 1 | 0 | 1.1024675 | ? | ? | ? |
| 2 | 1 | 1.494161876 | ? | ? | ? |
| 3 | 2 | 1.063347202 | ? | ? | ? |
| 4 | 3 | 0.61263748 | ? | ? | ? |
| 5 | 4 | 0.974764882 | ? | ? | ? |

## _SYS_AFL.PAL_UNIFIED_REGRESSION_SCORE

This function makes calculated predictions, and provides the evaluation statistics for the trained model.

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA Table>` | IN |
| 2 | `<MODEL Table>` | IN |
| 3 | `<PARAMETER Table>` | IN |
| 4 | `<RESULT Table>` | OUT |
| 5 | `<STATISTIC Table>` | OUT |

### Input Table

| Table | Column | Data Type | Description | Constraint |
|---|---|---|---|---|
| DATA | 1st column | INTEGER, VARCHAR, or NVARCHAR | Data ID. This column must be the first column in the table. | |
| | Feature columns | INTEGER, DOUBLE, DECIMAL(p,s), VARCHAR, or NVARCHAR | Independent variables. | |
| | LABEL column | INTEGER or DOUBLE | Dependent variable. | Usually this column is the last column except LABEL is set to be another column. |

### Model Table

| Table | Column | Data Type | Description | Constraint |
|---|---|---|---|---|
| MODEL | 1st column | INTEGER | | |
| | 2nd column | INTEGER | | |
| | 3rd column | VARCHAR, NVARCHAR, NCLOB | | |

### Parameter Table

See details in Parameter Tables [page 1910].

Mandatory parameters

| Name | Data Type | Description | Constraint |
|---|---|---|---|
| FUNCTION | VARCHAR or NVARCHAR | The algorithm name, including:<br>• GEO;<br>• LOG;<br>• EXP;<br>• GLM;<br>• MLR;<br>• POL;<br>• DT;<br>• HGBT;<br>• MLP;<br>• RDT;<br>• SVM | The parameter must be same with the FUCTION used to train the model. |

### Output Tables

| Table | Column | Column Name | Column Data Type | Description |
|---|---|---|---|---|
| RESULT | 1st column | ID column name | ID column data type | |
| | 2nd column | SCORE | DOUBLE | |
| | 3rd column | LOWER_BOUND | DOUBLE | |
| | 4th column | UPPER_BOUND | DOUBLE | |
| | 5th column | REASON | NCLOB | |
| STATISTICS | 1st column | STAT_NAME | NVARCHAR | |
| | 2nd column | STAT_VALUE | NVARCHAR | |

Example:

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_DATA_TBL;
CREATE COLUMN TABLE PAL_DATA_TBL(
     "ID" INTEGER,
     "X1" DOUBLE,
     "X2" VARCHAR (100),
     "X3" INTEGER,
     "Y" DOUBLE
);
INSERT INTO PAL_DATA_TBL VALUES (0, 0.0, 'A', 1, -6.879);
INSERT INTO PAL_DATA_TBL VALUES (1, 0.50, 'A', 1, -3.449);
INSERT INTO PAL_DATA_TBL VALUES (2, 0.54, 'B', 1, 6.635);
INSERT INTO PAL_DATA_TBL VALUES (3, 1.04, 'B', 1, 11.844);
INSERT INTO PAL_DATA_TBL VALUES (4, 1.50, 'A', 1, 2.786);
INSERT INTO PAL_DATA_TBL VALUES (5, 0.04, 'B', 2, 2.389);
INSERT INTO PAL_DATA_TBL VALUES (6, 2.00, 'A', 2, -0.011);
INSERT INTO PAL_DATA_TBL VALUES (7, 2.04, 'B', 2, 8.839);
INSERT INTO PAL_DATA_TBL VALUES (8, 1.54, 'B', 1, 4.689);
INSERT INTO PAL_DATA_TBL VALUES (9, 1.00, 'A', 2, -5.507);
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO PAL_PARAMETER_TBL VALUES ('FUNCTION',null,null,'MLR');
INSERT INTO PAL_PARAMETER_TBL VALUES ('PARTITION_METHOD',2,null,null);
INSERT INTO PAL_PARAMETER_TBL VALUES ('PARTITION_TRAINING_PERCENT',
null,0.7,null);
INSERT INTO PAL_PARAMETER_TBL VALUES ('OUTPUT_PARTITION_RESULT', 1,null, null);
INSERT INTO PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', null,0.5,null);
INSERT INTO PAL_PARAMETER_TBL VALUES ('LABEL', null,null,'X3');
--CALL _SYS_AFL.PAL_UNIFIED_REGRESSION(PAL_DATA_TBL,
PAL_PARAMETER_TBL, ?, ?, ?, ?, ?, ?);
DROP TABLE PAL_DATA_MODEL_TBL;
CREATE COLUMN TABLE PAL_DATA_MODEL_TBL(
     "ROW_INDEX" INTEGER,
     "PART_INDEX" INTEGER,
     "MODEL_CONTENT" VARCHAR (5000)
);
DO BEGIN
```

```
  data_tab = SELECT * FROM PAL_DATA_TBL;
  control_tab = SELECT * FROM PAL_PARAMETER_TBL;
  CALL _SYS_AFL.PAL_UNIFIED_REGRESSION(:data_tab, :control_tab, model_tab,
stats_tab, optimal_parameter_tab, partition_result, place_holder_1_tab,
place_holder_2_tab);
  INSERT into PAL_DATA_MODEL_TBL SELECT * FROM :model_tab;
END
-------------------------------Score------------------------------
DROP TABLE PAL_DATA_TBL_SCORE;
CREATE COLUMN TABLE PAL_DATA_TBL_SCORE(
     "ID" INTEGER,
     "X1" DOUBLE,
     "X2" VARCHAR (100),
     "X3" INTEGER,
     "Y" DOUBLE
);
INSERT INTO PAL_DATA_TBL_SCORE VALUES (0, 1.69, 'B', 1, 1.2);
INSERT INTO PAL_DATA_TBL_SCORE VALUES (1, 0.054, 'B', 2, 2.1);
INSERT INTO PAL_DATA_TBL_SCORE VALUES (2, 0.123, 'A', 2, 2.4);
INSERT INTO PAL_DATA_TBL_SCORE VALUES (3, 1.980, 'A', 1, 1.8);
INSERT INTO PAL_DATA_TBL_SCORE VALUES (4, 0.563, 'A', 1, 1.0);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FUNCTION',null,null,'MLR');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('LABEL', null,null,'X3');
CALL _SYS_AFL.PAL_UNIFIED_REGRESSION_SCORE(PAL_DATA_TBL_SCORE,
PAL_DATA_MODEL_TBL, #PAL_PARAMETER_TBL, ?,?);
```

**Expected Result**

RESULT:

| | ID | SCORE | UPPER_BOUND | LOWER_BOUND | REASON |
|---|---|---|---|---|---|
| 1 | 0 | 1.1024675 | ? | ? | ? |
| 2 | 1 | 1.494161876 | ? | ? | ? |
| 3 | 2 | 1.063347202 | ? | ? | ? |
| 4 | 3 | 0.61263748 | ? | ? | ? |
| 5 | 4 | 0.974764882 | ? | ? | ? |

STATISTICS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | EVAR | 0.43329943555046857 |
| 2 | MAE | 0.391511212 |
| 3 | MAPE | 0.24726211979999996 |
| 4 | MAX_ERR... | 0.9366527979999999 |
| 5 | MSE | 0.25887535866610845 |
| 6 | R2 | -0.07864732777545... |
| 7 | RMSE | 0.5087979546599106 |
| 8 | WMAPE | 0.2796508657142857 |

## 5.13.9.1 Parameters Specific to Unified Regression

The following table lists mapping parameters between unified regression procedures and original procedures.

| Algorithm | New Name | Original Name |
|---|---|---|
| GEO | DECOMPOSITION | ALG |
| | LABEL | DEPENDENT_VARIABLE |
| | KEY | HAS_ID |
| | FEATURES | SELECTED_FEATURES |
| LOG | DECOMPOSITION | ALG |
| | LABEL | DEPENDENT_VARIABLE |
| | KEY | HAS_ID |
| | FEATURES | SELECTED_FEATURES |
| EXP | DECOMPOSITION | ALG |
| | LABEL | DEPENDENT_VARIABLE |
| | KEY | HAS_ID |
| | FEATURES | SELECTED_FEATURES |
| GEO | DECOMPOSITION | ALG |
| | LABEL | DEPENDENT_VARIABLE |
| | KEY | HAS_ID |
| | FEATURES | SELECTED_FEATURES |
| GLM | TOL | CONVERGENCE_CRITERION |
| | LABEL | DEPENDENT_VARIABLE |
| | ALPHA | ENET_ALPHA |
| | ALPHA_RANGE | ENET_ALPHA_RANGE |
| | ALPHA_VALUES | ENET_ALPHA_VALUES |
| | LAMBDA | ENET_LAMBDA |
| | LAMBDA_RANGE | ENET_LAMBDA_RANGE |
| | LAMBDA_VALUES | ENET_LAMBDA_VALUES |
| | NUM_LAMBDA | ENET_NUM_LAMBDA |
| | HANDLE_MISSING_FIT | HANDLE_MISSING |
| | KEY | HAS_ID |
| | MAX_ITER | MAX_ITERATION |
| | QUASILIKELIHOOD | QUASI |
| MLR | SOLVER | ALG |
| | LABEL | DEPENDENT_VARIABLE |

| Algorithm | New Name | Original Name |
|---|---|---|
| | ALPHA | ENET_ALPHA |
| | ALPHA_RANGE | ENET_ALPHA_RANGE |
| | ALPHA_VALUES | ENET_ALPHA_VALUES |
| | LAMBDA | ENET_LAMBDA |
| | LAMBDA_RANGE | ENET_LAMBDA_RANGE |
| | LAMBDA_VALUES | ENET_LAMBDA_VALUES |
| | KEY | HAS_ID |
| | MAX_ITER | MAX_ITERATION |
| | INTERCEPT | NO_INTERCEPT |
| | TOL | THRESHOLD |
| | VAR_SELECT | VARIABLE_SELECTION |
| DT | ALLOW_MISSING_LABEL | ALLOW_MISSING_DEPENDENT |
| | LABEL | DEPENDENT_VARIABLE |
| | KEY | HAS_ID |
| | MIN_RECORDS_LEAF | MIN_RECORDS_OF_LEAF |
| | MIN_RECORDS_LEAF_RANGE | MIN_RECORDS_OF_LEAF_RANGE |
| | MIN_RECORDS_LEAF_VALUES | MIN_RECORDS_OF_LEAF_VALUES |
| | MIN_RECORDS_PARENT | MIN_RECORDS_OF_PARENT |
| | MIN_RECORDS_PARENT_RANGE | MIN_RECORDS_OF_ PARENT_RANGE |
| | MIN_RECORDS_PARENT_VALUES | MIN_RECORDS_OF_PARENT_VALUES |
| POL | DEGREE | POLYNOMIAL_NUM |
| | DECOMPOSITION | ALG |
| | LABEL | DEPENDENT_VARIABLE |
| | KEY | HAS_ID |
| | DEGREE_RANGE | POLYNOMIAL_NUM _RANGE |
| | DEGREE_VALUES | POLYNOMIAL_NUM _VALUES |
| | FEATURES | SELECTED_FEATURES |
| HGBT | COL_SUBSAMPLE_SPLIT | COL_SAMPLE_RATE_BYSPLIT |
| | COL_SUBSAMPLE_SPLIT_RANGE | COL_SAMPLE_RATE_BYSPLIT_RANGE |
| | COL_SUBSAMPLE_SPLIT_VALUES | COL_SAMPLE_RATE_BYSPLIT_VAL-UES |
| | COL_SUBSAMPLE_TREE | COL_SAMPLE_RATE_BYTREE |
| | COL_SUBSAMPLE_TREE_RANGE | COL_SAMPLE_RATE_BYTREE_RANGE |
| | COL_SUBSAMPLE_TREE_VALUES | COL_SAMPLE_RATE_BYTREE_VALUES |
| | LABEL | DEPENDENT_VARIABLE |

| Algorithm | New Name | Original Name |
|---|---|---|
| | LEARNING_RATE | ETA |
| | LEARNING_RATE_RANGE | ETA_RANGE |
| | LEARNING_RATE_VALUES | ETA_VALUES |
| | SPLIT_THRESHOLD | GAMMA |
| | SPLIT_THRESHOLD_RANGE | GAMMA_RANGE |
| | SPLIT_THRESHOLD_VALUES | GAMMA_VALUES |
| | KEY | HAS_ID |
| | N_ESTIMATORS | ITER_NUM |
| | N_ESTIMATORS_RANGE | ITER_NUM_RANGE |
| | N_ESTIMATORS_VALUES | ITER_NUM_VALUES |
| | LAMB | LAMBDA |
| | LAMB_RANGE | LAMBDA_RANGE |
| | LAMB_VALUES | LAMBDA_VALUES |
| | MIN_SAMPLES_WEIGHT_LEAF | MIN_CHILD_HESSIAN |
| | MIN_SAM-PLES_WEIGHT_LEAF_RANGE | MIN_CHILD_HESSIAN_RANGE |
| | MIN_SAMPLES_WEIGHT_LEAF_VAL-UES | MIN_CHILD_HESSIAN_VALUES |
| | MIN_SAMPLES_LEAF | NODE_SIZE |
| | MIN_SAMPLES_LEAF_RANGE | NODE_SIZE_RANGE |
| | MIN_SAMPLES_LEAF_VALUES | NODE_SIZE_VALUES |
| | MAX_W_IN_SPLIT | NODE_WEIGHT_CONSTRAINT |
| | MAX_W_IN_SPLIT_RANGE | NODE_WEIGHT_CONSTRAINT_RANGE |
| | MAX_W_IN_SPLIT_VALUES | NODE_WEIGHT_CONSTRAINT_VAL-UES |
| | SUBSAMPLE | ROW_SAMPLE_RATE |
| | SUBSAMPLE_RANGE | ROW_SAMPLE_RATE_RANGE |
| | SUBSAMPLE_VALUES | ROW_SAMPLE_RATE_VALUES |
| MLP | ACTIVATION | HIDDEN_LAYER_ACTIVE_FUNC |
| | OUTPUT_ACTIVATION | OUTPUT_LAYER_ACTIVE_FUNC |
| | LABEL | DEPENDENT_VARIABLE |
| | KEY | HAS_ID |
| | ACTIVATION_OPTIONS | HIDDEN_LAYER_ACTIVE_FUNC_VAL-UES |
| | HIDDEN_LAYER_SIZE_OPTIONS | HIDDEN_LAYER_SIZE_VALUES |
| | MAX_ITER | MAX_ITERATION |

| Algorithm | New Name | Original Name |
|---|---|---|
| | BATCH_SIZE | MINI_BATCH_SIZE |
| | BATCH_SIZE_RANGE | MINI_BATCH_SIZE_RANGE |
| | BATCH_SIZE_VALUES | MINI_BATCH_SIZE_VALUES |
| | MOMENTUM | MOMENTUM_FACTOR |
| | MOMENTUM_RANGE | MOMENTUM_FACTOR_RANGE |
| | MOMENTUM_VALUES | MOMENTUM_FACTOR_VALUES |
| | OUTPUT_ACTIVATION_OPTIONS | OUTPUT_LAYER_ACTIVE_FUNC_VALUES |
| RDT | ALLOW_MISSING_LABEL | ALLOW_MISSING_DEPENDENT |
| | LABEL | DEPENDENT_VARIABLE |
| | KEY | HAS_ID |
| | MIN_SAMPLES_LEAF | NODE_SIZE |
| | N_ESTIMATORS | TREES_NUM |
| | MAX_FEATURES | TRY_NUM |
| SVM | LABEL | DEPENDENT_VARIABLE |
| | TOL | ERROR_TOL |
| | KEY | HAS_ID |

## 5.13.9.2 Parameter Tables

This topic lists all parameters for unified regression.

### _SYS_AFL.PAL_UNIFIED_REGRESSION

#### UNIFIED REGRESSION

Mandatory parameters

| Name | Data Type | Description |
| --- | --- | --- |
| FUNCTION | VARCHAR or NVARCHAR | The algorithm name, including:<br>• GEO;<br>• LOG;<br>• EXP;<br>• GLM;<br>• MLR;<br>• POL;<br>• DT;<br>• HGBT;<br>• MLP;<br>• RDT;<br>• SVM |

Optional parameters

| Name | Data Type | Default Value | Description | Dependency |
| --- | --- | --- | --- | --- |
| PARTITION_METHOD | INTEGER | 0 | Data set Parition via different algorithm:<br>• 0: Do not use partition;<br>• 1: Data set partition with specific flag;<br>• 2: Data set partition with random. | If the parameter is set to be 1, an additional purpose column in data table is needed. |
| PARTITION_TRAINING_PERCENT | DOUBLE | 0.8 | Specific scale value to data partition with random.<br>Value: 0-1 | Valid only for partition_method=2. |
| PARTITION_RANDOM_SEED | INTEGER | 0 | Indicates the seed used to initialize the random number generator.<br>• 0: Uses the system time;<br>• Non-zero: Uses the specific seed. | Valid only for partition_method=2. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| OUTPUT_PARTITION_RESULT | INTEGER | 0 | Output partition result or not:<br>• 0: Does not output result<br>• 1: Output result | |
| IGNORE_ZERO | INTEGER | 0 | Valid values are:<br>• 0: Uses zero values in the actual dataset when calculating MPE or MAPE;<br>• 1: Ignores zero values in the actual dataset when calculating MPE or MAPE. | Only valid when MEASURE_NAME is MPE or MAPE. |
| LABEL | VARCHAR or NVARCHAR | No default value | Column name in the data table used as dependent variable. | Cannot be the first column name if KEY is 1. |
| KEY | INTEGER | 0 or 1 | Specifies whether the input data table has an ID column.<br>• 0: No<br>• 1: Yes | For GEO, LOG, EXP, GLM, MLR and POL, the default value is 1.<br>For DT, HGBT, MLP, RDT and SVM, the default value is 0. |
| HANDLE_MISSING_VALUE | INTEGER | 0 | Whether to handle missing values:<br>• 0: No<br>• 1: Yes | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| IMPUTATION_TYPE | INTEGER | 1 | Chooses the overall default imputation type for all columns:<br>• 0: NONE<br>• 1: MODE_MEAN<br>• 2: MODE_MEDIAN<br>• 3: MODE_ALL-ZERO<br>• 4: MODE_ALS<br>• 5: DELETE<br>NONE and DELETE: set all columns to imputation type NONE/DELETE.<br>MODE_MEAN, MODE_MEDIAN, MODE_ALLZERO, and MODE_ALS: set categorical columns to the first imputation type (the type before “”); set numerical columns to the second imputation type (the type after “”). ALLZERO means to fill all missing values in numerical columns with 0. | Only valid when HANDLE_MISSING_VALUE is 1. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| `<column name>`_IMPUTATION_TYPE | INTEGER | No default value | Sets imputation type of a specified column. This value overwrites the default imputation type deduced by the IMPUTATION_TYPE parameter:<br>• 0: NONE<br>• 1: DELETE<br>• 100: MODE<br>• 101: SPECIFIED_CATEGORICAL_VALUE<br>• 200: MEAN<br>• 201: MEDIAN<br>• 203:SPECIFIED_NUMERICAL_VALUE<br>• 204: ALS<br>For type 101 (SPECIFIED_CATEGORICAL_VALUE), use the following syntax to specify a categorical value to be filled in a column V0:<br>INSERT INTO #PAL_PARAMETER_TBL VALUES('V0_IMPUTATION_TYPE', 101, NULL, '`<CATEGORICAL_VALUE>`');<br>For type 203 (SPECIFIED_NUMERICAL_VALUE), use the following syntax to specify a numerical value to be filled in a column V1:<br>INSERT INTO #PAL_PARAMETER_TBL VALUES('V1_IMPUTA- | Only valid when HANDLE_MISSING_VALUE is 1. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| | | | TION_TYPE', 203, <NUMERICAL_VALUE>, NULL);<br><br>**Note**<br>A column of integer type can be set to type 101 (SPECIFIED_CATEGORICAL_VALUE) or type 203 (SPECIFIED_NUMERICAL_VALUE):<br>• when setting to type 101 (SPECIFIED_CATEGORICAL_VALUE), the string must contain a valid integer value, otherwise an error will be thrown;<br>• when setting to type 203 (SPECIFIED_NUMERICAL_VALUE), the double value will be converted to integer. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| ALS_FACTOR_NUMBER | INTEGER | 3 | Length of factor vectors, that is, the f in ALS model.<br>**Note**<br>ALS_FACTOR_NUMBER should be less than the number of numerical columns to get a meaningful result. | |
| ALS_REGULARIZATION | DOUBLE | 1e-2 | L2 regularization of the factors, that is, the λ in ALS model. | |
| ALS_MAX_ITERATION | INTEGER | 20 | Maximum number of iterations when training ALS model. | |
| ALS_SEED | INTEGER | 0 | Specifies the seed of the random number generator used in the training of ALS model:<br>• 0: Uses the current time as the seed<br>• Others: Uses the specified value as the seed. | |
| ALS_EXIT_THRESHOLD | DOUBLE | 0.0 | The training of ALS model exits if the value of cost function is not decreased more than the value of ALS_EXIT_THRESHOLD since the last check.<br>If ALS_EXIT_THRESHOLD is set to 0, there is no check and the training only exits on reaching the maximum number of iterations. Evaluations of cost function require additional calculations. You can set this parameter to 0 to avoid it. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| ALS_EXIT_INTERVAL | INTEGER | 5 | During the training of ALS model, it will calculate cost function and check the exit criteria every `<ALS_EXIT_INTERVAL>` iterations. Evaluations of cost function require additional calculations. | |
| ALS_LINEAR_SYSTEM_SOLVER | INTEGER | 0 | • 0: cholesky solver<br>• 1: cg solver (recommended when ALS_FACTOR_NUMBER is large) | |
| ALS_CG_MAX_ITERATION | INTEGER | 3 | Maximum number of iteration of cg solver. | |
| ALS_CENTERING | INTEGER | 1 | Whether to center the data by column before training ALS model.<br>• 0: no centering<br>• 1: centering. | |
| ALS_SCALING | INTEGER | 1 | Whether to scale the data by column before training ALS model.<br>• 0: no scaling<br>• 1: scaling | |
| SIGNIFICANCE_LEVEL | DOUBLE | 0.05 | The significance level for the confidence interval and prediction interval. Valid range is (0, 1). | |
| INTERVAL | INTEGER | 0 | Specifiecs the type of interval to output.<br>• 0: No interval will be caluclated and output;<br>• 1: Confidence Interval;<br>• 2: Prediction Interval. | |

## LOG

Mandatory parameters

None

Optional parameters

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| DECOMPOSITION | INTEGER | 1 | Matrix factorization type to use. Case-insensitive<br>• 0: LU Decomposition<br>• 1: QR Decomposition<br>• 2: SVD<br>• 5: Cholesky Decomposition |
| ADJUSTED_R2 | INTEGER | 0 | • 0: Does not output adjusted R square;<br>• 1: Outputs adjusted R square. |
| FEATURES | VARCHAR or NVARCHAR | No default value | A string to specify the features that will be processed. The pattern is "`X1`,...,`Xn`", where `Xi` is the corresponding column name in the data table. If this parameter is not specified, all the features will be processed. |

## GEO

Mandatory parameters

None

Optional parameters

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| DECOMPOSITION | INTEGER | 1 | Matrix factorization type to use. Case-insensitive.<br>• 0: LU Decomposition<br>• 1: QR Decomposition<br>• 2: SVD<br>• 5: Cholesky Decomposition |
| ADJUSTED_R2 | INTEGER | 0 | • 0: Does not output adjusted R square;<br>• 1: Outputs adjusted R square. |

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| FEATURES | VARCHAR or NVARCHAR | No default value | A string to specify the features that will be processed. The pattern is "`X1`,...,`Xn`", where `Xi` is the corresponding column name in the data table. If this parameter is not specified, all the features will be processed. |

## EXP

Mandatory parameters

None

Optional parameters

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| DECOMPOSITION | INTEGER | 1 | Matrix factorization type to use. Case-insensitive.<br>• 0: LU Decomposition<br>• 1: QR Decomposition<br>• 2: SVD<br>• 5: Cholesky Decomposition |
| ADJUSTED_R2 | INTEGER | 0 | • 0: Does not output adjusted R square;<br>• 1: Outputs adjusted R square. |
| FEATURES | VARCHAR or NVARCHAR | No default value | A string to specify the features that will be processed. The pattern is "`X1`,...,`Xn`", where `Xi` is the corresponding column name in the data table. If this parameter is not specified, all the features will be processed. |

## GLM

Mandatory parameters

None

Optional parameters

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| SOLVER | VARCHAR or NVARCHAR | irls | Specifies the optimisation algorithm.<br>• irls: iterative reweighted least square;<br>• cd: coordinate descent, solves ENET problem. | |
| FAMILY | VARCHAR or NVARCHAR | gaussian | Specifies the distribution:<br>• gaussian, normal;<br>• poisson;<br>• binomial;<br>• gamma;<br>• inversegaussian;<br>• negativebinomial. | |
| LINK | VARCHAR or NVARCHAR | • gaussian: identity;<br>• poisson: log;<br>• binomial: logit;<br>• gamma: reciprocal;<br>• inversegaussian: nversesquare;<br>• negativebinomial: log. | Specifies the link function:<br>• identity;<br>• log;<br>• logit;<br>• probit;<br>• comploglog;<br>• reciprocal, inverse;<br>• inversesquare;<br>• sqrt. | The supported link functions for each distribution are:<br>• gaussian, normal: identity, log, reciprocal, inverse;<br>• poisson: identity, log;<br>• binomial: logit, probit, comploglog, log;<br>• gamma: identity, reciprocal, inverse, log;<br>• inversegaussian: inversesquare, identity, reciprocal, inverse, log;<br>• negativebinomial: {identity, log, sqrt}. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| HANDLE_MISSING_FIT | INTEGER | 1 | Specify how to handle missing covariates:<br>• 0: throw error;<br>• 1: remove missing rows;<br>• 2: replace missing value with 0. | |
| QUASILIKELIHOOD | INTEGER | 0 | Indicates whether to use the quasi-likelihood to estimate over-dispersion.<br>• 0: Does not consider over-dispersion;<br>• 1: Considers over-dispersion. | Valid only for Poisson and Negative Binomial distribution. |
| GROUP_RESPONSE | INTEGER | 0 | Indicates whether the response consists of two columns.<br>• 0: Consists of only one column (the last column);<br>• 1: Consists of two columns (the 2nd last(k) and last(n) columns) | Valid only for Binomial distribution, n,k for $\binom{n}{k}$, respectively. |
| MAX_ITER | INTEGER | 100 for IRLS; 1e5 for coordinate descent | The maximum number of iterations for numeric optimization. | |
| TOL | DOUBLE | 1e-8 for IRLS; 1e-7 for coordinate descent. | The convergence criterion of coefficients for numeric optimization. | IRLS and coordinate descent evaluate the relative change of objective. |
| SIGNIFICANCE_LEVEL | DOUBLE | 0.05 | The significance level for the confidence interval of estimated coefficients. | Not for coordinate descent. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| ALPHA | DOUBLE | 1.0 | The elastic net mixing parameter. The value range is between 0 and 1 inclusively. | Valid only for coordinate descent. |
| LAMBDA | DOUBLE | No default value | Coefficient for L1 and L2 regularization. | Valid only for coordinate descent. |
| NUM_LAMBDA | INTEGER | 100 | The number of lambda values. | Valid only for coordinate descent. Takes effect only when LAMBDA is not given explicitly. |
| LAMBDA_MIN_RATIO | DOUBLE | 1e-2 or 1e-4 | The smallest value of lambda, as a fraction of the maximum lambda, where $\lambda_{max}$is the smallest value for which all coefficients are zero. | Valid only for coordinate descent. Takes effect only when LAMBDA is not given explicitly. When the number of observations is smaller than the number of covariates, LAMBDA_MIN_RATIO defaults to 1e-2; Otherwise, it defaults to 1e-4. |
| CATEGORICAL_VARIABLE | VARCHAR, or NVARCHAR | Detected from input data | Indicate which variables are cheated as categorical. The default behavior is:<br>• string: categorical;<br>• integer and double: continuous | Valid only for integer or string column. |
| EXCLUDED_FEATURE | VARCHAR or NVARCHAR | No default value | Excludes the indicated feature column. | Cannot be the ID column name and the column name of dependent variable. |

Model Evaluation and Parameter Selection

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| RESAMPLING_METHOD | VARCHAR or NVARCHAR | No default value | Specifies the resampling method for model evaluation or parameter selection.<br>• cv;<br>• bootstrap.<br>If no value is specified for this parameter, neither model evaluation nor parameter selection is activated. | |
| EVALUATION_METRIC | VARCHAR or NVARCHAR | No default value | Specifies the evaluation metric for model evaluation or parameter selection.<br>• RMSE;<br>• MAE. | |
| FOLD_NUM | INTEGER | No default value | Specifies the fold number for a cross validation method. | Mandatory and valid only when RESAMPLING_METHOD is set to cv. |
| REPEAT_TIMES | INTEGER | 1 | Specifies the number of repeat times for resampling. | |
| PARAM_SEARCH_STRATEGY | VARCHAR or NVARCHAR | No default value | Specifies the method to activate parameter selection.<br>• grid;<br>• random. | |
| RANDOM_SEARCH_TIMES | INTEGER | No default value | Specifies the number of times to randomly select candidate parameters for selection. | Mandatory and valid when PARAM_SEARCH_STRATEGY is set to random. |
| SEED | INTEGER | 0 | Specifies the seed for random generation.<br>Uses system time when 0 is specified. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| TIMEOUT | INTEGER | 0 | Specifies maximum running time for model evaluation or parameter selection, in seconds. No timeout when 0 is specified. | |
| PROGRESS_INDICATOR_ID | VARCHAR or NVARCHAR | No default value | Sets an ID of progress indicator for model evaluation or parameter selection. No progress indicator is active if no value is provided. | |
| LINK_VALUES | VARCHAR or NVARCHAR | No default value | The discrete set for candidate of link functions. | |
| LAMBDA_VALUES | VARCHAR or NVARCHAR | No default value | Specifies values of LAMBDA to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| LAMBDA_RANGE | VARCHAR or NVARCHAR | No default value | Specifies the range of LAMBDA to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| ALPHA_VALUES | VARCHAR or NVARCHAR | No default value | Specifies values of ALPHA to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| ALPHA_RANGE | VARCHAR or NVARCHAR | No default value | Specifies the range of parameter ALPHA to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |

## MLR

Mandatory Parameters

None.

Optional Parameters

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| THREAD_RATIO | DOUBLE | -1.0 | The ratio of available threads for multi-thread task:<br>• 0: single thread<br>• 0–1: uses the specified percentage of available threads. PAL uses all available threads if the number is 1.0.<br>• -1: uses all of available threads. | Only valid when SLOVER is 1,4,5 or 6. |
| SOLVER | INTEGER | 1 | Specifies algorithms for solving the least square problem:<br>• 1: QR decomposition (numerically stable, but fails when A is rank-deficient);<br>• 2: SVD (numerically stable and can handle rank deficiency but computationally expensive);<br>• 4: Cyclical coordinate descent method to solve elastic net regularized multiple linear regression;<br>• 5: Cholesky decomposition (fast but numerically unstable);<br>• 6: Alternating direction method of multipliers (ADMM) to solve elastic net regularized multiple linear regression. | The value 4 and 6 are supported only when VAR_SELECT is 0. |
| CATEGORICAL_VARIABLE | VARCHAR or NVARCHAR | No default value | Indicates by the column name whether or not a column data is actually corresponding to a category variable even the data type of this column is INTEGER.<br>By default, VARCHAR or NVARCHAR are category variable; INTEGER or DOUBLE are continuous variable. | The data type of the column specified by this parameter must be INTEGER. |
| MODEL_FORMAT | INTEGER | 1 | Specifies the tree model format for store:<br>• 1: JSON<br>• 2: PMML | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| VAR_SELECT | INTEGER | 0 | • 0: All variables are included;<br>• 1: Forward selection;<br>• 2: Backward selection;<br>• 3: Stepwise selection. | The value 1, 2, or 3 are supported only when SLOVER is not 4 or 6. |
| MANDATORY_FEATURE | VARCHAR or NVARCHAR | No default value | Specifies the column name that needs to be included in the final training model when executing the variable selection. This parameter can be specified multiple times, each time with one column name as feature.<br>**Note**<br>This parameter is a hint. There are exceptional cases that specified mandatory feature is excluded in the final model, for instance, some mandatory features can be represented as a linear combination of other features, among which some are also mandatory features. | Only valid when VAR_SELECT is not 0. |
| INTERCEPT | INTEGER | 0 | Specifies whether the intercept term should be ignored in the model.<br>• 0: Do not ignore;<br>• 1: Ignore. | |
| ALPHA_TO_ENTER | DOUBLE | 0.05 or 0.15 | P-value for forward selection and stepwise selection. | When VAR_SELECT is 1, default value is 0.05; when VAR_SELECT is 3, default value is 0.15. |
| ALPHA_TO_REMOVE | DOUBLE | 0.1 or 0.15 | P-value for backward selection and stepwise selection. | When VAR_SELECT is 2, default value is 0.1; when VAR_SELECT is 3, default value is 0.15. |
| LAMBDA | DOUBLE | No default value | Penalized weight. The value should be equal to or greater than 0. | Only valid when SLOVER is 4 or 6. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| ALPHA | DOUBLE | 1.0 | The elastic net mixing parameter. The value range is between 0 and 1 inclusively.<br>• 0: Ridge penalty;<br>• 1: LASSO penalty. | Only valid when SLOVER is 4 or 6. |
| HANDLE_MISSING | INTEGER | 1 | Whether to handle missing value:<br>• 0: No;<br>• 1: Yes. | |
| MAX_ITER | INTEGER | 1e5 | Maximum number of passes over training data. If convergence is not reached after the specified number of iterations, an error will be generated. | Only valid when SLOVER is 4 or 6. |
| TOL | DOUBLE | 1.0e-7 | Convergence threshold for coordinate descent. | Only valid when SLOVER is 4. |
| PHO | DOUBLE | 1.8 | Step size for ADMM. Generally, the value should be greater than 1. | Only valid when SLOVER is 6. |
| STAT_INF | INTEGER | 0 | Specifies whether to output t-value and Pr(>\|t\|) of coefficients in the Result table or not.<br>• 0: No;<br>• 1: Yes. | |
| ADJUSTED_R2 | INTEGER | 0 | Specifies whether to output adjusted R square or not.<br>• 0: No;<br>• 1: Yes. | |
| DW_TEST | INTEGER | 0 | Specifies whether to do Durbin-Watson test under the null hypothesis that the errors do not follow a first order autoregressive process.<br>• 0: No;<br>• 1: Yes. | Not available if elastic net regularization is enabled or INTERCEPT = 1. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| RESET_TEST | INTEGER | 1 | Specifies the order of Ramsey RESET test. Choosing 1 means this test will not be conducted. If you specify an INTEGER larger than 1, then the MLR function will run Ramsey RESET test with power of variables ranging from 2 to the value you specify. | Not available if elastic net regularization is enabled or INTERCEPT = 1. |
| BP_TEST | INTEGER | 0 | Specifies whether or not to do Breusch-Pagan test under the null hypothesis that homoscedasticity is satisfied.<br>• 0: No;<br>• 1: Yes. | Not available if elastic net regularization is enabled or INTERCEPT = 1. |
| KS_TEST | INTEGER | 0 | Specifies whether or not to do Kolmogorov-Smirnov normality test under the null hypothesis that if errors of MLR follow a normal distribution.<br>• 0: No;<br>• 1: Yes. | Not available if elastic net regularization is enabled or INTERCEPT = 1. |
| SELECTED_FEATURES | VARCHAR or NVARCHAR | No default value | A string to specify the features that will be processed. The pattern is "`X1`,...,`Xn`", where `Xi` is the corresponding column name in the data table. If this parameter is not specified, all the features will be processed. | |
| OUTPUT_COEFCOV | INTEGER | 0 | Specifies whether to output coefficient covariance information.<br>• 0: Doesn't output related information;<br>• 1: Outputs related information. | Valid only when JSON_EXPORT=1. |
| ONEHOT_MIN_FREQUENCY | INTEGER | 1 | Specifies the minimum frequency below which a category is considered infrequent. | Only valid when MODEL_FORMAT is 1. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| ONEHOT_MAX_CATEGORIES | INTEGER | 0 | Specifies an upper limit of the number of output features for each input feature. ONEHOT_MAX_CATEGORIES includes the feature that combines infrequent categories. | Only valid when MODEL_FORMAT is 1. |

Model Evaluation and Parameter Selection

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| RESAMPLING_METHOD | VARCHAR or NVARCHAR | No default value | Specifies the resampling method for model evaluation or parameter selection.<br>• cv<br>• cv_sha<br>• cv_hyperband<br>• bootstrap<br>• bootstrap_sha<br>• bootstrap_hyperband<br>If no value is specified for this parameter, neither model evaluation nor parameter selection is activated. | |
| RESOURCE | NVARCHAR | No default value | Resource type used in successive-halving and hyperband algorithm for parameter selection.<br>• MAX_ITER | Mandatory and valid only when RESAMPLING_METHOD parameter is cv_sha, cv_hyperband, bootstrap_sha, or bootstrap_hyperband. |
| MAX_RESOURCE | INTEGER | No default value | Maximum resource budget for single hyper-parameter candidate. Valid value must be larger than 0. | Mandatory and valid only when RESOURCE parameter is set. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| MIN_RESOURCE_RATE | DOUBLE | 0.0 | Minimum required resource budget comparing to maximum resource for single hyper-parameter candidate.<br><br>Valid value should be within [0, 1). | Valid only when RESOURCE parameter is set. |
| REDUCTION_RATE | DOUBLE | 3.0 | This parameter specifies the reduction rate of available size of hyper-parameter candidates. For each round, the available parameter candidate size will be divided by value of this parameter. Thus valid value for this parameter must be larger than 1.0. | Valid only when RESOURCE parameter is set. |
| AGGRESSIVE_ELIMINATION | INTEGER | 0 | Boolean value indicates whether aggressive elimination will be performed for successive-halving algorithm. When set to 1, it will eliminate more parameter candidates than expected, which is defined by REDUCTION_RATE parameter; This can enhance the run-time performance but may result in sub-optimal hyper-parameter candidate.<br>• 0: Does not perform aggressive elimination;<br>• 1: Performs aggressive elimination | Valid only when RESAMPLING_METHOD parameter is cv_sha or bootstrap_sha. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| PS_VERBOSE | INTEGER | 1 | Boolean value indicates whether optimal hyper-parameter and all evaluation statistics of related hyper-parameter candidates will be output in Statistics output table.<br>• 0: Does not output evaluation statistics<br>• 1: Outputs evaluation statistics | Valid only when RESAMPLING_METHOD is set. |
| EVALUATION_METRIC | VARCHAR or NVARCHAR | No default value | Specifies the evaluation metric for model evaluation or parameter selection.<br>• RMSE | |
| FOLD_NUM | INTEGER | No default value | Specifies the fold number for a cross validation method. | Mandatory and valid only when RESAMPLING_METHOD is set to cv, cv_sha, or cv_hyperband. |
| REPEAT_TIMES | INTEGER | 1 | Specifies the number of repeat times for resampling. | |
| PARAM_SEARCH_STRATEGY | VARCHAR or NVARCHAR | No default value | Specifies the method to activate parameter selection.<br>• grid;<br>• random. | This parameter is only valid when RESAMPLING_METHOD is not cv_hyperband nor bootstrap_hyperband; Otherwise, this parameter is always considered to have value `random`. |
| RANDOM_SEARCH_TIMES | INTEGER | No default value | Specifies the number of times to randomly select candidate parameters for selection. | Mandatory and valid when PARAM_SEARCH_STRATEGY is set to random. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| SEED | INTEGER | 0 | Specifies the seed for random generation. Use system time when 0 is specified. | |
| TIMEOUT | INTEGER | 0 | Specifies maximum running time for model evaluation or parameter selection, in seconds. No timeout when 0 is specified. | |
| PROGRESS_INDICATOR_ID | VARCHAR or NVARCHAR | No default value | Sets an ID of progress indicator for model evaluation or parameter selection. No progress indicator is active if no value is provided. | |
| LAMBDA_VALUES | VARCHAR or NVARCHAR | No default value | Specifies values of LAMBDA to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| LAMBDA_RANGE | VARCHAR or NVARCHAR | No default value | Specifies the range of LAMBDA to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| ALPHA_VALUES | VARCHAR or NVARCHAR | No default value | Specifies values of ALPHA to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| ALPHA_RANGE | VARCHAR or NVARCHAR | No default value | Specifies the range of parameter ALPHA to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| PRECOMPUTE_LMS_SKETCH | INTEGER | 1 | • 0: Do not perform LMS sketch<br>• 1: Performs LMS sketch | LMS sketch will only perform when RESAMPLING_METHOD is set and Data Row Number is larger than Data Feature Number. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| STABLE_SKETCH_ALG | INTEGER | 1 | When computing LMS sketch, there are two algorithms to choose. One algorithm is more numerical stable than the other one, but is slower.<br>• Do not use stable algorithm<br>• Uses stable algorithm | Only valid when LMS sketch is performed and SPARSE_SKETCH_ALG is 0. |
| SPARSE_SKETCH_ALG | INTEGER | 0 | This is a specific LMS sketch algorithm to cope with sparse data.<br>• 0: Do not use sparse LMS sketch algorithm<br>• Uses sparse LMS sketch algorithm<br>**Note**<br>Use this option with caution. If the provided data are not sparse, turning on this option will cause LMS sketch to be extremely slow! | Only valid when LMS sketch is performed. |

## POL

Mandatory Parameter

| Name | Data Type | Description |
|---|---|---|
| DEGREE | INTEGER | Degree of the polynomial model. |

Optional Parameters

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| DECOMPOSITION | INTEGER | 1 | Matrix factorization type to use. Case-insensitive.<br>• 0: LU Decomposition;<br>• 1: QR Decomposition;<br>• 2: SVD;<br>• 5: Cholesky Decomposition |
| ADJUSTED_R2 | INTEGER | 0 | • 0: Does not output adjusted R square;<br>• 1: Outputs adjusted R square |
| FEATURES | VARCHAR or NVARCHAR | No default value | A string to specify the features that will be processed. The pattern is "`X1`,...,`Xn`", where `Xi` is the corresponding column name in the data table. If this parameter is not specified, all the features will be processed. |

Model Evaluation and Parameter Selection

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| RESAMPLING_METHOD | VARCHAR or NVARCHAR | No default value | Specifies the resampling method for model evaluation or parameter selection.<br>• cv<br>• bootstrap<br>If no value is specified for this parameter, neither model evaluation nor parameter selection is activated. | |
| EVALUATION_METRIC | VARCHAR or NVARCHAR | No default value | Specifies the evaluation metric for model evaluation or parameter selection.<br>• RMSE;<br>• MAE. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| FOLD_NUM | INTEGER | No default value | Specifies the fold number for a cross validation method. | Mandatory and valid only when RESAMPLING_METHOD is set to cv. |
| REPEAT_TIMES | INTEGER | 1 | Specifies the number of repeat times for resampling. | |
| PARAM_SEARCH_STRATEGY | VARCHAR or NVARCHAR | No default value | Specifies the method to activate parameter selection.<br>• grid;<br>• random. | |
| RANDOM_SEARCH_TIMES | INTEGER | No default value | Specifies the number of times to randomly select candidate parameters for selection. | Mandatory and valid when PARAM_SEARCH_STRATEGY is set to random. |
| SEED | INTEGER | 0 | Specifies the seed for random generation. Use system time when 0 is specified. | |
| TIMEOUT | INTEGER | 0 | Specifies maximum running time for model evaluation or parameter selection, in seconds. No timeout when 0 is specified. | |
| PROGRESS_INDICATOR_ID | VARCHAR or NVARCHAR | No default value | Sets an ID of progress indicator for model evaluation or parameter selection. No progress indicator is active if no value is provided. | |
| DEGREE_VALUES | VARCHAR or NVARCHAR | No default value | Specifies values of DEGREE to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| DEGREE_RANGE | VARCHAR or NVARCHAR | No default value | Specifies the range of DEGREE to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |

## DT

Mandatory Parameters

None

Optional Parameters

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| THREAD_RATIO | DOUBLE | -1.0 | The ratio of available threads for multi-thread task:<br>• 0: single thread<br>• 0–1: uses the specified percentage of available threads. PAL uses all available threads if the number is 1.0.<br>• -1: uses all of available threads. | |
| ALLOW_MISSING_LABEL | INTEGER | 1 | Specifies if missing target value is allowed.<br>• 0: Not allowed. If missing target presents, an error occurs.<br>• 1: Allowed. The datum with missing target is removed. | |
| PERCENTAGE | DOUBLE | 1.0 | Specifies the percentage of the input data that will be used to build the tree model. For example, if you set this parameter to 0.7, then 70% of the training data will be used to build the tree model and 30% will be used to prune the tree model. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| MIN_RECORDS_PARENT | INTEGER | 2 | Specifies the stop condition: if the number of records in one node is less than the specified value, the algorithm stops splitting. | |
| MIN_RECORDS_LEAF | INTEGER | 1 | Promises the minimum number of records in a leaf. | |
| MAX_DEPTH | INTEGER | Unlimited | Specifies the stop condition: if the depth of the tree model is greater than the specified value, the algorithm stops splitting. | |
| CATEGORICAL_VARIABLE | VARCHAR or NVARCHAR | Detected from input data | Indicates which variables are treated as categorical. The default behavior is:<br>• STRING: categorical;<br>• INTEGER and DOUBLE: continuous. | Valid only for INTEGER variables; omitted otherwise. |
| SPLIT_THRESHOLD | DOUBLE | 1e-5 | Specifies the stop condition for a node. The reduction of Gini index or relative MSE of the best split is less than this value. The smaller the SPLIT_THRESHOLD value is, the larger a CART tree grows. | |
| USE_SURROGATE | INTEGER | 1 | Indicates whether to use surrogate split when NULL values are encountered.<br>• 0: Does not use surrogate split.<br>• 1: Uses surrogate split. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| MODEL_FORMAT | INTEGER | 1 | Specifies the tree model format for store:<br>• 1: JSON;<br>• 2: PMML. | |

Parameters for Model Evaluation and Parameter Search

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| RESAM-PLING_METHOD | VARCHAR or NVARCHAR | No default value | Specifies the resampling method for model evaluation or parameter selection.<br>• cv;<br>• bootstrap;<br>If no value is specified for this parameter, neither model evaluation nor parameter selection is activated. | |
| EVALUATION_METRIC | VARCHAR or NVARCHAR | No default value | Specifies the evaluation metric for model evaluation or parameter selection.<br>• RMSE;<br>• MAE. | |
| FOLD_NUM | INTEGER | No default value | Specifies the fold number for a cross validation method. | Valid only for cross validation; mandatory for cross validation. |
| REPEAT_TIMES | INTEGER | 1 | The number of repeated times for model evaluation or parameter search. | |
| PARAM_SEARCH_STRATEGY | VARCHAR or NVARCHAR | No default value | The search strategy for parameters:<br>• random: random search;<br>• grid: grid search | If not specified, only model evaluation is carried out. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| RANDOM_SEARCH_TIMES | INTEGER | No default value | Specifies the number of search times for random search. | Mandatory for random search. |
| TIMEOUT | INTEGER | 0 | The time allocated (seconds) for program running. 0 indicates unlimited. | |
| PROGRESS_INDICATOR_ID | VARCHAR or NVARCHAR | No default value | Sets an ID of progress indicator for model evaluation or parameter selection. | No progress indicator is active if no value is provided. |
| MIN_RECORDS_LEAF_VALUES | VARCHAR or NVARCHAR | No default value | The discrete set for the MIN_RECORDS_LEAF candidate. | |
| MIN_RECORDS_LEAF_RANGE | VARCHAR or NVARCHAR | No default value | The range for the MIN_RECORDS_LEAF candidate. | |
| MIN_RECORDS_PARENT_VALUES | VARCHAR or NVARCHAR | No default value | The discrete set for the MIN_RECORDS_PARENT candidate. | |
| MIN_RECORDS_PARENT_RANGE | VARCHAR or NVARCHAR | No default value | The range for the MIN_RECORDS_PARENT candidate. | |
| MAX_DEPTH_VALUES | VARCHAR or NVARCHAR | No default value | The discrete set for the MAX_DEPTH candidate. | |
| MAX_DEPTH_RANGE | VARCHAR, or NVARCHAR | No default value | The range for the MAX_DEPTH candidate. | |
| SPLIT_THRESHOLD_VALUES | VARCHAR or NVARCHAR | No default value | The discrete set for the SPLIT_THRESHOLD candidate. | |
| SPLIT_THRESHOLD_RANGE | VARCHAR or NVARCHAR | No default value | The range for the SPLIT_THRESHOLD candidate. | |

## HGBT

Mandatory Parameters

None.

Optional Parameters

| Name | Data Type | Default Value | Description | Denpendency |
|---|---|---|---|---|
| THREAD_RATIO | DOUBLE | -1.0 | The ratio of available threads for multi-thread task:<br>• 0: single thread<br>• 0–1: uses the specified percentage of available threads. PAL uses all available threads if the number is 1.0.<br>• -1: uses all of available threads. | |
| SEED | INTEGER | 0 | The seed for random number generating.<br>• 0: current time;<br>• Others: the seed. | |
| SPLIT_METHOD | VARCHAR or NVARCHAR | exact | The method to find split point for numeric feature.<br>• exact: the exact method, trying all possible points;<br>• sketch: the sketch method, accounting for the distribution of the sum of hessian;<br>• sampling: samples the split point randomly;<br>• histogram:builds histogram upon data and uses it as split point. | The exact method comparably has the highest test accuracy, but costs more time. The other three methods have relative higher computational efficiency but might lead to lower test accuracy, and are considered to be adopted as the training data set is huge. Valid only for numeric features. |

| Name | Data Type | Default Value | Description | Denpendency |
|---|---|---|---|---|
| SKETCH_EPS | DOUBLE | 0.1 | The epsilon of the sketch method. It indicates that the sum of hessian between two split points is not larger than this value. That is, the number of bins is approximately 1/eps. | The less is this value, the more split points are tried. |
| MAX_BIN_NUM | INTEGER | 256 | The maximum bin number for histogram method. Decreasing this number gains better performance in terms of running time at a cost of accuracy. | Only valid when SPLIT_METHOD is set to histogram. |
| N_ESTIMATORS | INTEGER | 10 | Total iteration number, which is equivalent to the number of trees in the final model. | |
| MAX_DEPTH | INTEGER | 6 | The maximum depth of each tree. | |
| MIN_SAMPLES_LEAF | INTEGER | 1 | The minimum number of data in a leaf. | |
| LEARNING_RATE | DOUBLE | 0.3 | Learning rate of each iteration. Range: (0, 1). | |
| SPLIT_THRESHOLD | DOUBLE | 0.0 | The minimum loss change value to make a split in tree growth (gamma in the equation). | |
| MIN_SAMPLES_WEIGHT_LEAF | DOUBLE | 1.0 | The minimum summation of sample weights(hessian) in the leaf node. | |
| MAX_W_IN_SPLIT | DOUBLE | 0.0 | The maximum weight constraint assigned to each tree node. 0 for no constraint. | |

| Name | Data Type | Default Value | Description | Denpendency |
|---|---|---|---|---|
| SUBSAMPLE | DOUBLE | 1.0 | The sample rate of row (data points). Range: (0, 1). | |
| COL_SUBSAMPLE_SPLIT | DOUBLE | 1.0 | The sample rate of feature set in each split. Range: (0, 1). | |
| COL_SUBSAMPLE_TREE | DOUBLE | 1.0 | The sample rate of feature set in each tree growth. Range: (0, 1). | |
| LAMB | DOUBLE | 1.0 | L2 regularization. Range: [0, 1]. | |
| ALPHA | DOUBLE | 0.0 | L1 regularization. Range: [0, 1] | |
| BASE_SCORE | DOUBLE | 0.0 | Initial prediction score of all instances. Global bias for sufficient number of iterations. | |
| START_FROM_AVERAGE | INTEGER | 0 | Indicates whether to adopt the prior distribution as the initial point. To be specific, use average value if it is a regression problem.<br>• 0: no;<br>• 1: yes. | BASE_SCORE is ignored if this parameter is set to 1. |
| CATEGORICAL_VARIABLE | VARCHAR or NVARCHAR | Detected from input data | Indicates which variables are treated as categorical. The default behavior is:<br>• STRING: categorical;<br>• INTEGER and DOUBLE: continuous. | Valid only for INTEGER variables; omitted otherwise. |

| Name | Data Type | Default Value | Description | Denpendency |
|---|---|---|---|---|
| CALCULATE_IMPORTANCE | INTEGER | 1 | Determines whether to calculate variable importance:<br>• 0: no;<br>• 1: yes. | |
| OBJ_FUNC | INTEGER | • 0 for regression problem.<br>• 5 for binary classification<br>• 7 for multiple classification | For regression:<br>• 0: Squared error<br>• 1: Squared log error<br>• 2: Pseudo Huber error<br>• 3: Gamma<br>• 4: Tweedie<br>For binary classification:<br>• 5: Logistic<br>• 6: Hinge<br>For multiple classification:<br>• 7: Softmax | |
| TWEEDIE_POWER | DOUBLE | 1.5 | For Tweedie object function; must be in range [1.0, 2.0]. | Only valid when OBJ_FUNC equals 4. |
| REPLACE_MISSING | INTEGER | 1 | Replace missing value by another value in the feature.<br>If it is a continuous feature, the value is the mean value;<br>If it is a categorical feature, the value is the most frequent one.<br>• 0: no<br>• 1: yes | |

| Name | Data Type | Default Value | Description | Denpendency |
|---|---|---|---|---|
| DEFAULT_MISSING_DIRECTION | INTEGER | 1 | Define the default direction where missing value will go to while splitting.<br>• 0: left<br>• 1: right | |

Early Stop

See Early Stop of Hybrid Gradient Boosting Tree [page 231] for details.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| VALIDATION_SET_RATE | DOUBLE | 0.0 | Specify the rate of validation set that be sampled from the data set. If 0.0 is set, then no early stop will be applied. | |
| STRATIFIED_VALIDATION_SET | INTEGER | 0 | Indicates weather or not to apply the stratified method while sampling validation set. | Valid only when VALIDATION_SET_RATE is greater than 0. |
| TOLERANT_ITER_NUM | INTEGER | 10 | Indicates how many consecutive deteriorated iterations are observed before applying early stop. | Valid only when VALIDATION_SET_RATE is greater than 0. |

Feature Grouping

See Feature Grouping for Hybrid Gradient Boosting Tree [page 232] for details.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| FEATURE_GROUPING | INTEGER | 1 | Applies feature grouping by grouping sparse features that only contain one significant value in each row.<br>• 0: no<br>• 1: yes | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| TOLERANT_RATE | DOUBLE | 0.0001 | While applying feature grouping, features are still merged when there are rows containing more than one significant value. This parameter specifies the rate of such rows that are allowed. | Only available when FEATURE_GROUPING is set to 1. |
| FG_MIN_ZERO_RATE | DOUBLE | 0.5 | Specifies the minimum zero rate that is used to indicate sparse columns, only which may take part in feature grouping. | Only available when FEATURE_GROUPING is set to 1. |

Model Evaluation and Parameter Selection

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| RESAMPLING_METHOD | VARCHAR or NVARCHAR | No default value | Specifies the resampling method for model evaluation or parameter selection.<br>• cv<br>• cv_sha<br>• cv_hyperband<br>• bootstrap<br>• bootstrap_sha<br>• bootstrap_hyperband<br>If no value is specified for this parameter, neither model evaluation nor parameter selection is activated. | Options like *_sha and *_hyperband can only be used in parameter selection, can not be used in model evaluation. |
| EVALUATION_METRIC | VARCHAR or NVARCHAR | No default value | Specify evaluation metric for model evaluation or parameter selection:<br>• RMSE;<br>• MAE | It is mandatory if RESAMPLING_METHOD is set. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| REF_METRIC | VARCHAR or NVARCHAR | No default value | The metric(s) for reference. Could be of multi-lines, for example, more than one REF_METRIC. | The metric that is identical to EVALUATION_METRIC will be omitted. |
| FOLD_NUM | INTEGER | No default value | Specifies the fold number for a cross validation method. | Mandatory and valid only when RESAMPLING_METHOD is set to cv. |
| REPEAT_TIMES | INTEGER | 1 | Specifies the number of repeat times for resampling. | |
| PARAM_SEARCH_STRATEGY | VARCHAR or NVARCHAR | No default value | Specify value to this parameter to active parameter selection.<br>• grid;<br>• random. | If this parameter is not set, then only model evaluation is activated. |
| RANDOM_SEARCH_TIMES | INTEGER | No default value | Specifies the number of times to randomly select candidate parameters for selection. | Mandatory and valid when PARAM_SEARCH_STRATEGY is set to random. |
| REDUCTION_RATE | DOUBLE | 3.0 | Specifies reduction rate in SHA or Hyperband method. | Only valid when RESAMPLING_METHOD is specified as *_sha or *_hyperband. |
| RESOURCE | NVARCHAR | No default value | Specifies resource definition for SHA or Hyperband method. If nothing is specified, size of data set is used as resource by default.<br>• Not specified, size of data set is used as resource by default.<br>• 'N_ESTIMATORS' | Only valid when RESAMPLING_METHOD is specified as *_sha or *_hyperband. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| MIN_RESOURCE_RATE | DOUBLE | Algorithm specified | Specifies the minimum resource rate that should be used in SHA or Hyperband iteration. | Only valid when RESAMPLING_METHOD is specified as *_sha or *_hyperband.<br>Default value 0.0 for default RESOURCE and default value 1/MAX_RESOURCE for RESOURCE = N_ESTIMATORS. |
| MAX_RESOURCE | INTEGER | No default value | Specifies the maximum number of N_ESTIMATORS that should be used in SHA or Hyperband method. | Only valid when RESAMPLING_METHOD is specified as *_sha or *_hyperband. It is valid and mandatory if RESOURCE is set. |
| AGGRESSIVE_ELIMINATION | INTEGER | 0 | Specifies whether to apply aggressive elimination while using SHA method. Aggressive elimination happens when the data size and parameters size to be searched does not match and there are still bunch of parameters to be searched while data size reaches its upper limits. If aggressive elimination is applied, lower bound of limit of data size will be used multiple times first to reduce number of parameters.<br>• 0: doe not apply aggressive elimination<br>• 1: applies aggressive elimination | Only valid when RESAMPLING_METHOD is specified as *_sha. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| TIMEOUT | INTEGER | 0 | Specifies maximum running time for model evaluation or parameter selection, in seconds. No timeout when 0 is specified. | |
| PROGRESS_INDICATOR_ID | VARCHAR or NVARCHAR | No default value | Sets an ID of progress indicator for model evaluation or parameter selection. No progress indicator is active if no value is provided. | |
| N_ESTIMATORS_VALUES | VARCHAR or NVARCHAR | No default value | Specifies values of parameter N_ESTIMATORS to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| N_ESTIMATORS_RANGE | VARCHAR or NVARCHAR | No default value | Specifies range of parameter N_ESTIMATORS to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| MAX_DEPTH_VALUES | VARCHAR or NVARCHAR | No default value | Specifies values of parameter MAX_DEPTH to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| MAX_DEPTH_RANGE | VARCHAR or NVARCHAR | No default value | Specifies range of parameter MAX_DEPTH to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| MIN_SAMPLES_LEAF_VALUES | VARCHAR or NVARCHAR | No default value | Specifies values of parameter MIN_SAMPLES_LEAF to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| MIN_SAMPLES_LEAF_RANGE | VARCHAR or NVARCHAR | No default value | Specifies range of parameter MIN_SAMPLES_LEAF to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| LEARNING_RATE_VALUES | VARCHAR or NVARCHAR | No default value | Specifies values of parameter LEARNING_RATE to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| LEARNING_RATE_RANGE | VARCHAR or NVARCHAR | No default value | Specifies range of parameter LEARNING_RATE to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| SPLIT_THRESHOLD_VALUES | VARCHAR or NVARCHAR | No default value | Specifies values of parameter SPLIT_THRESHOLD to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| SPLIT_THRESHOLD_RANGE | VARCHAR or NVARCHAR | No default value | Specifies range of parameter SPLIT_THRESHOLD to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| MIN_SAMPLES_WEIGHT_LEAF_VALUES | VARCHAR or NVARCHAR | No default value | Specifies values of parameter MIN_SAMPLES_WEIGHT_LEAF to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| MIN_SAMPLES_WEIGHT_LEAF_RANGE | VARCHAR or NVARCHAR | No default value | Specifies range of parameter MIN_SAMPLES_WEIGHT_LEAF to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| MAX_W_IN_SPLIT_VALUES | VARCHAR or NVARCHAR | No default value | Specifies values of parameter MAX_W_IN_SPLIT to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| MAX_W_IN_SPLIT_RANGE | VARCHAR or NVARCHAR | No default value | Specifies range of parameter MAX_W_IN_SPLIT to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| COL_SUBSAMPLE_SPLIT_VALUES | VARCHAR or NVARCHAR | No default value | Specifies values of parameter COL_SUBSAMPLE_SPLIT to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| COL_SUBSAMPLE_SPLIT_RANGE | VARCHAR or NVARCHAR | No default value | Specifies range of parameter COL_SUBSAMPLE_SPLIT to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| COL_SUBSAMPLE_TREE_VALUES | VARCHAR or NVARCHAR | No default value | Specifies values of parameter COL_SUBSAMPLE_TREE to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| COL_SUBSAMPLE_TREE_RANGE | VARCHAR or NVARCHAR | No default value | Specifies range of parameter COL_SUBSAMPLE_TREE to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| LAMB_VALUES | VARCHAR or NVARCHAR | No default value | Specifies values of parameter LAMB to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| LAMB_RANGE | VARCHAR or NVARCHAR | No default value | Specifies range of parameter LAMB to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| ALPHA_VALUES | VARCHAR or NVARCHAR | No default value | Specifies values of parameter ALPHA to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| ALPHA_RANGE | VARCHAR or NVARCHAR | No default value | Specifies range of parameter ALPHA to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |

ⓘ Note

- To active parameter selection or model evaluation, parameter RESAMPLING_METHOD must be set. Then, to active parameter selection, parameter PARAM_SEARCH_STRATEGY must be set, or only model evaluation is carried out. For parameters like *_VALUES, its format is {`val1`, `val2`, ..., `valn`}. For parameters like *_RANGE, its format is [RANGE_START, STEP, RANGE_END]. STEP can be omitted if using random search, in which case its format is [RANGE_START ,, RANGE_END]. Commas are still needed.
- To make this function back compatible, old cross validation fashion is still supported. In this case, RESAMPLING_METHOD is not set and FOLD_NUM is set to a value greater than 1. Parameter EVALUATION_METRIC and REF_METRIC has default value RMSE for regression. And parameters for searching purpose are of name RANGE_* and format like [RANGE_START, RANGE_END, NUM], of which the first and second ones are of integer or double types, whereas the last one is of integer type (not less than 2), indicating that the number of candidate values by equal space constrained by range of [RANGE_START, RANGE_END].

## MLP

Mandatory Parameters

| Name | Data Type | Description |
| --- | --- | --- |
| ACTIVATION | INTEGER | Active function code for the hidden layer.<br>• TANH = 1;<br>• LINEAR = 2;<br>• SIGMOID_ASYMMETRIC = 3;<br>• SIGMOID_SYMMETRIC = 4;<br>• GAUSSIAN_ASYMMETRIC = 5;<br>• GAUSSIAN_SYMMETRIC = 6;<br>• ELLIOT_ASYMMETRIC = 7;<br>• ELLIOT_SYMMETRIC = 8;<br>• SIN_ASYMMETRIC = 9;<br>• SIN_SYMMETRIC = 10;<br>• COS_ASYMMETRIC = 11;<br>• COS_SYMMETRIC = 12;<br>• ReLU = 13. |
| OUTPUT_ACTIVATION | INTEGER | Active function code for the output layer. The code is the same as ACTIVATION. |
| HIDDEN_LAYER_SIZE | VARCHAR or NVARCHAR | Specifies the size of each hidden layer in the format of "2, 3, 4". The value 0 will be ignored, for example, "2, 0, 3" is equal to "2, 3". |

Optional Parameters

| Name | Data Type | Default Value | Description | Dependency |
| --- | --- | --- | --- | --- |
| THREAD_RATIO | DOUBLE | -1.0 | The ratio of available threads.<br>• 0: single thread<br>• 0–1: uses the specified percentage of available threads. PAL uses all available threads if the number is 1.0.<br>• -1: uses all of available threads. | |
| MAX_ITER | INTEGER | 100 | Maximum number of iterations. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| TRAINING_STYLE | INTEGER | 1 | Specifies the training style:<br>• 0: Batch;<br>• 1: Stochastic. | |
| LEARNING_RATE | DOUBLE | No default value | Specifies the learning rate. | Mandatory and valid only when TRAINING_STYLE = 1. |
| MOMENTUM | DOUBLE | No default value | Specifies the momentum factor. | Mandatory and valid only when TRAINING_STYLE = 1. |
| BATCH_SIZE | INTEGER | 1 | Specifies the size of mini batch. | Valid only when TRAINING_STYLE = 1. |
| NORMALIZATION | INTEGER | 0 | Specifies the normalization type:<br>• 0: None;<br>• 1: Z-transform;<br>• 2: Scalar. | |
| WEIGHT_INIT | INTEGER | 0 | Specifies the weight initial value:<br>• 0: All zeros;<br>• 1: Normal distribution;<br>• 2: Uniform distribution in range (0, 1);<br>• 3: Variance scale normal;<br>• 4: Variance scale uniform. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| CATEGORICAL_VARIABLE | VARCHAR or NVARCHAR | No default value | Indicates whether or not a column data is actually corresponding to a category variable even the data type of this column is INTEGER. By default, 'VARCHAR' or 'NVARCHAR' is category variable, and 'INTEGER' or 'DOUBLE' is continuous variable. | |
| ONEHOT_MIN_FREQUENCY | INTEGER | 1 | Specifies the minimum frequency below which a category is considered infrequent. | |
| ONEHOT_MAX_CATEGORIES | INTEGER | 0 | Specifies an upper limit of the number of output features for each input feature. ONEHOT_MAX_CATEGORIES includes the feature that combines infrequent categories. | |

Model Evaluation and Parameter Selection

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| RESAMPLING_METHOD | VARCHAR or NVARCHAR | No default value | Specifies the resampling method for model evaluation or parameter selection.<br>• cv;<br>• bootstrap;<br>If no value is specified for this parameter, neither model evaluation nor parameter selection is activated. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| EVALUATION_METRIC | VARCHAR or NVARCHAR | No default value | Specifies the evaluation metric for model evaluation or parameter selection.<br>• RMSE. | |
| FOLD_NUM | INTEGER | No default value | Specifies the fold number for a cross validation method. | Mandatory and valid only when RESAMPLING_METHOD is set to cv. |
| REPEAT_TIMES | INTEGER | 1 | Specifies the number of repeat times for resampling. | |
| PARAM_SEARCH_STRATEGY | VARCHAR or NVARCHAR | No default value | Specifies the method to activate parameter selection.<br>• grid;<br>• random. | |
| RANDOM_SEARCH_TIMES | INTEGER | No default value | Specifies the number of times to randomly select candidate parameters for selection. | Mandatory and valid when PARAM_SEARCH_STRATEGY is set to random. |
| SEED | INTEGER | 0 | Specifies the seed for random generation. Use system time when 0 is specified. | |
| TIMEOUT | INTEGER | 0 | Specifies maximum running time for model evaluation or parameter selection, in seconds. No timeout when 0 is specified. | |
| PROGRESS_INDICATOR_ID | VARCHAR or NVARCHAR | No default value | Sets an ID of progress indicator for model evaluation or parameter selection. No progress indicator is active if no value is provided. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| ACTIVATION_OPTIONS | VARCHAR or NVARCHAR | No default value | Specifies values of ACTIVATION to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| OUTPUT_ACTIVATION_OPTIONS | VARCHAR or NVARCHAR | No default value | Specifies values of OUTPUT_ACTIVATION to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| HIDDEN_LAYER_SIZE_OPTIONS | VARCHAR or NVARCHAR | No default value | Specifies values of HIDDEN_LAYER_SIZE to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| LEARNING_RATE_VALUES | VARCHAR or NVARCHAR | No default value | Specifies values of LEARNING_RATE to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| LEARNING_RATE_RANGE | VARCHAR or NVARCHAR | No default value | Specifies range of LEARNING_RATE to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| MOMENTUM_VALUES | VARCHAR or NVARCHAR | No default value | Specifies values of MOMENTUM to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| MOMENTUM_RANGE | VARCHAR or NVARCHAR | No default value | Specifies range of MOMENTUM to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| BATCH_SIZE_VALUES | VARCHAR or NVARCHAR | No default value | Specifies values of BATCH_SIZE to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| BATCH_SIZE_RANGE | VARCHAR or NVARCHAR | No default value | Specifies range of BATCH_SIZE to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |

## RDT

Mandatory Parameters

None

Optional Parameters

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| THREAD_RATIO | DOUBLE | -1.0 | The ratio of available threads.<br>• 0: single thread<br>• 0–1: uses the specified percentage of available threads. PAL uses all available threads if the number is 1.0.<br>• -1: uses all of available threads. | |
| N_ESTIMATORS | INTEGER | 100 | Specifies the number of trees to grow. | |
| MAX_FEATURES | INTEGER | p/3, where p is the number of valid independent variables. | Specifies the number of randomly selected variables for splitting. | Should not be larger than the valid number of features. |
| ALLOW_MISSING_LABEL | INTEGER | 1 | Specifies if missing target value is allowed.<br>• 0: Not allowed. An error occurs if missing dependent presents.<br>• 1: Allowed. The datum with missing dependent is removed. | |
| CATEGORICAL_VARIABLE | VARCHAR or NVARCHAR | Detected from input data | Indicates which variables are treated as categorical. The default behavior is:<br>• STRING: categorical;<br>• NTEGER and DOUBLE: continuous. | VALID only for INTEGER variables; omitted otherwise. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| SEED | INTEGER | 0 | Specifies the seed for random number generator.<br>• 0: Uses the current time (in second) as seed.<br>• Others: Uses the specified value as seed. | |
| MIN_SAMPLES_LEAF | INTEGER | 5 | Specifies the minimum number of records in a leaf. | |
| SPLIT_THRESHOLD | DOUBLE | 1e-5 | Specifies the stop condition: if the improvement value of the best split is less than this value, the tree stops growing. | |
| MAX_DEPTH | INTEGER | 56 | The maximum depth of a tree. | If MAX_DEPTH is set to -1, there are no restrictions on the depth of the tree. Otherwise, MAX_DEPTH cannot be larger than 56. |
| SAMPLE_FRACTION | DOUBLE | 1 | The fraction of data used for training. Assume there are n pieces of data, SAMPLE_FRACTION is `r`, then `n*r` data is selected for training. | In the range of [0,1]. If n*r is less than the number of classes, the sample size is restored to the number of classes. |
| COMPRESSION | INTEGER | 0 | Specifies if the model is stored in compressed format.<br>• 0: No;<br>• 1: Yes. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| MAX_BITS | INTEGER | 12 | Maximal number of bits to quantize continuous features. Equivalent to use $2^{MAX_BITS}$ bins. | Must be less than 31. |
| QUANTIZE_RATE | DOUBLE | 0.005 | Quantizes a categorical feature if the largest class frequency of the feature is less than QUANTIZE_RATE. | Valid only when COMPRESSION is equal to 1. |
| OUTPUT_LEAF_VALUES | INTEGER | 0 | Controls whether to save target values in each leaf node.<br>• 0: Doesn't output target values in each leaf node;<br>• 1: Outputs target values in each leaf node. | Valid only when JSON_EXPORT = 1 or COMPRESSION = 1. |
| FITTINGS_QUANTIZATION | INTEGER | 0 | Indicates whether to quantize fitting values.<br>• 0: No;<br>• 1: Yes. | Valid only for regression when COMPRESSION is equal to 1. |
| MODEL_FORMAT | INTEGER | 1 | Specifies the tree model format for store:<br>• 1: JSON<br>• 2: PMML | If both MODEL_FORMAT and COMPRESSION are specified, COMPRESSION takes precedence. |

## SVM

Mandatory Parameter

None

Optional Parameters

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| KERNEL_TYPE | INTEGER | 2 | Kernel type:<br>• 0: LINEAR KERNEL;<br>• 1: POLY KERNEL;<br>• 2: RBF KERNEL;<br>• 3: SIGMOID KERNEL. | |
| THREAD_RATIO | DOUBLE | -1.0 | The ratio of available threads.<br>• 0: single thread<br>• 0–1: uses the specified percentage of available threads. PAL uses all available threads if the number is 1.0.<br>• -1: uses all of available threads. | |
| POLY_DEGREE | INTEGER | 3 | Coefficient for the PLOY KERNEL type.<br>Value range: ≥ 1 | Only valid when KERNEL_TYPE is 1. |
| RBF_GAMMA | DOUBLE | 1.0/number of features in the dataset | Coefficient for the RBF KERNEL type.<br>Value range: > 0 | Only valid when KERNEL_TYPE is 2. |
| COEF_LIN | DOUBLE | 0.0 | Coefficient for the POLY/SIGMOID KERNEL type. | Only valid when KERNEL_TYPE is 1 or 3. |
| COEF_CONST | DOUBLE | 0.0 | Coefficient for the POLY/SIGMOID KERNEL type. | Only valid when KERNEL_TYPE is 1 or 3. |
| SVM_C | DOUBLE | 100.0 | Trade-off between training error and margin.<br>Value range: > 0 | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| REGRESSION_EPS | DOUBLE | 0.1 | Epsilon width of tube for regression.<br>Value range: > 0 | |
| SCALE_INFO | INTEGER | 1 | • 0: No scale.<br>• 1: Standardization. The algorithm transforms the data to have zero mean and unit variance. The general formula is given as: $x_{new}=\frac{x-\mu}{\sigma}$, where x is the origin feature vector, μ is the mean of that feature vector, and σ is its standard deviation.<br>• 2: Rescaling. The algorithm rescales the range of the features to scale the range in [-1,1]. The general formula is given as: $x_{new}=-1+\frac{x-\min(x)}{\max(x)-\min(x)}*2$, where x is the origin feature vector. | |
| SCALE_LABEL | INTEGER | 1 | Controls whether to standardize the label.<br>• 0: No;<br>• 1: Yes. | Only valid when SCALE_INFO is 1. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| SHRINK | INTEGER | 1 | Decides whether to use shrink strategy or not:<br>• 0: Does not use shrink strategy.<br>• 1: Uses shrink strategy. | Using shrink strategy may accelerate the training process. |
| HANDLE_MISSING | INTEGER | 1 | Whether to handle missing values:<br>• 0: No;<br>• 1: Yes. | |
| CATEGORICAL_VARIABLE | VARCHAR or NVARCHAR | No default value | Indicates whether or not a column data is actually corresponding to a category variable even the data type of this column is INTEGER. By default, 'VARCHAR' or 'NVARCHAR' is category variable, and 'INTEGER' or 'DOUBLE' is continuous variable.. | |
| CATEGORY_WEIGHT | DOUBLE | 0.707 | Represents the weight of category attributes ($\gamma$). The value must be greater than 0. | |
| TOL | DOUBLE | 0.001 | Specifies the error tolerance in the training process. The value must be greater than 0. | |
| COMPRESSION | INTEGER | 1 | Specifies if the model is stored in compressed format.<br>• 0 : no;<br>• 1: yes. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| MAX_BITS | INTEGER | 12 | Maximal number of bits to quantize continuous features. Equivalent to use $2^{MAX_BITS}$ bins. | Must be less than 31. |
| MAX_QUANTIZATION_ITER | DOUBLE | 1000 | The maximum iteration steps for quantization. | Valid only when COMPRESSION is equal to 1. |
| ONEHOT_MIN_FREQUENCY | INTEGER | 1 | Specifies the minimum frequency below which a category is considered infrequent. | |
| ONEHOT_MAX_CATEGORIES | INTEGER | 0 | Specifies an upper limit of the number of output features for each input feature. ONEHOT_MAX_CATEGORIES includes the feature that combines infrequent categories. | |

Model Evaluation and Parameter Selection

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| RESAMPLING_METHOD | VARCHAR or NVARCHAR | No default value | Specifies the resampling method for model evaluation or parameter selection.<br>• cv<br>• cv_sha<br>• cv_hyperband<br>• bootstrap<br>• bootstrap_sha<br>• bootstrap_hyperband<br>If no value is specified for this parameter, neither model evaluation nor parameter selection is activated. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| EVALUATION_METRIC | VARCHAR or NVARCHAR | No default value | Specifies the evaluation metric for model evaluation or parameter selection.<br>• RMSE. | |
| FOLD_NUM | INTEGER | No default value | Specifies the fold number for a cross validation method. | Mandatory and valid only when RESAMPLING_METHOD is set to cv. |
| REPEAT_TIMES | INTEGER | 1 | Specifies the number of repeat times for resampling. | |
| PARAM_SEARCH_STRATEGY | VARCHAR or NVARCHAR | No default value | Specifies the method to activate parameter selection.<br>• grid;<br>• random. | When *_sha is specified in RESAMPLING_METHOD, PARAM_SEARCH_STRATEGY is mandatory. When *_hyperband is specified in RESAMPLING_METHOD, PARAM_SEARCH_STRATEGY is set to random whatever other value is provided. |
| RANDOM_SEARCH_TIMES | INTEGER | No default value | Specifies the number of times to randomly select candidate parameters for selection. | Mandatory and valid when PARAM_SEARCH_STRATEGY is set to random. |
| REDUCTION_RATE | DOUBLE | 3.0 | Specifies reduction rate in SHA or Hyperband method. | Only valid when RESAMPLING_METHOD is specified as *_sha or *_hyperband. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| AGGRESSIVE_ELIMINATION | INTEGER | 0 | Specifies whether to apply aggressive elimination while using SHA method. Aggressive elimination happens when the data size and parameters size to be searched does not match and there are still bunch of parameters to be searched while data size reaches its upper limits. If aggressive elimination is applied, lower bound of limit of data size will be used multiple times first to reduce number of parameters.<br>• 0: doe not apply aggressive elimination<br>• 1: applies aggressive elimination | Only valid when RESAMPLING_METHOD is specified as *_sha. |
| SEED | INTEGER | 0 | Specifies the seed for random generation. Use system time when 0 is specified. | |
| TIMEOUT | INTEGER | 0 | Specifies maximum running time for model evaluation or parameter selection, in seconds. No timeout when 0 is specified. | |
| PROGRESS_INDICATOR_ID | VARCHAR or NVARCHAR | No default value | Sets an ID of progress indicator for model evaluation or parameter selection. No progress indicator is active if no value is provided. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| SVM_C_VALUES | VARCHAR or NVARCHAR | No default value | Specifies values of SVM_C to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| SVM_C_RANGE | VARCHAR or NVARCHAR | No default value | Specifies range of SVM_C to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| RBF_GAMMA_VALUES | VARCHAR or NVARCHAR | No default value | Specifies values of RBF_GAMMA to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| RBF_GAMMA_RANGE | VARCHAR or NVARCHAR | No default value | Specifies range of RBF_GAMMA to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| POLY_DEGREE_VALUES | VARCHAR or NVARCHAR | No default value | Specifies values of POLY_DEGREE to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| POLY_DEGREE_RANGE | VARCHAR or NVARCHAR | No default value | Specifies range of POLY_DEGREE to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| COEF_LIN_VALUES | VARCHAR or NVARCHAR | No default value | Specifies values of COEF_LIN to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| COEF_LIN_RANGE | VARCHAR or NVARCHAR | No default value | Specifies range of COEF_LIN to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| COEF_CONST_VALUES | VARCHAR or NVARCHAR | No default value | Specifies values of COEF_CONST to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| COEF_CONST_RANGE | VARCHAR or NVARCHAR | No default value | Specifies range of COEF_CONST to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |

# _SYS_AFL.PAL_UNIFIED_REGRESSION_PREDICT

## UNIFIED REGRESSION PREDICT

Mandatory parameters

| Name | Data Type | Description | Constraint |
| --- | --- | --- | --- |
| FUNCTION | VARCHAR or NVARCHAR | The algorithm name, including:<br>• GEO;<br>• LOG;<br>• EXP;<br>• GLM;<br>• MLR;<br>• POL;<br>• DT;<br>• HGBT;<br>• MLP;<br>• RDT;<br>• SVM | The parameter must be same with the FUCTION used to train the model. |

Optional parameters

| Name | Data Type | Default Value | Description | Constraint |
| --- | --- | --- | --- | --- |
| THREAD_RATIO | DOUBLE | -1.0 | The ratio of available threads for multi-thread task:<br>• 0: single thread<br>• 0–1: uses the specified percentage of available threads. PAL uses all available threads if the number is 1.0.<br>• -1: uses all of available threads. | Excludes: GLM |
| HANDLE_MISSING_VALUE | INTEGER | 0 | Whether to handle missing values:<br>• 0: No<br>• 1: Yes | |

| Name | Data Type | Default Value | Description | Constraint |
|---|---|---|---|---|
| IMPUTATION_TYPE | INTEGER | 1 | Chooses the overall default imputation type for all columns:<br>• 0: NONE<br>• 1: MODE_MEAN<br>• 2: MODE_MEDIAN<br>• 3: MODE_ALL-ZERO<br>• 4: MODE_ALS<br>• 5: DELETE<br>NONE and DELETE: set all columns to imputation type NONE/DELETE.<br>MODE_MEAN, MODE_MEDIAN, MODE_ALLZERO, and MODE_ALS: set categorical columns to the first imputation type (the type before “”); set numerical columns to the second imputation type (the type after “”). ALLZERO means to fill all missing values in numerical columns with 0. | Only valid when HANDLE_MISSING_VALUE is 1. |

| Name | Data Type | Default Value | Description | Constraint |
|---|---|---|---|---|
| `<column name>`_IMPUTATION_TYPE | INTEGER | No default value | Sets imputation type of a specified column. This value overwrites the default imputation type deduced by the IMPUTATION_TYPE parameter:<br>• 0: NONE<br>• 1: DELETE<br>• 100: MODE<br>• 101: SPECIFIED_CATEGORICAL_VALUE<br>• 200: MEAN<br>• 201: MEDIAN<br>• 203:SPECIFIED_NUMERICAL_VALUE<br>• 204: ALS<br>For type 101 (SPECIFIED_CATEGORICAL_VALUE), use the following syntax to specify a categorical value to be filled in a column V0:<br>INSERT INTO #PAL_PARAMETER_TBL VALUES('V0_IMPUTATION_TYPE', 101, NULL, '`<CATEGORICAL_VALUE>`');<br>For type 203 (SPECIFIED_NUMERICAL_VALUE), use the following syntax to specify a numerical value to be filled in a column V1:<br>INSERT INTO #PAL_PARAMETER_TBL VALUES('V1_IMPUTA- | Only valid when HANDLE_MISSING_VALUE is 1. |

| Name | Data Type | Default Value | Description | Constraint |
|---|---|---|---|---|
| | | | TION_TYPE', 203, `<NUMERICAL_VALUE>`, NULL);<br><br>**Note**<br>A column of integer type can be set to type 101 (SPECIFIED_CATEGORICAL_VALUE) or type 203 (SPECIFIED_NUMERICAL_VALUE):<br>• when setting to type 101 (SPECIFIED_CATEGORICAL_VALUE), the string must contain a valid integer value, otherwise an error will be thrown;<br>• when setting to type 203 (SPECIFIED_NUMERICAL_VALUE), the double value will be converted to integer. | |

| Name | Data Type | Default Value | Description | Constraint |
|---|---|---|---|---|
| ALS_FACTOR_NUMBER | INTEGER | 3 | Length of factor vectors, that is, the f in ALS model.<br>**Note**<br>ALS_FACTOR_NUMBER should be less than the number of numerical columns to get a meaningful result. | |
| ALS_REGULARIZATION | DOUBLE | 1e-2 | L2 regularization of the factors, that is, the λ in ALS model. | |
| ALS_MAX_ITERATION | INTEGER | 20 | Maximum number of iterations when training ALS model. | |
| ALS_SEED | INTEGER | 0 | Specifies the seed of the random number generator used in the training of ALS model:<br>• 0: Uses the current time as the seed<br>• Others: Uses the specified value as the seed. | |
| ALS_EXIT_THRESHOLD | DOUBLE | 0.0 | The training of ALS model exits if the value of cost function is not decreased more than the value of ALS_EXIT_THRESHOLD since the last check.<br>If ALS_EXIT_THRESHOLD is set to 0, there is no check and the training only exits on reaching the maximum number of iterations. Evaluations of cost function require additional calculations. You can set this parameter to 0 to avoid it. | |

| Name | Data Type | Default Value | Description | Constraint |
|---|---|---|---|---|
| ALS_EXIT_INTERVAL | INTEGER | 5 | During the training of ALS model, it will calculate cost function and check the exit criteria every `<ALS_EXIT_INTERVAL>` iterations. Evaluations of cost function require additional calculations. | |
| ALS_LINEAR_SYSTEM_SOLVER | INTEGER | 0 | • 0: cholesky solver<br>• 1: cg solver (recommended when ALS_FACTOR_NUMBER is large) | |
| ALS_CG_MAX_ITERATION | INTEGER | 3 | Maximum number of iteration of cg solver. | |
| ALS_CENTERING | INTEGER | 1 | Whether to center the data by column before training ALS model.<br>• 0: no centering<br>• 1: centering. | |
| ALS_SCALING | INTEGER | 1 | Whether to scale the data by column before training ALS model.<br>• 0: no scaling<br>• 1: scaling | |

## GLM

Mandatory parameters

None

Optional parameters

| Name | Data Type | Default Value | Description | Constraint |
|---|---|---|---|---|
| TYPE | VARCHAR or NVARCHAR | response | The prediction type:<br>• response: Predicts response μ;<br>• link: Predicts linear predictor η. | |

| Name | Data Type | Default Value | Description | Constraint |
|---|---|---|---|---|
| SIGNIFICANCE_LEVEL | DOUBLE | 0.05 | The significance level for the confidence interval and prediction interval. | Valid only for IRLS. |
| HANDLE_MISSING | INTEGER | 2 | Specify how to handle missing covariates.<br>• 1: Remove missing rows;<br>• 2: Replace missing with 0. | |

### RDT

Mandatory parameters

None

Optional parameters

| Name | Data Type | Default Value | Description | Constraint |
|---|---|---|---|---|
| BLOCK_SIZE | INTEGER | 0 | Specifies the number of data loaded per time during scoring.<br>• 0: load all data once;<br>• Others: the specified number. | This parameter is for reducing memory consumption, especially as the predict data is huge, or it consists of a large number of missing independent variables. However, you might lose some efficiency. |

## _SYS_AFL.PAL_UNIFIED_REGRESSION_SCORE

### UNIFIED REGRESSION SCORE

Mandatory parameters

| Name | Data Type | Description | Constraint |
| --- | --- | --- | --- |
| FUNCTION | VARCHAR or NVARCHAR | The algorithm name, including:<br>• GEO;<br>• LOG;<br>• EXP;<br>• GLM;<br>• MLR;<br>• POL;<br>• DT;<br>• HGBT;<br>• MLP;<br>• RDT;<br>• SVM | The parameter must be the same as the FUCTION used to train the model. |

Optional parameters

| Name | Data Type | Default Value | Description | Constraint |
| --- | --- | --- | --- | --- |
| THREAD_RATIO | DOUBLE | -1.0 | The ratio of available threads for multi-thread task:<br>• 0: single thread<br>• 0–1: uses the specified percentage of available threads. PAL uses all available threads if the number is 1.0.<br>• -1: uses all of available threads. | Excludes: GLM |
| IGNORE_ZERO | INTEGER | 0 | Valid values are:<br>• 0: Uses zero values in the actual dataset when calculating MPE or MAPE;<br>• 1: Ignores zero values in the actual dataset when calculating MPE or MAPE. | Only valid when MEASURE_NAME is MPE or MAPE. |

| Name | Data Type | Default Value | Description | Constraint |
|---|---|---|---|---|
| LABEL | VARCHAR or NVARCHAR | No default value | Column name in the data table used as dependent variable. | The setting of the parameter should be same with the one in train. |
| HANDLE_MISSING_VALUE | INTEGER | 0 | Whether to handle missing values:<br>• 0: No<br>• 1: Yes | |
| IMPUTATION_TYPE | INTEGER | 1 | Chooses the overall default imputation type for all columns:<br>• 0: NONE<br>• 1: MODE_MEAN<br>• 2: MODE_MEDIAN<br>• 3: MODE_ALLZERO<br>• 4: MODE_ALS<br>• 5: DELETE<br>NONE and DELETE: set all columns to imputation type NONE/DELETE.<br>MODE_MEAN, MODE_MEDIAN, MODE_ALLZERO, and MODE_ALS: set categorical columns to the first imputation type (the type before “”); set numerical columns to the second imputation type (the type after “”). ALLZERO means to fill all missing values in numerical columns with 0. | Only valid when HANDLE_MISSING_VALUE is 1. |

| Name | Data Type | Default Value | Description | Constraint |
|---|---|---|---|---|
| `<column name>`_IMPUTATION_TYPE | INTEGER | No default value | Sets imputation type of a specified column. This value overwrites the default imputation type deduced by the IMPUTATION_TYPE parameter:<br>• 0: NONE<br>• 1: DELETE<br>• 100: MODE<br>• 101: SPECIFIED_CATEGORICAL_VALUE<br>• 200: MEAN<br>• 201: MEDIAN<br>• 203:SPECIFIED_NUMERICAL_VALUE<br>• 204: ALS<br>For type 101 (SPECIFIED_CATEGORICAL_VALUE), use the following syntax to specify a categorical value to be filled in a column V0:<br>INSERT INTO #PAL_PARAMETER_TBL VALUES('V0_IMPUTATION_TYPE', 101, NULL, '`<CATEGORICAL_VALUE>`');<br>For type 203 (SPECIFIED_NUMERICAL_VALUE), use the following syntax to specify a numerical value to be filled in a column V1:<br>INSERT INTO #PAL_PARAMETER_TBL VALUES('V1_IMPUTA- | Only valid when HANDLE_MISSING_VALUE is 1. |

| Name | Data Type | Default Value | Description | Constraint |
|---|---|---|---|---|
| | | | TION_TYPE', 203, <NUMERICAL_VALUE>, NULL);<br><br>**Note**<br>A column of integer type can be set to type 101 (SPECIFIED_CATEGORICAL_VALUE) or type 203 (SPECIFIED_NUMERICAL_VALUE):<br>• when setting to type 101 (SPECIFIED_CATEGORICAL_VALUE), the string must contain a valid integer value, otherwise an error will be thrown;<br>• when setting to type 203 (SPECIFIED_NUMERICAL_VALUE), the double value will be converted to integer. | |

| Name | Data Type | Default Value | Description | Constraint |
|---|---|---|---|---|
| ALS_FACTOR_NUMBER | INTEGER | 3 | Length of factor vectors, that is, the f in ALS model.<br>**Note**<br>ALS_FACTOR_NUMBER should be less than the number of numerical columns to get a meaningful result. | |
| ALS_REGULARIZATION | DOUBLE | 1e-2 | L2 regularization of the factors, that is, the λ in ALS model. | |
| ALS_MAX_ITERATION | INTEGER | 20 | Maximum number of iterations when training ALS model. | |
| ALS_SEED | INTEGER | 0 | Specifies the seed of the random number generator used in the training of ALS model:<br>• 0: Uses the current time as the seed<br>• Others: Uses the specified value as the seed. | |
| ALS_EXIT_THRESHOLD | DOUBLE | 0.0 | The training of ALS model exits if the value of cost function is not decreased more than the value of ALS_EXIT_THRESHOLD since the last check.<br>If ALS_EXIT_THRESHOLD is set to 0, there is no check and the training only exits on reaching the maximum number of iterations. Evaluations of cost function require additional calculations. You can set this parameter to 0 to avoid it. | |

| Name | Data Type | Default Value | Description | Constraint |
|---|---|---|---|---|
| ALS_EXIT_INTERVAL | INTEGER | 5 | During the training of ALS model, it will calculate cost function and check the exit criteria every `<ALS_EXIT_INTERVAL>` iterations. Evaluations of cost function require additional calculations. | |
| ALS_LINEAR_SYSTEM_SOLVER | INTEGER | 0 | • 0: cholesky solver<br>• 1: cg solver (recommended when ALS_FACTOR_NUMBER is large) | |
| ALS_CG_MAX_ITERATION | INTEGER | 3 | Maximum number of iteration of cg solver. | |
| ALS_CENTERING | INTEGER | 1 | Whether to center the data by column before training ALS model.<br>• 0: no centering<br>• 1: centering. | |
| ALS_SCALING | INTEGER | 1 | Whether to scale the data by column before training ALS model.<br>• 0: no scaling<br>• 1: scaling | |

## GLM

Mandatory parameters

None

Optional parameters

| Name | Data Type | Default Value | Description | Constraint |
|---|---|---|---|---|
| TYPE | VARCHAR or NVARCHAR | response | The prediction type:<br>• response: Predicts response μ;<br>• link: Predicts linear predictor η. | |

| Name | Data Type | Default Value | Description | Constraint |
|---|---|---|---|---|
| SIGNIFICANCE_LEVEL | DOUBLE | 0.05 | The significance level for the confidence interval and prediction interval. | Valid only for IRLS. |
| HANDLE_MISSING | INTEGER | 2 | Specify how to handle missing covariates.<br>• 1: Remove missing rows.<br>• 2: Replace missing with 0. | |

**RDT**

Mandatory parameters

None

Optional parameters

| Name | Data Type | Default Value | Description | Constraint |
|---|---|---|---|---|
| BLOCK_SIZE | INTEGER | 0 | Specifies the number of data loaded per time during scoring.<br>• 0: load all data once;<br>• Others: the specified number. | This parameter is for reducing memory consumption, especially as the predict data is huge, or it consists of a large number of missing independent variables. However, you might lose some efficiency. |

## 5.13.9.3 Data Partition

Data partition process is provided in the train procedure implementation.

The input data can be divided into the train set and the validation set. The train set is used for generating the training model and the validation set is used to evaluate the quality of the model.

You can execute data partition by setting the following parameters.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| PARTITION_METHOD | INTEGER | 0 | Data set Parition via different algorithm:<br>• 0: Do not use partition;<br>• 1: Data set partition with specific flag;<br>• 2: Data set partition with random. | If the parameter is set to be 1, an additional purpose column in data table is needed. |
| PARTITION_TRAINING_PERCENT | DOUBLE | 0.8 | Specific scale value to data partition with random.<br>Value: 0-1 | Valid only for partition_method=2. |
| PARTITION_RANDOM_SEED | INTEGER | 0 | Indicates the seed used to initialize the random number generator.<br>• 0: Uses the system time;<br>• Not 0: Uses the specific seed. | Valid only for partition_method=2. |
| OUTPUT_PARTITION_RESULT | INTEGER | 0 | Output partition result or not:<br>• 0: Do not output result;<br>• 1: Output result. | |

### Examples

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

**Example 1: Data set partition with specific flag(PARTITION_METHOD=1)**

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_DATA_TBL;
CREATE COLUMN TABLE PAL_DATA_TBL(
    "ID" INTEGER,
    "OUTLOOK" VARCHAR(20),
    "TEMP" INTEGER,
    "HUMIDITY" DOUBLE,
    "WINDY" VARCHAR(10),
    "CLASS" INTEGER, -- for regression
    "Purpose" INTEGER
);
INSERT INTO PAL_DATA_TBL VALUES (1, 'Sunny', 75, 70, 'Yes', 1, 1);
INSERT INTO PAL_DATA_TBL VALUES (2, 'Sunny', 80, 90, 'Yes', 0, 1);
INSERT INTO PAL_DATA_TBL VALUES (3, 'Sunny', 85, 85, 'No', 0, 2);
```

```
INSERT INTO PAL_DATA_TBL VALUES (4, 'Sunny', 72, 95, 'No', 0, 1);
INSERT INTO PAL_DATA_TBL VALUES (5, 'Sunny', 69, 70, 'No', 1, 1);
INSERT INTO PAL_DATA_TBL VALUES (6, 'Overcast', 72, 90, 'Yes', 1, 2);
INSERT INTO PAL_DATA_TBL VALUES (7, 'Overcast', 83, 78, 'No', 0, 2);
INSERT INTO PAL_DATA_TBL VALUES (8, 'Overcast', 64, 65, 'Yes', 1, 1);
INSERT INTO PAL_DATA_TBL VALUES (9, 'Overcast', 81, 75, 'No', 1, 1);
INSERT INTO PAL_DATA_TBL VALUES (10, 'Rain', 71, 80, 'Yes', 0, 1);
INSERT INTO PAL_DATA_TBL VALUES (11, 'Rain', 65, 70, 'Yes', 0, 1);
INSERT INTO PAL_DATA_TBL VALUES (12, 'Rain', 75, 80, 'No', 1, 2);
INSERT INTO PAL_DATA_TBL VALUES (13, 'Rain', 68, 80, 'No', 1, 1);
INSERT INTO PAL_DATA_TBL VALUES (14, 'Rain', 70, 96, 'No', 0, 1);
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO PAL_PARAMETER_TBL VALUES ('FUNCTION',null,null,'DT');
INSERT INTO PAL_PARAMETER_TBL VALUES ('KEY',1,null,null);
INSERT INTO PAL_PARAMETER_TBL VALUES ('MODEL_FORMAT', 2, NULL, NULL); -- PMML
INSERT INTO PAL_PARAMETER_TBL VALUES ('SPLIT_THRESHOLD', NULL, 1e-5, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('MIN_RECORDS_PARENT', 2, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('MIN_RECORDS_LEAF', 1, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('PARTITION_METHOD',1,null,null);
INSERT INTO PAL_PARAMETER_TBL VALUES ('OUTPUT_PARTITION_RESULT', 1,null, null);
INSERT INTO PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', null,0.5,null);
CALL _SYS_AFL.PAL_UNIFIED_REGRESSION(PAL_DATA_TBL,
PAL_PARAMETER_TBL, ?, ?, ?, ?, ?, ?);
```

### Excepted Result

PARTITION RESULT

| | ID | TYPE |
|---|---|---|
| 1 | 1 | 1 |
| 2 | 2 | 1 |
| 3 | 4 | 1 |
| 4 | 5 | 1 |
| 5 | 8 | 1 |
| 6 | 9 | 1 |
| 7 | 10 | 1 |
| 8 | 11 | 1 |
| 9 | 13 | 1 |
| 10 | 14 | 1 |
| 11 | 3 | 2 |
| 12 | 6 | 2 |
| 13 | 7 | 2 |
| 14 | 12 | 2 |

### Example 2: Data set partition with random(PARTITION_METHOD=2)

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_DATA_TBL;
CREATE COLUMN TABLE PAL_DATA_TBL(
    "OUTLOOK" VARCHAR(20),
    "TEMP" INTEGER,
    "HUMIDITY" DOUBLE,
    "WINDY" VARCHAR(10),
```

```
    "CLASS" INTEGER -- for regression
);
INSERT INTO PAL_DATA_TBL VALUES ('Sunny', 75, 70, 'Yes', 1);
INSERT INTO PAL_DATA_TBL VALUES ('Sunny', 80, 90, 'Yes', 0);
INSERT INTO PAL_DATA_TBL VALUES ('Sunny', 85, 85, 'No', 0);
INSERT INTO PAL_DATA_TBL VALUES ('Sunny', 72, 95, 'No', 0);
INSERT INTO PAL_DATA_TBL VALUES ('Sunny', 69, 70, 'No', 1);
INSERT INTO PAL_DATA_TBL VALUES ('Overcast', 72, 90, 'Yes', 1);
INSERT INTO PAL_DATA_TBL VALUES ('Overcast', 83, 78, 'No', 0);
INSERT INTO PAL_DATA_TBL VALUES ('Overcast', 64, 65, 'Yes', 1);
INSERT INTO PAL_DATA_TBL VALUES ('Overcast', 81, 75, 'No', 1);
INSERT INTO PAL_DATA_TBL VALUES ('Rain', 71, 80, 'Yes', 0);
INSERT INTO PAL_DATA_TBL VALUES ('Rain', 65, 70, 'Yes', 0);
INSERT INTO PAL_DATA_TBL VALUES ('Rain', 75, 80, 'No', 1);
INSERT INTO PAL_DATA_TBL VALUES ('Rain', 68, 80, 'No', 1);
INSERT INTO PAL_DATA_TBL VALUES ('Rain', 70, 96, 'No', 0);
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO PAL_PARAMETER_TBL VALUES ('FUNCTION',null,null,'DT');
INSERT INTO PAL_PARAMETER_TBL VALUES ('MODEL_FORMAT', 2, NULL, NULL); -- PMML
INSERT INTO PAL_PARAMETER_TBL VALUES ('SPLIT_THRESHOLD', NULL, 1e-5, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('MIN_RECORDS_PARENT', 2, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('MIN_RECORDS_LEAF', 1, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('PARTITION_METHOD',2,null,null);
--INSERT INTO PAL_PARAMETER_TBL VALUES ('PARTITION_TRAINING_PERCENT',
null,0.5,null);--Default value: 0.8
INSERT INTO PAL_PARAMETER_TBL VALUES ('OUTPUT_PARTITION_RESULT', 1,null, null);
INSERT INTO PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', null,0.5,null);
CALL _SYS_AFL.PAL_UNIFIED_REGRESSION(PAL_DATA_TBL,
PAL_PARAMETER_TBL, ?, ?, ?, ?, ?, ?);
```

**Excepted Result**

PARTITION RESULT

| | ID | TYPE |
|---|---|---|
| 1 | 2 | 1 |
| 2 | 4 | 1 |
| 3 | 5 | 1 |
| 4 | 6 | 1 |
| 5 | 7 | 1 |
| 6 | 8 | 1 |
| 7 | 9 | 1 |
| 8 | 11 | 1 |
| 9 | 12 | 1 |
| 10 | 13 | 1 |
| 11 | 14 | 1 |
| 12 | 1 | 2 |
| 13 | 3 | 2 |
| 14 | 10 | 2 |

## 5.13.9.4 Model Evaluation

Unified regression provides one output tables as the evaluating results of the training model: statistics table.

It is generated in both the train and scoring procedures. In the training procedure, the model evaluation works only if the PARTITION_METHOD is 1 or 2.

The statistic without "TEST_" is calculated by calling the original algorithm for all training data; while the statistic with "TEST_" is calculated by the unified function of the partitioned test data. For model evaluation, it is better to use the latter.

### Statistics Table

In the statistics table, the following statistics are supported:

- R2: R-squared (R2) is a statistical measure that represents the proportion of the variance for a dependent variable that's explained by an independent variable or variables in a regression model. R2 is calculated by the following equation:

  $$R^2 = 1 - \frac{RSS}{TSS} = 1 - \frac{\sum_i (y_i - y_i')^2}{\sum_i (y_i - \bar{y})^2}$$

  Where $y_i$ is an observed value, $y'_i$ is an expected value and $\bar{y}$ is a mean observed value.
- MSE: See Mean squared error for more details;
- RMSE: See Root mean squared error for more details;
- MAE: Mean Absolute Error (MAE) is a measure of errors between paired observations expressing the same phenomenon. MAE is calculated by the following equation:

  $$MAE = \frac{1}{m}\sum_i^m |y_i - y_i'|$$

  Where $y_i$ is an observed value and $y'_i$ is an expected value.
- MAPE: See Mean absolute percentage error for more details;
- WMAPE: See Weighted mean absolute percentage error for more details;
- EVar: Explained Variance(EVar) measures the proportion to which a mathematical model accounts for the variation (dispersion) of a given data set. EVar is calculated by the following equation:

  $$EVar = 1 - \frac{Var\{y-y'\}}{Var\{y\}}$$

  Where `Var` is Variance and calculated by the following equation:

  $$Var = \frac{\sum (x_i - \bar{x})^2}{n-1}.$$
- Max Error: MAx Error represents the maximum absolute difference between the observed value and the expected value. Max Error is calculated by the following equation:
  `Max_Error=MAX(|y-y'|)`
  Where y is an observed value and y' is an expected value.
- MSLE: Mean squared logarithmic error(MSLE) computes a risk metric corresponding to the expected value of the squared logarithmic (quadratic) error or loss. MSLE is calculated by the following equation:

  $$MSLE = \frac{1}{n}\sum_i^{n-1} (\log_e(1+y_i) - \log_e(1+y_i'))^2$$

  Where y is the observed value, `y'` is the expected value. It's only for exponential regression.

#### Example

Assume that:

- DM_PAL is a schema belonging to USER1;
- USER1 has been assigned the AFL__SYS_AFL_AFLPAL_EXECUTE or AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION role.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_DATA_TBL;
CREATE COLUMN TABLE PAL_DATA_TBL(
    "OUTLOOK" VARCHAR(20),
    "TEMP" INTEGER,
    "HUMIDITY" DOUBLE,
    "WINDY" VARCHAR(10),
    "CLASS" INTEGER -- for regression
);
INSERT INTO PAL_DATA_TBL VALUES ('Sunny', 75, 70, 'Yes', 1);
INSERT INTO PAL_DATA_TBL VALUES ('Sunny', 80, 90, 'Yes', 0);
INSERT INTO PAL_DATA_TBL VALUES ('Sunny', 85, 85, 'No', 0);
INSERT INTO PAL_DATA_TBL VALUES ('Sunny', 72, 95, 'No', 0);
INSERT INTO PAL_DATA_TBL VALUES ('Sunny', 69, 70, 'No', 1);
INSERT INTO PAL_DATA_TBL VALUES ('Overcast', 72, 90, 'Yes', 1);
INSERT INTO PAL_DATA_TBL VALUES ('Overcast', 83, 78, 'No', 0);
INSERT INTO PAL_DATA_TBL VALUES ('Overcast', 64, 65, 'Yes', 1);
INSERT INTO PAL_DATA_TBL VALUES ('Overcast', 81, 75, 'No', 1);
INSERT INTO PAL_DATA_TBL VALUES ('Rain', 71, 80, 'Yes', 0);
INSERT INTO PAL_DATA_TBL VALUES ('Rain', 65, 70, 'Yes', 0);
INSERT INTO PAL_DATA_TBL VALUES ('Rain', 75, 80, 'No', 1);
INSERT INTO PAL_DATA_TBL VALUES ('Rain', 68, 80, 'No', 1);
INSERT INTO PAL_DATA_TBL VALUES ('Rain', 70, 96, 'No', 0);
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO PAL_PARAMETER_TBL VALUES ('FUNCTION',null,null,'DT');
INSERT INTO PAL_PARAMETER_TBL VALUES ('MODEL_FORMAT', 2, NULL, NULL); -- PMML
INSERT INTO PAL_PARAMETER_TBL VALUES ('SPLIT_THRESHOLD', NULL, 1e-5, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('MIN_RECORDS_PARENT', 2, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('MIN_RECORDS_LEAF', 1, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('PARTITION_METHOD',2,null,null);
--INSERT INTO PAL_PARAMETER_TBL VALUES ('PARTITION_TRAINING_PERCENT',
null,0.5,null);--Default value: 0.8
INSERT INTO PAL_PARAMETER_TBL VALUES ('OUTPUT_PARTITION_RESULT', 1,null, null);
INSERT INTO PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', null,0.5,null);
CALL _SYS_AFL.PAL_UNIFIED_REGRESSION(PAL_DATA_TBL,
PAL_PARAMETER_TBL, ?, ?, ?, ?, ?, ?);
```

**Excepted Result**

STATISTIC

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | TEST_EVAR | -1.1111118888889995 |
| 2 | TEST_MAE | 0.5555556666666667 |
| 3 | TEST_MAPE | 0.5555556666666667 |
| 4 | TEST_MAX_ERROR | 1 |
| 5 | TEST_MSE | 0.4814816296296667 |
| 6 | TEST_R2 | -1.1666673333334998 |
| 7 | TEST_RMSE | 0.693888773240832 |
| 8 | TEST_WMAPE | 1.666667 |

## 5.13.9.5 Model Compression

The training process may cost a large size of memory. Model compression aims to reduce the size of the model with minimum loss of the accuracy.

Model compression is provided for Support Vector, Random Decision Trees, and Hybrid Gradient Boosting Tree. The following tables list the parameters for using model compression in Support Vector Machine, Random Decision Trees, and Hybrid Gradient Boosting Tree respectively.

**Support Vector Machine**

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| COMPRESSION | INTEGER | 1 | Specifies if the model is stored in compressed format.<br>• 0: no;<br>• 1: yes. | |
| MAX_BITS | INTEGER | 12 | The maximum number of bits to quantize continuous features. Equivalent to use $2^{MAX_BITS}$ bins.<br>Must be less than 31. | Valid only when the value of COMPRESSION is 1. |
| MAX_QUANTIZATION_ITER | INTEGER | 1000 | The maximum iteration steps for quantization. | Valid only when the value of COMPRESSION is 1. |

**Random Decision Trees**

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| COMPRESSION | INTEGER | 0 | Specifies if the model is stored in compressed format.<br>• 0: no;<br>• 1: yes. | |
| MAX_BITS | INTEGER | 12 | The maximum number of bits to quantize continuous features. Equivalent to use $2^{MAX_BITS}$ bins.<br>Must be less than 31. | Valid only when the value of COMPRESSION is 1. |
| QUANTIZE_RATE | DOUBLE | 0.005 | Quantize a feature if the largest class frequency of which is less than QUANTIZE_RATE. | Valid only when the value of COMPRESSION is 1. |
| FITTINGS_QUANTIZATION | INTEGER | 0 | Indicate whether to quantize the fitting values.<br>• 0: no;<br>• 1: yes. | |

### Hybrid Gradient Boosting Tree

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| COMPRESSION | INTEGER | 0 | Indicates whether result model should be compressed:<br>• 0. no;<br>• 1. yes. | |
| MAX_BITS | INTEGER | 12 | Maximum number of bits to quantize continuous features. Equivalent to use $2^{MAX_BITS}$ bins. | Must be less than 31. |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA DM_PAL;
DROP TABLE  PAL_DATA_TBL;
CREATE COLUMN TABLE PAL_DATA_TBL (
    "OUTLOOK" VARCHAR(20),
    "TEMP" DOUBLE,
    "HUMIDITY" DOUBLE,
```

```
    "WINDY" VARCHAR(10),
    "CLASS" INTEGER
);
INSERT INTO PAL_DATA_TBL VALUES ('Sunny', 75, 70.0, 'Yes', 1);
INSERT INTO PAL_DATA_TBL VALUES ('Sunny', 80, 90.0, 'Yes', 0);
INSERT INTO PAL_DATA_TBL VALUES ('Sunny', 85, 91.0, 'No', 0);
INSERT INTO PAL_DATA_TBL VALUES ('Sunny', 72, 95.0, 'No', 0);
INSERT INTO PAL_DATA_TBL VALUES ('Sunny', 73, 70.0, 'No', 1);
INSERT INTO PAL_DATA_TBL VALUES ('Overcast', 72.0, 90, 'Yes', 1);
INSERT INTO PAL_DATA_TBL VALUES ('Overcast', 83.0, 78, 'No', 1);
INSERT INTO PAL_DATA_TBL VALUES ('Overcast', 64.0, 65, 'Yes', 1);
INSERT INTO PAL_DATA_TBL VALUES ('Overcast', 81.0, 75, 'No', 1);
INSERT INTO PAL_DATA_TBL VALUES ('Rain', 71, 80.0, 'Yes', 0);
INSERT INTO PAL_DATA_TBL VALUES ('Rain', 65, 70.0, 'Yes', 0);
INSERT INTO PAL_DATA_TBL VALUES ('Rain', 75, 80.0, 'No', 1);
INSERT INTO PAL_DATA_TBL VALUES ('Rain', 68, 80.0, 'No', 1);
INSERT INTO PAL_DATA_TBL VALUES ('Rain', 70, 96.0, 'No', 1);
DROP TABLE PAL_PARAMETER_TBL;
CREATE  COLUMN TABLE PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (100),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (100)
);
INSERT INTO PAL_PARAMETER_TBL VALUES ('FUNCTION', NULL, NULL, 'RDT');
INSERT INTO PAL_PARAMETER_TBL VALUES ('SEED', 2, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('N_ESTIMATORS', 200, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('MAX_FEATURES', 3, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('COMPRESSION', 1, NULL,
NULL);                        ------parameter for model compression
DROP TABLE PAL_MODEL_TBL;
CREATE COLUMN TABLE PAL_MODEL_TBL (
    "ROW_INDEX" INTEGER,
    "PART_INDEX" INTEGER,
    "MODEL_CONTENT" NCLOB
);
DO
BEGIN
    lt_data = SELECT * FROM PAL_DATA_TBL;
    lt_ctrl = SELECT * FROM PAL_PARAMETER_TBL;
    CALL _SYS_AFL.PAL_UNIFIED_CLASSIFICATION (:lt_data, :lt_ctrl, lt_model,
lt_imp,lt_stat, lt_opt,lt_cm,lt_metrics,lt_partition,lt_ph1);
    INSERT INTO PAL_MODEL_TBL SELECT * FROM :lt_model;
END;
SELECT * FROM PAL_MODEL_TBL;
```

### Excepted Result

MODEL

| | ROW_INDEX | PART_INDEX | MODEL_CONTENT |
|---|---|---|---|
| 1 | 0 | -1 | {"atFields":[{"dataType":"string","name":"OUTLOOK","opType":"cat","values":["Sunny"]},{"... |
| 2 | 1 | 0 | {"cw":"ppIQQ%\u001a\u001a\u0014S\f>\u0010\|!Ilq\u000f\biaSBbc,\u001aaF\u0014SCBV\u... |
| 3 | 2 | 1 | {"cw":"(7L\u0005\u0006y@4P\r\u0014\u0007 :ih\u000eZ4\u0004QP\u0012J\u001c\u0011\t... |
| 4 | 3 | 2 | {"cw":"\|YKYmoK\u0019;\u001d]\u0007g,\u0010\|uApo`\n\u001c\u001e\r\|\u0001#@/S4@\u... |
| 5 | 4 | 2 | 7\u000e\u0007\u0002\u0001\u0001\u0005\u0003\u0002\u0001\u0001\u0007\u000e\t\u000... |
| 6 | 5 | 2 | 003\u000e\n\u0002\u0001\u0001\b\u0004\u0003\u0001\u000e\n\u0002\u0001\u0001\b\u... |
| 7 | 6 | 3 | {"at":0,"cvs":["\u0000","\u0002\u0001","\u0002","\u0001\u0000","\u0002\u0000","\u0001"],... |
| 8 | 7 | 3 | 0002\u0002\u0002\u0002\u0002\u0002","trees":"Hlpqvw\u0092\u0093\u00c0\u00c1"},{"cw... |
| 9 | 8 | 4 | {"at":1,"cvs":["77.5","72.5","84","71","66.5","64.5","68.5","73","80","69.5","76.5","70.5","71.5",... |
| 10 | 9 | 4 | u0081\u0086\u0087\u0088\u0089\u008e\u008f\u0092\u0093\u0098\u0099\u00a0\u00a1\u... |
| 11 | 10 | 5 | {"at":2,"cvs":["80","85","90.5","92.5","75","95.5","79","80.5","77.5","82.5","72.5","87.5"],"dicts... |
| 12 | 11 | 6 | {"at":3,"cvs":["\u0000","\u0001"],"dicts":[[{"p":0.5,"v":0},{"p":0.5,"v":1}],[{"p":0.5,"v":0},{"p":... |

## 5.13.9.6 Compatibility

The unified regression interface and original PAL procedures can use models obtained from each other for prediction and scoring.

Unified regression interface can use the models generated from the original PAL procedures for prediction and scoring without any additional processing.

The following table shows whether the model needs the remove processing.

| Model Source | Model Destination | Model Compatibility | Remove Processing |
|---|---|---|---|
| Unified Regression | GEO | YES | YES |
| | LOG | YES | YES |
| | EXP | YES | YES |
| | GLM | NO | N/A |
| | MLR | YES | YES |
| | POL | YES | YES |
| | DT | YES | YES |
| | HGBT | YES | NO |
| | MLP | YES | YES |
| | RDT | YES | NO |
| | SVM | YES | YES |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

**Example 1: Using a unified regression model in an original SVM prediction procedure**

```
SET SCHEMA DM_PAL;
------------------------Unified Regression Training
Procedure-----------------------
--Data Table
DROP TABLE PAL_DATA_TBL;
CREATE COLUMN TABLE PAL_DATA_TBL (
    ID INTEGER,
    ATTRIBUTE1 DOUBLE,
    ATTRIBUTE2 DOUBLE,
    ATTRIBUTE3 DOUBLE,
    ATTRIBUTE4 VARCHAR(10),
    LABEL INTEGER
);
INSERT INTO PAL_DATA_TBL VALUES(0,1,10,100,'A',1);
INSERT INTO PAL_DATA_TBL VALUES(1,1.1,10.1,100,'A',1);
INSERT INTO PAL_DATA_TBL VALUES(2,1.2,10.2,100,'A',1);
INSERT INTO PAL_DATA_TBL VALUES(3,1.3,10.4,100,'A',1);
INSERT INTO PAL_DATA_TBL VALUES(4,1.2,10.3,100,'AB',1);
INSERT INTO PAL_DATA_TBL VALUES(5,4,40,400,'AB',1);
INSERT INTO PAL_DATA_TBL VALUES(6,4.1,40.1,400,'AB',1);
```

```
INSERT INTO PAL_DATA_TBL VALUES(7,4.2,40.2,400,'AB',1);
INSERT INTO PAL_DATA_TBL VALUES(8,4.3,40.4,400,'AB',2);
INSERT INTO PAL_DATA_TBL VALUES(9,4.2,40.3,400,'AB',2);
INSERT INTO PAL_DATA_TBL VALUES(10,9,90,900,'B',2);
INSERT INTO PAL_DATA_TBL VALUES(11,9.1,90.1,900,'A',2);
INSERT INTO PAL_DATA_TBL VALUES(12,9.2,90.2,900,'B',2);
INSERT INTO PAL_DATA_TBL VALUES(13,9.3,90.4,900,'A',2);
INSERT INTO PAL_DATA_TBL VALUES(14,9.2,90.3,900,'A',2);
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO PAL_PARAMETER_TBL VALUES ('FUNCTION',null,null,'SVM');
INSERT INTO PAL_PARAMETER_TBL VALUES ('PARTITION_METHOD',2,null,null);
INSERT INTO PAL_PARAMETER_TBL VALUES ('KEY',1,null,null);
INSERT INTO PAL_PARAMETER_TBL VALUES ('RANDOM_SEED',2,null,null);
INSERT INTO PAL_PARAMETER_TBL VALUES ('OUTPUT_PARTITION_RESULT',1,null,null);
INSERT INTO PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', null,0.5,null);
--Model Table
DROP TABLE PAL_DATA_MODEL_TBL;
CREATE COLUMN TABLE PAL_DATA_MODEL_TBL(
     "ROW_INDEX" INTEGER,
     "PART_INDEX" INTEGER,
     "MODEL_CONTENT" VARCHAR (5000)
);
DO BEGIN
  data_tab = SELECT * FROM PAL_DATA_TBL;
  control_tab = SELECT * FROM PAL_PARAMETER_TBL;
  CALL _SYS_AFL.PAL_UNIFIED_REGRESSION(:data_tab, :control_tab, model_tab,
stats_tab, optimal_parameter_tab, partition_result, place_holder_1_tab,
place_holder_2_tab);
  INSERT into PAL_DATA_MODEL_TBL SELECT * FROM :model_tab;
END
--select * from PAL_DATA_MODEL_TBL;
-------------------------Origin SVM Procedure------------------------
DROP TABLE PAL_ORIGIN_MODEL_TBL;
CREATE COLUMN TABLE PAL_ORIGIN_MODEL_TBL (
"ROW_INDEX" INTEGER,
"MODEL_CONTENT" VARCHAR (5000)
);
------------------Remove Processing-----------------------
INSERT INTO PAL_ORIGIN_MODEL_TBL(ROW_INDEX,MODEL_CONTENT) SELECT ROW_INDEX,
MODEL_CONTENT from PAL_DATA_MODEL_TBL;
--select * from PAL_ORIGIN_MODEL_TBL;
DROP TABLE PAL_SVM_PREDICT_DATA_TBL;
CREATE COLUMN TABLE PAL_SVM_PREDICT_DATA_TBL(
    ID INTEGER,
    ATTRIBUTE1 DOUBLE,
    ATTRIBUTE2 DOUBLE,
    ATTRIBUTE3 DOUBLE,
    ATTRIBUTE4 VARCHAR(50)
);
INSERT INTO PAL_SVM_PREDICT_DATA_TBL VALUES(0,1,10,100,'A');
INSERT INTO PAL_SVM_PREDICT_DATA_TBL VALUES(1,1.1,10.1,100,'A');
INSERT INTO PAL_SVM_PREDICT_DATA_TBL VALUES(2,1.2,10.2,100,'A');
INSERT INTO PAL_SVM_PREDICT_DATA_TBL VALUES(3,1.3,10.4,100,'A');
INSERT INTO PAL_SVM_PREDICT_DATA_TBL VALUES(4,1.2,10.3,100,'AB');
INSERT INTO PAL_SVM_PREDICT_DATA_TBL VALUES(5,4,40,400,'AB');
INSERT INTO PAL_SVM_PREDICT_DATA_TBL VALUES(6,4.1,40.1,400,'AB');
INSERT INTO PAL_SVM_PREDICT_DATA_TBL VALUES(7,4.2,40.2,400,'AB');
INSERT INTO PAL_SVM_PREDICT_DATA_TBL VALUES(8,4.3,40.4,400,'AB');
INSERT INTO PAL_SVM_PREDICT_DATA_TBL VALUES(9,4.2,40.3,400,'AB');
INSERT INTO PAL_SVM_PREDICT_DATA_TBL VALUES(10,9,90,900,'B');
INSERT INTO PAL_SVM_PREDICT_DATA_TBL VALUES(11,9.1,90.1,900,'A');
INSERT INTO PAL_SVM_PREDICT_DATA_TBL VALUES(12,9.2,90.2,900,'B');
```

```
INSERT INTO PAL_SVM_PREDICT_DATA_TBL VALUES(13,9.3,90.4,900,'A');
INSERT INTO PAL_SVM_PREDICT_DATA_TBL VALUES(14,9.2,90.3,900,'A');
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES('THREAD_RATIO',NULL,0.1,NULL);
CALL _SYS_AFL.PAL_SVM_PREDICT(PAL_SVM_PREDICT_DATA_TBL, PAL_ORIGIN_MODEL_TBL,
#PAL_PARAMETER_TBL,?);
```

**Excepted Result**

RESULT

| | ID | SCORE | PROBABILITY |
|---|---|---|---|
| 1 | 0 | 1.00961 | ? |
| 2 | 1 | 1.02803 | ? |
| 3 | 2 | 1.0474 | ? |
| 4 | 3 | 1.06663 | ? |
| 5 | 4 | 0.965765 | ? |
| 6 | 5 | 1.76642 | ? |
| 7 | 6 | 1.83203 | ? |
| 8 | 7 | 1.89769 | ? |
| 9 | 8 | 1.96355 | ? |
| 10 | 9 | 1.89792 | ? |
| 11 | 10 | 2.19385 | ? |
| 12 | 11 | 2.05301 | ? |
| 13 | 12 | 2.19204 | ? |
| 14 | 13 | 2.04936 | ? |
| 15 | 14 | 2.05072 | ? |

**Example 2: Using an origin training model in a unified regression prediction procedure**

```
SET SCHEMA DM_PAL;
-------------------------Origin SVM Training Procedure------------------------
DROP TABLE PAL_SVM_DATA_TBL;
CREATE COLUMN TABLE PAL_SVM_DATA_TBL (
    ID INTEGER,
    ATTRIBUTE1 DOUBLE,
    ATTRIBUTE2 DOUBLE,
    ATTRIBUTE3 DOUBLE,
    ATTRIBUTE4 VARCHAR(10),
    LABEL INTEGER
);
INSERT INTO PAL_SVM_DATA_TBL VALUES(0,1,10,100,'A',1);
INSERT INTO PAL_SVM_DATA_TBL VALUES(1,1.1,10.1,100,'A',1);
INSERT INTO PAL_SVM_DATA_TBL VALUES(2,1.2,10.2,100,'A',1);
INSERT INTO PAL_SVM_DATA_TBL VALUES(3,1.3,10.4,100,'A',1);
INSERT INTO PAL_SVM_DATA_TBL VALUES(4,1.2,10.3,100,'AB',1);
INSERT INTO PAL_SVM_DATA_TBL VALUES(5,4,40,400,'AB',1);
INSERT INTO PAL_SVM_DATA_TBL VALUES(6,4.1,40.1,400,'AB',1);
INSERT INTO PAL_SVM_DATA_TBL VALUES(7,4.2,40.2,400,'AB',1);
INSERT INTO PAL_SVM_DATA_TBL VALUES(8,4.3,40.4,400,'AB',2);
INSERT INTO PAL_SVM_DATA_TBL VALUES(9,4.2,40.3,400,'AB',2);
INSERT INTO PAL_SVM_DATA_TBL VALUES(10,9,90,900,'B',2);
```

```
INSERT INTO PAL_SVM_DATA_TBL VALUES(11,9.1,90.1,900,'A',2);
INSERT INTO PAL_SVM_DATA_TBL VALUES(12,9.2,90.2,900,'B',2);
INSERT INTO PAL_SVM_DATA_TBL VALUES(13,9.3,90.4,900,'A',2);
INSERT INTO PAL_SVM_DATA_TBL VALUES(14,9.2,90.3,900,'A',2);
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL (
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR (1000)
);
INSERT INTO PAL_PARAMETER_TBL VALUES('KERNEL_TYPE',2,NULL,NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES('TYPE',2,NULL,NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES('RBF_GAMMA',NULL,0.005,NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES('HANDLE_MISSING', 0, NULL, NULL);
DROP TABLE PAL_SVM_MODEL_TBL_EX3;
CREATE COLUMN TABLE PAL_SVM_MODEL_TBL_EX3 (
    "ROW_INDEX" INTEGER,
    "MODEL_CONTENT" NVARCHAR(5000)
);
DO BEGIN
  data_tab = SELECT * FROM PAL_SVM_DATA_TBL;
  control_tab = SELECT * FROM PAL_PARAMETER_TBL;
  CALL _SYS_AFL.PAL_SVM(:data_tab, :control_tab, model_tab, stats_tab,
optimal_parameter_tab);
  INSERT into PAL_SVM_MODEL_TBL_EX3 SELECT * FROM :model_tab;
END
--select * from PAL_SVM_MODEL_TBL_EX3;
-------------------------Unified Regression
Predict-------------------------------
DROP TABLE PAL_DATA_TBL_PRE;
CREATE COLUMN TABLE PAL_DATA_TBL_PRE(
    ID INTEGER,
    ATTRIBUTE1 DOUBLE,
    ATTRIBUTE2 DOUBLE,
    ATTRIBUTE3 DOUBLE,
    ATTRIBUTE4 VARCHAR(10)
);
INSERT INTO PAL_DATA_TBL_PRE VALUES(0,1,10,100,'A');
INSERT INTO PAL_DATA_TBL_PRE VALUES(1,1.2,10.2,100,'A');
INSERT INTO PAL_DATA_TBL_PRE VALUES(2,4.1,40.1,400,'AB');
INSERT INTO PAL_DATA_TBL_PRE VALUES(3,4.2,40.3,400,'AB');
INSERT INTO PAL_DATA_TBL_PRE VALUES(4,9.1,90.1,900,'A');
INSERT INTO PAL_DATA_TBL_PRE VALUES(5,9.2,90.2,900,'A');
INSERT INTO PAL_DATA_TBL_PRE VALUES(6,4,40,400,'A');
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FUNCTION',null,null,'SVM');
CALL _SYS_AFL.PAL_UNIFIED_REGRESSION_PREDICT(PAL_DATA_TBL_PRE,
PAL_SVM_MODEL_TBL_EX3, #PAL_PARAMETER_TBL, ?,?);
```

### Excepted Result

RESULT

| | ID | SCORE | UPPER_BOUND | LOWER_BOUND | REASON |
|---|---|---|---|---|---|
| 1 | 0 | 1.00961 | ? | ? | ? |
| 2 | 1 | 1.0474 | ? | ? | ? |
| 3 | 2 | 1.83203 | ? | ? | ? |
| 4 | 3 | 1.89792 | ? | ? | ? |
| 5 | 4 | 2.05301 | ? | ? | ? |
| 6 | 5 | 2.05202 | ? | ? | ? |
| 7 | 6 | 1.68458 | ? | ? | ? |

## 5.13.9.7 Local Interpretability of Models

In the process of machine learning predictive modeling, you can also view why the predicative results are made.

Based on the Shapley Values in game theory, Shapley Additive Explanations (SHAP) was proposed to explain individual predictions.

When it comes to specified machine learning model, the implementations of SHAP values have variant versions.

For tree based machine learning models, there are two explainers for the tree model: tree SHAP and Saabas. Tree SHAP explainer can exactly compute the Shapley values in polynomial time and satisfy the interpretation consistency. Saabas explainer is also implemented to calculate the attribution values, which takes only the single decision path into account and could run in a logarithmic time (over the number of nodes).

For linear based models, for example, Multiple Linear Regression, a linear SHAP expainer (linear SHAP) is provided. Linear SHAP uses samplings to estimate the SHAP values. The samplings called background data are sampled from the train dataset in the training process and stored in the model table.

For nonlinear and non-tree based models, for example, Support Vector Machine, a kernel SHAP explainer (kernel SHAP) is provided. Kernel SHAP does the same thing as linear SHAP, but using a kernel sampling. It is also be sampled from the train dataset in the training process and stored in the model table.

As mentioned in Exponential Regression [page 448], `y'` and `x1`...`xn` is a linear relationship and can be solved using the linear regression method. Therefore, the `y'(ln(y))` is explained via the linear SHAP of EXP, not the original output y.

As mentioned in Generalised Linear Models [page 456], compared with the classical linear regression, GLM regresses a linear predictor η instead of the response itself. Therefore, the η is explained via the linear SHAP of GLM, not the original output y. For example, `η=log(μ)` when the link function is Log.

The following table gives an overview of the supported SHAP versions for different methods.

| Function Name | Method |
|---|---|
| DT | • tree SHAP<br>• Saabas |
| RDT | • tree SHAP<br>• Saabas |

| Function Name | Method |
| --- | --- |
| HGBT | • tree SHAP<br>• Saabas |
| EXP | linear SHAP |
| GLM | linear SHAP |
| MLR | linear SHAP |
| MLP | kernel SHAP |
| SVM | kernel SHAP |

## _SYS_AFL.PAL_UNIFIED_REGRESSION

### Parameter Table

| Name | Data Type | Default Value | Description | Dependency |
| --- | --- | --- | --- | --- |
| BACKGROUND_SIZE | INTEGER | 0 | The row size of background data. | It should not be larger than the row size of train data. Valid for the functions:<br>• EXP<br>• GLM<br>• MLR<br>• MLP<br>• SVM |
| BACKGROUND_SAMPLING_SEED | INTEGER | 0 | Specified the seed for random number generator in the background sampling:<br>• 0: Uses the current time(in second) as seed;<br>• Others: Uses the specified value as seed. | Valid for the functions:<br>• EXP<br>• GLM<br>• MLR<br>• MLP<br>• SVM |

### Examples

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

#### Example 1: tree SHAP(Decision Tree)

```
SET SCHEMA DM_PAL;
```

```
DROP TABLE PAL_DATA_TBL;
CREATE COLUMN TABLE PAL_DATA_TBL(
    "OUTLOOK" VARCHAR(20),
    "TEMP" INTEGER,
    "HUMIDITY" DOUBLE,
    "WINDY" VARCHAR(10),
    "CLASS" INTEGER -- for regression
);
INSERT INTO PAL_DATA_TBL VALUES ('Sunny', 75, 70, 'Yes', 1);
INSERT INTO PAL_DATA_TBL VALUES ('Sunny', 80, 90, 'Yes', 0);
INSERT INTO PAL_DATA_TBL VALUES ('Sunny', 85, 85, 'No', 0);
INSERT INTO PAL_DATA_TBL VALUES ('Sunny', 72, 95, 'No', 0);
INSERT INTO PAL_DATA_TBL VALUES ('Sunny', 69, 70, 'No', 1);
INSERT INTO PAL_DATA_TBL VALUES ('Overcast', 72, 90, 'Yes', 1);
INSERT INTO PAL_DATA_TBL VALUES ('Overcast', 83, 78, 'No', 0);
INSERT INTO PAL_DATA_TBL VALUES ('Overcast', 64, 65, 'Yes', 1);
INSERT INTO PAL_DATA_TBL VALUES ('Overcast', 81, 75, 'No', 1);
INSERT INTO PAL_DATA_TBL VALUES ('Rain', 71, 80, 'Yes', 0);
INSERT INTO PAL_DATA_TBL VALUES ('Rain', 65, 70, 'Yes', 0);
INSERT INTO PAL_DATA_TBL VALUES ('Rain', 75, 80, 'No', 1);
INSERT INTO PAL_DATA_TBL VALUES ('Rain', 68, 80, 'No', 1);
INSERT INTO PAL_DATA_TBL VALUES ('Rain', 70, 96, 'No', 0);
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO PAL_PARAMETER_TBL VALUES ('FUNCTION',null,null,'DT');
DROP TABLE PAL_DATA_MODEL_TBL;
CREATE COLUMN TABLE PAL_DATA_MODEL_TBL(
     "ROW_INDEX" INTEGER,
     "PART_INDEX" INTEGER,
     "MODEL_CONTENT" VARCHAR (5000)
);
DO BEGIN
  data_tab = SELECT * FROM PAL_DATA_TBL;
  control_tab = SELECT * FROM PAL_PARAMETER_TBL;
  CALL _SYS_AFL.PAL_UNIFIED_REGRESSION(:data_tab, :control_tab, model_tab,
stats_tab, optimal_parameter_tab, partition_result, place_holder_1_tab,
place_holder_2_tab);
  INSERT into PAL_DATA_MODEL_TBL SELECT * FROM :model_tab;
END
```

### Expected Result

MODEL:

| | ROW_INDEX | PART_INDEX | MODEL_CONTENT |
|---|---|---|---|
| 1 | 0 | ? | {"CurrentVersion":"VERSION1.3","DataDictionary":[{"DataType":"String","ID":0,"Name":"OU... |

### Example 2: Based linear SHAP(MLR)

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_DATA_TBL;
CREATE COLUMN TABLE PAL_DATA_TBL(
     "ID" INTEGER,
     "X1" DOUBLE,
     "X2" VARCHAR (100),
     "X3" INTEGER,
     "Y" DOUBLE
);
INSERT INTO PAL_DATA_TBL VALUES (0, 0.0, 'A', 1, -6.879);
INSERT INTO PAL_DATA_TBL VALUES (1, 0.50, 'A', 1, -3.449);
INSERT INTO PAL_DATA_TBL VALUES (2, 0.54, 'B', 1, 6.635);
```

```
INSERT INTO PAL_DATA_TBL VALUES (3, 1.04, 'B', 1, 11.844);
INSERT INTO PAL_DATA_TBL VALUES (4, 1.50, 'A', 1, 2.786);
INSERT INTO PAL_DATA_TBL VALUES (5, 0.04, 'B', 2, 2.389);
INSERT INTO PAL_DATA_TBL VALUES (6, 2.00, 'A', 2, -0.011);
INSERT INTO PAL_DATA_TBL VALUES (7, 2.04, 'B', 2, 8.839);
INSERT INTO PAL_DATA_TBL VALUES (8, 1.54, 'B', 1, 4.689);
INSERT INTO PAL_DATA_TBL VALUES (9, 1.00, 'A', 2, -5.507);
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO PAL_PARAMETER_TBL VALUES ('FUNCTION',null,null,'MLR');
INSERT INTO PAL_PARAMETER_TBL VALUES ('BACKGROUND_SIZE', 4,null,null);
DROP TABLE PAL_DATA_MODEL_TBL;
CREATE COLUMN TABLE PAL_DATA_MODEL_TBL(
     "ROW_INDEX" INTEGER,
     "PART_INDEX" INTEGER,
     "MODEL_CONTENT" VARCHAR (5000)
);
DO BEGIN
  data_tab = SELECT * FROM PAL_DATA_TBL;
  control_tab = SELECT * FROM PAL_PARAMETER_TBL;
  CALL _SYS_AFL.PAL_UNIFIED_REGRESSION(:data_tab, :control_tab, model_tab,
stats_tab, optimal_parameter_tab, partition_result, place_holder_1_tab,
place_holder_2_tab);
  INSERT into PAL_DATA_MODEL_TBL SELECT * FROM :model_tab;
END
```

### Expected Result

MODEL:

| | ROW_INDEX | PART_INDEX | MODEL_CONTENT |
|---|---|---|---|
| 1 | 1 | ? | <PMML version="4.0" xmlns="http://www.dmg.org/PMML-4_0" xmlns:xsi="http://www.... |
| 2 | -10 | ? | {"cate_vals":[{"cateVal":["B","B","A"]}],"categorical_size":1,"cont_vals":[{"contVal":[0.540000... |

### Example 3: Based kernel SHAP(SVM)

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_DATA_TBL;
CREATE COLUMN TABLE PAL_DATA_TBL (
    ID INTEGER,
    ATTRIBUTE1 DOUBLE,
    ATTRIBUTE2 DOUBLE,
    ATTRIBUTE3 DOUBLE,
    ATTRIBUTE4 VARCHAR(10),
    LABEL INTEGER
);
INSERT INTO PAL_DATA_TBL VALUES(0,1,10,100,'A',1);
INSERT INTO PAL_DATA_TBL VALUES(1,1.1,10.1,100,'A',1);
INSERT INTO PAL_DATA_TBL VALUES(2,1.2,10.2,100,'A',1);
INSERT INTO PAL_DATA_TBL VALUES(3,1.3,10.4,100,'A',1);
INSERT INTO PAL_DATA_TBL VALUES(4,1.2,10.3,100,'AB',1);
INSERT INTO PAL_DATA_TBL VALUES(5,4,40,400,'AB',1);
INSERT INTO PAL_DATA_TBL VALUES(6,4.1,40.1,400,'AB',1);
INSERT INTO PAL_DATA_TBL VALUES(7,4.2,40.2,400,'AB',1);
INSERT INTO PAL_DATA_TBL VALUES(8,4.3,40.4,400,'AB',2);
INSERT INTO PAL_DATA_TBL VALUES(9,4.2,40.3,400,'AB',2);
INSERT INTO PAL_DATA_TBL VALUES(10,9,90,900,'B',2);
INSERT INTO PAL_DATA_TBL VALUES(11,9.1,90.1,900,'A',2);
INSERT INTO PAL_DATA_TBL VALUES(12,9.2,90.2,900,'B',2);
INSERT INTO PAL_DATA_TBL VALUES(13,9.3,90.4,900,'A',2);
```

```
INSERT INTO PAL_DATA_TBL VALUES(14,9.2,90.3,900,'A',2);
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO PAL_PARAMETER_TBL VALUES ('FUNCTION',null,null,'SVM');
INSERT INTO PAL_PARAMETER_TBL VALUES ('BACKGROUND_SIZE', 4,null,null);
INSERT INTO PAL_PARAMETER_TBL VALUES ('SCALE_INFO', 2,null,null);
DROP TABLE PAL_DATA_MODEL_TBL;
CREATE COLUMN TABLE PAL_DATA_MODEL_TBL(
     "ROW_INDEX" INTEGER,
     "PART_INDEX" INTEGER,
     "MODEL_CONTENT" VARCHAR (5000)
);
DO BEGIN
  data_tab = SELECT * FROM PAL_DATA_TBL;
  control_tab = SELECT * FROM PAL_PARAMETER_TBL;
  CALL _SYS_AFL.PAL_UNIFIED_REGRESSION(:data_tab, :control_tab, model_tab,
stats_tab, optimal_parameter_tab, partition_result, place_holder_1_tab,
place_holder_2_tab);
  INSERT into PAL_DATA_MODEL_TBL SELECT * FROM :model_tab;
END
```

### Expected Result

MODEL:

| | ROW_INDEX | PART_INDEX | MODEL_CONTENT |
|---|---|---|---|
| 1 | 0 | ? | {"C":100.0,"CONTENT":"{\"impute_model\":{\"column_statistics\":[{\"COLUMN_NAME\":\... |
| 2 | -10 | ? | {"cate_vals":[{"cateVal":["A","A"]}],"categorical_size":1,"cont_vals":[{"contVal":[1.2,1.100000... |

## _SYS_AFL.PAL_UNIFIED_REGRESSION_PREDICT

### Parameter Table

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| TOP_K_ATTRIBUTIONS | INTEGER | 10 | Outputs the attributions of top k features which contribute the most. | |
| FEATURE_ATTRIBUTION_METHOD | INTEGER | 2 | Controls which method to use in tree model reasoning:<br>• 0: no reasoning;<br>• 1: Saabas;<br>• 2: tree SHAP. | Valid for the functions:<br>• DT<br>• RDT<br>• HGBT |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| SAMPLESIZE | INTEGER | 0 | Number of sampled combinations of features. If it equals to 0, then it will be determined by algorithm heuristically. | It is better to use a number which is larger than the number of the columns of the data table.<br>Available for the following functions:<br>• EXP<br>• GLM<br>• MLR<br>• MLP<br>• SVM |
| SEED | INTEGER | 0 | Specifies the seed for random number generator:<br>• 0: Uses the current time(in second) as seed;<br>• Others: Uses the specified value as seed. | Valid for the functions:<br>• EXP<br>• GLM<br>• MLR<br>• MLP<br>• SVM |
| IGNORE_CORRELATION | INTEGER | 0 | Specifies whether or not to ignore the correlation between the features:<br>• 0: Not ignore the correlation;<br>• 1: Ignore the correlation. | Valid for the functions:<br>• EXP<br>• GLM<br>• MLR |

## Examples

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

### Example 1: tree SHAP(Decision Tree)

```
DROP TABLE PAL_DATA_TBL_PRE;
CREATE COLUMN TABLE PAL_DATA_TBL_PRE(
    "ID" INTEGER,
    "OUTLOOK" VARCHAR(20),
    "HUMIDITY" DOUBLE,
    "TEMP" INTEGER,
    "WINDY" VARCHAR(10)
);
INSERT INTO PAL_DATA_TBL_PRE VALUES (0, 'Overcast', 75, 70, 'Yes');
INSERT INTO PAL_DATA_TBL_PRE VALUES (1, 'Rain', 78, 70, 'Yes');
INSERT INTO PAL_DATA_TBL_PRE VALUES (2, 'Sunny', 66, 70, 'Yes');
```

```
INSERT INTO PAL_DATA_TBL_PRE VALUES (3, 'Sunny', 69, 70, 'Yes');
INSERT INTO PAL_DATA_TBL_PRE VALUES (4, 'Rain', null, 70, 'Yes');
INSERT INTO PAL_DATA_TBL_PRE VALUES (5, null, 70, 70, 'Yes');
INSERT INTO PAL_DATA_TBL_PRE VALUES (6, '***', 70, 70, 'Yes');
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FUNCTION',null,null,'DT');
CALL _SYS_AFL.PAL_UNIFIED_REGRESSION_PREDICT(PAL_DATA_TBL_PRE,
PAL_DATA_MODEL_TBL, #PAL_PARAMETER_TBL, ?,?);
```

### Expected Result

Result:

| | ID | SCORE | UPPER_BOUND | LOWER_BOUND | REASON |
|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | ? | [{"attr":"HUMIDITY","pct":60,"val":0.45357142857142868},{"attr":"OUTLOOK","pct":23,"val":0.17142857142857143},{"attr":"TEMP","pct":17,"val":-0.125}] |
| 2 | 1 | 0 | 0 | ? | [{"attr":"TEMP","pct":69,"val":-0.34722222222222221},{"attr":"OUTLOOK","pct":16,"val":-0.079365079365079361},{"attr":"HUMIDITY","pct":15,"val":-0.073412698412698457}] |
| 3 | 2 | 1 | 0 | ? | [{"attr":"HUMIDITY","pct":79,"val":0.50714285714285723},{"attr":"TEMP","pct":11,"val":-0.07142857142857425},{"attr":"OUTLOOK","pct":10,"val":0.06428571428571427 9}] |
| 4 | 3 | 1 | 0 | ? | [{"attr":"HUMIDITY","pct":79,"val":0.50714285714285723},{"attr":"TEMP","pct":11,"val":-0.07142857142857425},{"attr":"OUTLOOK","pct":10,"val":0.06428571428571427 9}] |
| 5 | 4 | 0.5 | 0 | ? | null |
| 6 | 5 | 0.8 | 0 | ? | [{"attr":"HUMIDITY","pct":100,"val":0.30000021428571422}] |
| 7 | 6 | 0.8 | 0 | ? | [{"attr":"HUMIDITY","pct":100,"val":0.30000021428571422}] |

### Example 2: linear SHAP(MLR)

```
DROP TABLE PAL_DATA_TBL_PRE;
CREATE COLUMN TABLE PAL_DATA_TBL_PRE(
     "ID" INTEGER,
     "X1" DOUBLE,
     "X2" VARCHAR (100),
     "X3" DOUBLE
);
INSERT INTO PAL_DATA_TBL_PRE VALUES (0, 1.69, 'B', 1);
INSERT INTO PAL_DATA_TBL_PRE VALUES (1, 0.054, 'B', 2);
INSERT INTO PAL_DATA_TBL_PRE VALUES (2, 0.123, 'A', 2);
INSERT INTO PAL_DATA_TBL_PRE VALUES (3, 1.980, 'A', 1);
INSERT INTO PAL_DATA_TBL_PRE VALUES (4, 0.563, 'A', 1);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FUNCTION',null,null,'MLR');
CALL _SYS_AFL.PAL_UNIFIED_REGRESSION_PREDICT(PAL_DATA_TBL_PRE,
PAL_DATA_MODEL_TBL, #PAL_PARAMETER_TBL, ?,?);
```

### Expected Result

Result:

| | ID | SCORE | UPPER_BOUND | LOWER_BOUND | REASON |
|---|---|---|---|---|---|
| 1 | 0 | 10.314766000000002 | ? | ? | [{"attr":"X2","pct":47.200000000000003,"val":-43.1740318713221},{"attr":"X3","pct":38.30000... |
| 2 | 1 | 1.6859316000000018 | ? | ? | [{"attr":"X2","pct":48.899999999999999,"val":78.745889925878771},{"attr":"X3","pct":37.1000... |
| 3 | 2 | -7.4095508 | ? | ? | [{"attr":"X2","pct":45.700000000000003,"val":67.491524141551693},{"attr":"X3","pct":39.3999... |
| 4 | 3 | 2.0216020000000006 | ? | ? | [{"attr":"X2","pct":51.300000000000004,"val":-62.896437592807033},{"attr":"X3","pct":35.299... |
| 5 | 4 | -3.1226748 | ? | ? | [{"attr":"X2","pct":96.299999999999997,"val":-8.6013579957359632},{"attr":"X1","pct":2.1000... |

### Example 3: kernel SHAP(SVM)

```
DROP TABLE PAL_DATA_TBL_PRE;
CREATE COLUMN TABLE PAL_DATA_TBL_PRE(
    ID INTEGER,
    ATTRIBUTE1 DOUBLE,
    ATTRIBUTE2 DOUBLE,
    ATTRIBUTE3 DOUBLE,
    ATTRIBUTE4 VARCHAR(10)
);
INSERT INTO PAL_DATA_TBL_PRE VALUES(0,1,10,100,'A');
INSERT INTO PAL_DATA_TBL_PRE VALUES(1,1.2,10.2,100,'A');
INSERT INTO PAL_DATA_TBL_PRE VALUES(2,4.1,40.1,400,'AB');
INSERT INTO PAL_DATA_TBL_PRE VALUES(3,4.2,40.3,400,'AB');
INSERT INTO PAL_DATA_TBL_PRE VALUES(4,9.1,90.1,900,'A');
INSERT INTO PAL_DATA_TBL_PRE VALUES(5,9.2,90.2,900,'A');
INSERT INTO PAL_DATA_TBL_PRE VALUES(6,4,40,400,'A');
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FUNCTION',null,null,'SVM');
CALL _SYS_AFL.PAL_UNIFIED_REGRESSION_PREDICT(PAL_DATA_TBL_PRE,
PAL_DATA_MODEL_TBL, #PAL_PARAMETER_TBL, ?,?);
```

### Expected Result

Result:

| | ID | SCORE | UPPER_BOUND | LOWER_BOUND | REASON |
|---|---|---|---|---|---|
| 1 | 0 | 0.980143 | ? | ? | [{"attr":"ATTRIBUTE1","pct":94.799999999999997,"val":-0.023070936477644466},{"attr":"ATTR... |
| 2 | 1 | 1.00507 | ? | ? | [{"attr":"ATTRIBUTE1","pct":94.799999999999997,"val":0.003342174651676188 5},{"attr":"ATTR... |
| 3 | 2 | 1.1328 | ? | ? | [{"attr":"ATTRIBUTE1","pct":51.700000000000003,"val":0.62405282899661552},{"attr":"ATTRIB... |
| 4 | 3 | 1.1645 | ? | ? | [{"attr":"ATTRIBUTE1","pct":52.700000000000003,"val":0.64860160926991406},{"attr":"ATTRIB... |
| 5 | 4 | 2.03555 | ? | ? | [{"attr":"ATTRIBUTE1","pct":79.199999999999989,"val":1.2925079895727964},{"attr":"ATTRIBU... |
| 6 | 5 | 2.04525 | ? | ? | [{"attr":"ATTRIBUTE1","pct":79.399999999999991,"val":1.3024597341997233},{"attr":"ATTRIBU... |
| 7 | 6 | 1.17406 | ? | ? | [{"attr":"ATTRIBUTE1","pct":53.5,"val":0.52977141543495121},{"attr":"ATTRIBUTE3","pct":32.7... |

## _SYS_AFL.PAL_UNIFIED_REGRESSION_SCORE

### Parameter Table

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| TOP_K_ATTRIBUTIONS | INTEGER | 10 | Outputs the attributions of top k features which contribute the most. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| FEATURE_ATTRIBUTION_METHOD | INTEGER | 2 | Controls which method to use in tree model reasoning:<br>• 0: no reasoning;<br>• 1: Saabas;<br>• 2: tree SHAP. | Valid for the functions:<br>• DT<br>• RDT<br>• HGBT |
| SAMPLESIZE | INTEGER | 0 | Number of sampled combinations of features. If it equals to 0, then it will be determined by algorithm heuristically. | It is better to use a number which is larger than the number of the columns of the data table.<br>Available for the following functions:<br>• EXP<br>• GLM<br>• MLR<br>• MLP<br>• SVM |
| SEED | INTEGER | 0 | Specifies the seed for random number generator:<br>• 0: Uses the current time(in second) as seed;<br>• Others: Uses the specified value as seed. | Valid for the functions:<br>• EXP<br>• GLM<br>• MLR<br>• MLP<br>• SVM |
| IGNORE_CORRELATION | INTEGER | 0 | Specified if ignore the correlation between the features or not:<br>• 0: Ignore the correlation;<br>• 1: Not ignore the correlation. | Valid for the functions:<br>• EXP<br>• GLM<br>• MLR |

### Examples

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

#### Example 1: tree SHAP(Decision Tree)

```
DROP TABLE PAL_DATA_TBL_SCORE;
CREATE COLUMN TABLE PAL_DATA_TBL_SCORE(
    "ID" INTEGER,
```

```
    "OUTLOOK" VARCHAR(20),
    "HUMIDITY" DOUBLE,
    "TEMP" INTEGER,
    "WINDY" VARCHAR(10),
    "SCORE" DOUBLE
);
INSERT INTO PAL_DATA_TBL_SCORE VALUES (0, 'Overcast', 75, 70, 'Yes', 1);
INSERT INTO PAL_DATA_TBL_SCORE VALUES (1, 'Rain', 78, 70, 'Yes', 1);
INSERT INTO PAL_DATA_TBL_SCORE VALUES (2, 'Sunny', 66, 70, 'Yes', 1);
INSERT INTO PAL_DATA_TBL_SCORE VALUES (3, 'Sunny', 69, 70, 'Yes', 1);
INSERT INTO PAL_DATA_TBL_SCORE VALUES (4, 'Rain', null, 70, 'Yes', 0);
INSERT INTO PAL_DATA_TBL_SCORE VALUES (5, null, 70, 70, 'Yes', 0.3);
INSERT INTO PAL_DATA_TBL_SCORE VALUES (6, '***', 70, 70, 'Yes', 0.5);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FUNCTION',null,null,'DT');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FEATURE_ATTRIBUTION_METHOD',1,null,null);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('TOP_K_ATTRIBUTIONS',1,null,null);
CALL _SYS_AFL.PAL_UNIFIED_REGRESSION_SCORE(PAL_DATA_TBL_SCORE,
PAL_DATA_MODEL_TBL, #PAL_PARAMETER_TBL, ?,?);
```

**Expected Result**

Result:

| | ID | SCORE | UPPER_BOUND | LOWER_BOUND | REASON |
|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | ? | [{"attr":"HUMIDITY","pct":60,"val":0.30000000000000004}] |
| 2 | 1 | 0 | 0 | ? | [{"attr":"TEMP","pct":60,"val":-0.5}] |
| 3 | 2 | 1 | 0 | ? | [{"attr":"HUMIDITY","pct":60,"val":0.30000000000000004}] |
| 4 | 3 | 1 | 0 | ? | [{"attr":"HUMIDITY","pct":60,"val":0.30000000000000004}] |
| 5 | 4 | 0.5 | 0 | ? | null |
| 6 | 5 | 0.8 | 0 | ? | [{"attr":"HUMIDITY","pct":100,"val":0.30000000000000004}] |
| 7 | 6 | 0.8 | 0 | ? | [{"attr":"HUMIDITY","pct":100,"val":0.30000000000000004}] |

Statistic:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | EVAR | -0.5040871934604905 |
| 2 | MAE | 0.32857142857142857 |
| 3 | MAPE | 0.6095238095238097 |
| 4 | MAX_ERROR | 1 |
| 5 | MSE | 0.22714285714285715 |
| 6 | R2 | -0.5163487738419621 |
| 7 | RMSE | 0.47659506621749365 |
| 8 | WMAPE | 0.47916666666666663 |

**Example 2: linear SHAP(MLR)**

```
DROP TABLE PAL_DATA_TBL_SCORE;
CREATE COLUMN TABLE PAL_DATA_TBL_SCORE(
     "ID" INTEGER,
     "X1" DOUBLE,
```

```
     "X2" VARCHAR (100),
     "X3" INTEGER,
     "Y" DOUBLE
);
INSERT INTO PAL_DATA_TBL_SCORE VALUES (0, 1.69, 'B', 1, 1.2);
INSERT INTO PAL_DATA_TBL_SCORE VALUES (1, 0.054, 'B', 2, 2.1);
INSERT INTO PAL_DATA_TBL_SCORE VALUES (2, 0.123, 'A', 2, 2.4);
INSERT INTO PAL_DATA_TBL_SCORE VALUES (3, 1.980, 'A', 1, 1.8);
INSERT INTO PAL_DATA_TBL_SCORE VALUES (4, 0.563, 'A', 1, 1.0);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FUNCTION',null,null,'MLR');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('TOP_K_ATTRIBUTIONS',1,null,null);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SAMPLESIZE',4,null,null);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SEED',1234,null,null);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('IGNORE_CORRELATION',1,null,null);
CALL _SYS_AFL.PAL_UNIFIED_REGRESSION_SCORE(PAL_DATA_TBL_SCORE,
PAL_DATA_MODEL_TBL, #PAL_PARAMETER_TBL, ?,?);
```

### Expected Result

Result:

| | ID | SCORE | UPPER_BOUND | LOWER_BOUND | REASON |
|---|---|---|---|---|---|
| 1 | 0 | 10.314766000000002 | ? | ? | [{"attr":"X2","pct":45.200000000000003,"val":2.3364950000000002}] |
| 2 | 1 | 1.6859316000000018 | ? | ? | [{"attr":"X1","pct":54.700000000000003,"val":-4.4508703999999994}] |
| 3 | 2 | -7.4095508 | ? | ? | [{"attr":"X2","pct":55.799999999999997,"val":-7.0094850000000006}] |
| 4 | 3 | 2.0216020000000006 | ? | ? | [{"attr":"X2","pct":64.299999999999997,"val":-7.0094850000000006}] |
| 5 | 4 | -3.1226748 | ? | ? | [{"attr":"X2","pct":64.0,"val":-7.0094850000000006}] |

Statistic:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | EVAR | -135.78813001448506 |
| 2 | MAE | 4.7365324 |
| 3 | MAPE | 3.225182723492064 |
| 4 | MAX_ERROR | 9.8095508 |
| 5 | MSE | 39.30465074507485 |
| 6 | R2 | -139.3737526609816 |
| 7 | RMSE | 6.26934213016604 |
| 8 | WMAPE | 2.786195529411765 |

### Example 3: kernel SHAP(SVM)

```
DROP TABLE PAL_DATA_TBL_SCORE;
CREATE COLUMN TABLE PAL_DATA_TBL_SCORE(
    ID INTEGER,
    ATTRIBUTE1 DOUBLE,
    ATTRIBUTE2 DOUBLE,
    ATTRIBUTE3 DOUBLE,
    ATTRIBUTE4 VARCHAR(10),
    LABEL INTEGER
);
INSERT INTO PAL_DATA_TBL_SCORE VALUES(0,1,10,100,'A',2);
INSERT INTO PAL_DATA_TBL_SCORE VALUES(1,1.2,10.2,100,'A',3);
```

```
INSERT INTO PAL_DATA_TBL_SCORE VALUES(2,4.1,40.1,400,'AB',1);
INSERT INTO PAL_DATA_TBL_SCORE VALUES(3,4.2,40.3,400,'AB',7);
INSERT INTO PAL_DATA_TBL_SCORE VALUES(4,9.1,90.1,900,'A',4);
INSERT INTO PAL_DATA_TBL_SCORE VALUES(5,9.2,90.2,900,'A',3);
INSERT INTO PAL_DATA_TBL_SCORE VALUES(6,4,40,400,'A',2);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FUNCTION',null,null,'SVM');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('TOP_K_ATTRIBUTIONS',1,null,null);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SAMPLESIZE',4,null,null);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SEED',1234,null,null);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('IGNORE_CORRELATION',1,null,null);
CALL _SYS_AFL.PAL_UNIFIED_REGRESSION_SCORE(PAL_DATA_TBL_SCORE,
PAL_DATA_MODEL_TBL, #PAL_PARAMETER_TBL, ?,?);
```

**Expected Result**

Result:

| | ID | SCORE | UPPER_BOUND | LOWER_BOUND | REASON |
|---|---|---|---|---|---|
| 1 | 0 | 0.980143 | ? | ? | [{"attr":"ATTRIBUTE1","pct":94.799999999999997,"val":-0.023070936477644466}] |
| 2 | 1 | 1.00507 | ? | ? | [{"attr":"ATTRIBUTE1","pct":94.799999999999997,"val":0.003342174651676188 5}] |
| 3 | 2 | 1.1328 | ? | ? | |
| 4 | 3 | 1.1645 | ? | ? | |
| 5 | 4 | 2.03555 | ? | ? | [{"attr":"ATTRIBUTE2","pct":null,"val":1e+9999}] |
| 6 | 5 | 2.04525 | ? | ? | |
| 7 | 6 | 1.17406 | ? | ? | |

Statistic:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | EVAR | 0.021960645834287584 |
| 2 | MAE | 1.8183181428571427 |
| 3 | MAPE | 0.48052578911564625 |
| 4 | MAX_ERROR | 5.8355 |
| 5 | MSE | 6.363334049135569 |
| 6 | R2 | -0.9487710525477682 |
| 7 | RMSE | 2.5225649742148506 |
| 8 | WMAPE | 0.5785557727272727 |

## 5.13.9.8 Missing Value Handling

Missing values in each column of the table can be handled by one of the following imputation types:

- MODE: Fills all missing values by the mode of the column, that is, the value that appears most often in the column. This type only suits column with categorical variables.
- MEAN: Fills all missing values by the mean of the column, that is, the average of all non-missing values in the column. This type only suits column with numerical variables.

- MEDIAN: Fills all missing values by the median of the column, that is, the value that separates the higher half of the non-missing values from the lower half. If there is an even number of observations, then the median is given by the mean of the two middle values. This type only suits column with numerical variables.
- ALS: Fills each missing value by the value imputed by a matrix completion model trained using alternating least squares method. This type only suits column with numerical variables.
- SPECIFIED_CATEGORICAL_VALUE: Fills all missing values by a specified categorical value. This type only suits column with categorical variables.
- SPECIFIED_NUMERICAL_VALUE: Fills all missing values by a specified numerical value. This type only suits column with numerical variables.
- DELETE: Deletes all rows with missing values. The entire row in table will be deleted.
- NONE: Does nothing. Leave the column untouched.

More detailed information is provided with the guidance of Missing Value Handling algorithm (preprocessing algorithms section).

**Parameter Table**

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| HANDLE_MISSING_VALUE | INTEGER | 0 | Whether to handle missing values:<br>• 0: No<br>• 1: Yes | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| IMPUTATION_TYPE | INTEGER | 1 | Chooses the overall default imputation type for all columns:<br>• 0: NONE<br>• 1: MODE_MEAN<br>• 2: MODE_MEDIAN<br>• 3: MODE_ALL-ZERO<br>• 4: MODE_ALS<br>• 5: DELETE<br>NONE and DELETE: set all columns to imputation type NONE/DELETE.<br>MODE_MEAN, MODE_MEDIAN, MODE_ALLZERO, and MODE_ALS: set categorical columns to the first imputation type (the type before “”); set numerical columns to the second imputation type (the type after “”). ALLZERO means to fill all missing values in numerical columns with 0. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| `<column name>`_IMPUTATION_TYPE | INTEGER | No default value | Sets imputation type of a specified column. This value overwrites the default imputation type deduced by the IMPUTATION_TYPE parameter:<br>• 0: NONE<br>• 1: DELETE<br>• 100: MODE<br>• 101: SPECIFIED_CATEGORICAL_VALUE<br>• 200: MEAN<br>• 201: MEDIAN<br>• 203:SPECIFIED_NUMERICAL_VALUE<br>• 204: ALS<br>For type 101 (SPECIFIED_CATEGORICAL_VALUE), use the following syntax to specify a categorical value to be filled in a column V0:<br>INSERT INTO #PAL_PARAMETER_TBL VALUES('V0_IMPUTATION_TYPE', 101, NULL, '`<CATEGORICAL_VALUE>`');<br>For type 203 (SPECIFIED_NUMERICAL_VALUE), use the following syntax to specify a numerical value to be filled in a column V1:<br>INSERT INTO #PAL_PARAMETER_TBL VALUES('V1_IMPUTA- | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| | | | TION_TYPE', 203, `<NUMERICAL_VALUE>`, NULL);<br><br>**Note**<br>A column of integer type can be set to type 101 (SPECIFIED_CATEGORICAL_VALUE) or type 203 (SPECIFIED_NUMERICAL_VALUE):<br>• when setting to type 101 (SPECIFIED_CATEGORICAL_VALUE), the string must contain a valid integer value, otherwise an error will be thrown;<br>• when setting to type 203 (SPECIFIED_NUMERICAL_VALUE), the double value will be converted to integer. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| ALS_FACTOR_NUMBER | INTEGER | 3 | Length of factor vectors, that is, the f in ALS model.<br>**Note**<br>ALS_FACTOR_NUMBER should be less than the number of numerical columns to get a meaningful result. | |
| ALS_REGULARIZATION | DOUBLE | 1e-2 | L2 regularization of the factors, that is, the λ in ALS model. | |
| ALS_MAX_ITERATION | INTEGER | 20 | Maximum number of iterations when training ALS model. | |
| ALS_SEED | INTEGER | 0 | Specifies the seed of the random number generator used in the training of ALS model:<br>• 0: Uses the current time as the seed<br>• Others: Uses the specified value as the seed. | |
| ALS_EXIT_THRESHOLD | DOUBLE | 0.0 | The training of ALS model exits if the value of cost function is not decreased more than the value of ALS_EXIT_THRESHOLD since the last check.<br>If ALS_EXIT_THRESHOLD is set to 0, there is no check and the training only exits on reaching the maximum number of iterations. Evaluations of cost function require additional calculations. You can set this parameter to 0 to avoid it. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| ALS_EXIT_INTERVAL | INTEGER | 5 | During the training of ALS model, it will calculate cost function and check the exit criteria every `<ALS_EXIT_INTERVAL>` iterations. Evaluations of cost function require additional calculations. | |
| ALS_LINEAR_SYSTEM_SOLVER | INTEGER | 0 | • 0: cholesky solver<br>• 1: cg solver (recommended when ALS_FACTOR_NUMBER is large) | |
| ALS_CG_MAX_ITERATION | INTEGER | 3 | Maximum number of iteration of cg solver. | |
| ALS_CENTERING | INTEGER | 1 | Whether to center the data by column before training ALS model.<br>• 0: no centering<br>• 1: centering. | |
| ALS_SCALING | INTEGER | 1 | Whether to scale the data by column before training ALS model.<br>• 0: no scaling<br>• 1: scaling | |

### Examples

Assume that:

- DM_PAL is a schema belonging to USER1;
- USER1 has been assigned the AFL__SYS_AFL_AFLPAL_EXECUTE or AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION role.

#### Example 1: Exponential Regression (Training)

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_DATA_TBL;
CREATE COLUMN TABLE PAL_DATA_TBL(
     "ID" INTEGER,
     "X1" DOUBLE,
     "X2" DOUBLE,
     "Y" DOUBLE
);
INSERT INTO PAL_DATA_TBL VALUES (0,0.5,0.13,0.33);
INSERT INTO PAL_DATA_TBL VALUES (1,0.15,0.14,0.37);
INSERT INTO PAL_DATA_TBL VALUES (2,0.25,0.15,0.36);
INSERT INTO PAL_DATA_TBL VALUES (3,0.35,null,0.35);
INSERT INTO PAL_DATA_TBL VALUES (4,0.45,0.17,0.37);
```

```
INSERT INTO PAL_DATA_TBL VALUES (5,null,0.18,0.38);
INSERT INTO PAL_DATA_TBL VALUES (6,0.65,null,0.39);
INSERT INTO PAL_DATA_TBL VALUES (7,0.75,0.19,0.31);
INSERT INTO PAL_DATA_TBL VALUES (8,0.85,0.11,0.32);
INSERT INTO PAL_DATA_TBL VALUES (9,0.95,0.12,0.33);
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO PAL_PARAMETER_TBL VALUES ('FUNCTION',null,null,'EXP');
INSERT INTO PAL_PARAMETER_TBL VALUES ('PMML_EXPORT',2,null,null);
INSERT INTO PAL_PARAMETER_TBL VALUES ('PARTITION_METHOD',0,null,null);
INSERT INTO PAL_PARAMETER_TBL VALUES ('PARTITION_TRAINING_PERCENT',
null,0.5,null);
INSERT INTO PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', null,0.5,null);
INSERT INTO PAL_PARAMETER_TBL VALUES ('HANDLE_MISSING_VALUE',1,null,null);
DROP TABLE PAL_DATA_MODEL_TBL;
CREATE COLUMN TABLE PAL_DATA_MODEL_TBL(
     "ROW_INDEX" INTEGER,
     "PART_INDEX" INTEGER,
     "MODEL_CONTENT" VARCHAR (5000)
);
DO BEGIN
  data_tab = SELECT * FROM PAL_DATA_TBL;
  control_tab = SELECT * FROM PAL_PARAMETER_TBL;
  CALL _SYS_AFL.PAL_UNIFIED_REGRESSION(:data_tab, :control_tab, model_tab,
fit_tab, stats_tab, optimal_parameter_tab, place_holder_1_tab,
place_holder_2_tab);
  INSERT into PAL_DATA_MODEL_TBL SELECT * FROM :model_tab;
END;
```

**Example 2: Exponential Regression (Prediction)**

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_DATA_TBL_PRE;
CREATE COLUMN TABLE PAL_DATA_TBL_PRE(
     "ID" INTEGER,
     "X1" DOUBLE,
     "X2" DOUBLE
);
INSERT INTO PAL_DATA_TBL_PRE VALUES (0,0.5,0.3);
INSERT INTO PAL_DATA_TBL_PRE VALUES (1,null,0.4);
INSERT INTO PAL_DATA_TBL_PRE VALUES (2,0,null);
INSERT INTO PAL_DATA_TBL_PRE VALUES (3,0.3,0.45);
INSERT INTO PAL_DATA_TBL_PRE VALUES (4,0.4,1.7);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FUNCTION',null,null,'EXP');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('HANDLE_MISSING_VALUE',1,null,null);
CALL _SYS_AFL.PAL_UNIFIED_REGRESSION_PREDICT(PAL_DATA_TBL_PRE,
PAL_DATA_MODEL_TBL, #PAL_PARAMETER_TBL, ?,?);
```

**Example 3: Exponential Regression (Scoring)**

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_DATA_TBL_SCORE;
CREATE COLUMN TABLE PAL_DATA_TBL_SCORE(
     "ID" INTEGER,
```

```
     "X1" DOUBLE,
     "X2" DOUBLE,
     "Y" DOUBLE
);
INSERT INTO PAL_DATA_TBL_SCORE VALUES (0,4,0.5,0.3);
INSERT INTO PAL_DATA_TBL_SCORE VALUES (1,3,4,0.4);
INSERT INTO PAL_DATA_TBL_SCORE VALUES (2,7,null,1.6);
INSERT INTO PAL_DATA_TBL_SCORE VALUES (3,null,0.3,1.3);
INSERT INTO PAL_DATA_TBL_SCORE VALUES (4,0.5,0.4,1.7);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FUNCTION',null,null,'EXP');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('HANDLE_MISSING_VALUE',1,null,null);
CALL _SYS_AFL.PAL_UNIFIED_REGRESSION_SCORE(PAL_DATA_TBL_SCORE,
PAL_DATA_MODEL_TBL, #PAL_PARAMETER_TBL, ?,?);
```

## 5.13.9.9 Prediction Intervals and Confidence Intervals

PAL provides interval estimates and the point estimate of response values in the unified prediction or scoring regression procedure.

PAL supports two types of interval estimates: confidence intervals and prediction intervals. You can specify the `INTERVAL` parameter to denote the desired type of interval for the response value. Except the interval type, you also need to specify the `SIGNIFICANCE_LEVEL` parameter to represent the long-run proportion of corresponding intervals that fall outside of the true estimated target. The difference between the two types of intervals is tiny and confusing. The prediction interval predicts distribution of individual future observation of the target, while the confidence interval predicts the mean statistic of the future target population.

Prediction intervals and confidence intervals are provided for Generalised Linear Models (GLM), Multiple Linear Regression (MLR), and Random Decision Trees (RDT) in the prediction and scoring procedure. The following tables list the parameters for using prediction intervals or confidence intervals in GLM, MLR, and RDT respectively.

### Unified Regression Prediction or Scoring

**Parameter Table**

Mandatory Parameters

None.

Optional Parameters

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| SIGNIFICANCE_LEVEL | DOUBLE | 0.05 | The significance level for the confidence interval and the prediction interval. Valid range is (0, 1). | |
| INTERVAL | INTEGER | 0 | Specifies the type of interval to output.<br>• 0: No interval is calculated and output;<br>• 1: Confidence Interval;<br>• 2: Prediction Interval. | |

## GLM

Currently confidence intervals and prediction intervals are only output for models obtained via the IRLS method. Since the `INTERVAL` parameter is 0, it doesn't output either interval by default. In order to output an interval, set the value to 1 or 2.

The interval target can be either response or link, which can be specified with the `TYPE` parameter. The intervals are calculated at link scale with approximate Student-t Distribution (for continuous distribution family) or Normal Distribution (for discrete distribution family), based on the standard error of each provided data instance. If the interval target is response, convert the link interval to the response interval with the inverse of link function.

### Prediction Procedures

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| TYPE | VARCHAR or NVARCHAR | response | The prediction type:<br>• response: Predicts response μ;<br>• link: Predicts linear predictor η. | |
| SIGNIFICANCE_LEVEL | DOUBLE | 0.05 | The significance level for the confidence interval and the prediction interval. Valid range is (0, 1). | Valid only for IRLS. |

## MLR

Currently confidence intervals and prediction intervals are only output for models trained with the parameter `OUTPUT_COEFCOV` set to 1 and model format as JSON. Therefore, it also requires `JSON_EXPORT` to be set to 1, which is the default. Note that `OUTPUT_COEFCOV` is 0 by default, so the trained model can't be used to calculate intervals during prediction or scoring. In order to output either interval during prediction or scoring, it requires that the model contains coefficient covariances information. Otherwise, the prediction or scoring procedure outputs error messages when outputting either interval.

The interval target is response. Intervals are calculated at response level with Student-t Distribution, based on the standard error for each provided data instance. When computing the standard error, it also requires unbiased MSE statistic in model. However, if the residual degree freedom is 0, there's no definition for it. So, in this case neither interval can be obtained and is output as a NULL value.

### Training Procedures

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| OUTPUT_COEFCOV | INTEGER | 0 | Specifies whether to output coefficient co-variance information.<br>• 0: Doesn't output related information;<br>• 1: Output related information. | Valid only when JSON_EXPORT=1. |

## RDT

Currently only prediction intervals are available in function RDT. It is output for model trained with parameter `OUTPUT_LEAF_VALUES` set to 1 and model format is JSON (MODEL_FORMAT is 1 or COMPRESSION is 1).

### Training Procedures

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| OUTPUT_LEAF_VALUES | INTEGER | 0 | Controls whether to save target values in each leaf node.<br>• 0: Doesn't output target values in each leaf node;<br>• 1: Outputs target values in each leaf node. | Valid only when JSON_EXPORT = 1 or COMPRESSION = 1. |

## Examples

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_DATA_TBL;
CREATE COLUMN TABLE PAL_DATA_TBL(
     "ID" INTEGER,
     "X1" DOUBLE,
     "X2" VARCHAR (100),
     "X3" INTEGER,
     "Y" DOUBLE
);
INSERT INTO PAL_DATA_TBL VALUES (0, 0.0, 'A', 1, -6.879);
INSERT INTO PAL_DATA_TBL VALUES (1, 0.50, 'A', 1, -3.449);
INSERT INTO PAL_DATA_TBL VALUES (2, 0.54, 'B', 1, 6.635);
```

```
INSERT INTO PAL_DATA_TBL VALUES (3, 1.04, 'B', 1, 11.844);
INSERT INTO PAL_DATA_TBL VALUES (4, 1.50, 'A', 1, 2.786);
INSERT INTO PAL_DATA_TBL VALUES (5, 0.04, 'B', 2, 2.389);
INSERT INTO PAL_DATA_TBL VALUES (6, 2.00, 'A', 2, -0.011);
INSERT INTO PAL_DATA_TBL VALUES (7, 2.04, 'B', 2, 8.839);
INSERT INTO PAL_DATA_TBL VALUES (8, 1.54, 'B', 1, 4.689);
INSERT INTO PAL_DATA_TBL VALUES (9, 1.00, 'A', 2, -5.507);
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO PAL_PARAMETER_TBL VALUES ('FUNCTION',null,null,'MLR');
INSERT INTO PAL_PARAMETER_TBL VALUES ('OUTPUT_COEFCOV',1,null,null);
DROP TABLE PAL_DATA_MODEL_TBL;
CREATE COLUMN TABLE PAL_DATA_MODEL_TBL(
     "ROW_INDEX" INTEGER,
     "PART_INDEX" INTEGER,
     "MODEL_CONTENT" VARCHAR (5000)
);
DO BEGIN
  data_tab = SELECT * FROM PAL_DATA_TBL;
  control_tab = SELECT * FROM PAL_PARAMETER_TBL;
  CALL _SYS_AFL.PAL_UNIFIED_REGRESSION(:data_tab, :control_tab, model_tab,
stats_tab, optimal_parameter_tab, partition_result, place_holder_1_tab,
place_holder_2_tab);
  INSERT into PAL_DATA_MODEL_TBL SELECT * FROM :model_tab;
END;
-------------------------------Predict-------------------------------
DROP TABLE PAL_DATA_TBL_PRE;
CREATE COLUMN TABLE PAL_DATA_TBL_PRE(
     "ID" INTEGER,
     "X1" DOUBLE,
     "X2" VARCHAR (100),
     "X3" INTEGER
);
INSERT INTO PAL_DATA_TBL_PRE VALUES (0, 1.69, 'B', 1);
INSERT INTO PAL_DATA_TBL_PRE VALUES (1, 0.054, 'B', 2);
INSERT INTO PAL_DATA_TBL_PRE VALUES (2, 0.123, 'A', 2);
INSERT INTO PAL_DATA_TBL_PRE VALUES (3, 1.980, 'A', 1);
INSERT INTO PAL_DATA_TBL_PRE VALUES (4, 0.563, 'A', 1);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FUNCTION',null,null,'MLR');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SIGNIFICANCE_LEVEL',null,0.1,null);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('INTERVAL',2,null,null);
CALL _SYS_AFL.PAL_UNIFIED_REGRESSION_PREDICT(PAL_DATA_TBL_PRE,
PAL_DATA_MODEL_TBL, #PAL_PARAMETER_TBL, ?,?);
```

**Expected Result**:

RESULT:

Result 1 × Result 2 × Messages × History

Rows (5) SQL

| | ID | SCORE | LOWER_BOUND | UPPER_BOUND | REASON |
|---|---|---|---|---|---|
| 1 | 0 | 10.314759999999998 | 1.674993006571098 | 18.9545269934289 | *NULL* |
| 2 | 1 | 1.6859255999999974 | -7.746849551875767 | 11.118700751875762 | *NULL* |
| 3 | 2 | -7.4095608 | -16.68354577101079 | 1.86442417101079 | *NULL* |
| 4 | 3 | 2.0215920000000014 | -7.040285694119818 | 11.083469694119822 | *NULL* |
| 5 | 4 | -3.1226847999999996 | -11.356559867982662 | 5.111190267982662 | *NULL* |

## 5.13.10 Unified Regression with Pivoted Input Data

The unified regression with pivoted is a unified interface to call some regression algorithms with pivot data format.

The regression algorithms include:

- GEO (Bi-Variate Geometric Regression)
- LOG (Bi-Variate Natural Logarithmic Regression)
- EXP (Exponential Regression)
- GLM (Generalised Linear Models)
- MLR (Multiple Linear Regression)
- POL(Polynomial Regression)
- DT (Decision Trees)
- HGBT (Hybrid Gradient Boosting Tree)
- MLP (Multilayer Perceptron)
- RDT (Random Decision Trees)
- SVM (Support Vector Machine)

For MLP, unified regression with pivoted just can support single class and it will throw a error if calling multiple class.

For DT, HGBT, RDT and SVM, unified regression with pivoted supports continue target column and runs successful with regression algorithm only.

See the corresponding topic for details about a specific regression method.

### Prerequisites

- All prerequisites for the original corresponding regression function apply in the unified regression interface.

# _SYS_AFL.PAL_UNIFIED_REGRESSION_PIVOT

## Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<METADATA Table>` | IN |
| 2 | `<DATA Table>` | IN |
| 3 | `<PARAMETER Table>` | IN |
| 4 | `<MODEL Table>` | OUT |
| 5 | `<STATISTICS Table>` | OUT |
| 6 | `<OPTIMAL_PARAMETER Table>` | OUT |
| 7 | `<PARTITION_TYPE Table>` | OUT |
| 8 | `<PLACE_HOLDER_1 Table>` | OUT |
| 9 | `<PLACE_HOLDER_2 Table>` | OUT |

## Input Table

| Table | Column | Data Type | Description |
|---|---|---|---|
| METADATA | 1st column | VARCHAR or NVARCHAR | The name of the variable. |
| | 2nd column | VARCHAR or NVARCHAR | The type of the variable:<br>• CONTINUOUS;<br>• CATEGORICAL;<br>• TARGET. |
| DATA | 1st column | INTEGER, VRACHAR or NVARCHAR | Record ID. |
| | 2nd column | VARCHAR or NVARCHAR | Variable name. |
| | 3rd column | VARCHAR or NVARCHAR | Variable value. |
| | 4th column | INTEGER | Self-defined data partition:<br>• 1: training;<br>• 2: validation. |

## Parameter Table

Mandatory Parameters

The following parameters are mandatory and must be given a value.

| Name | Data Type | Description |
| --- | --- | --- |
| FUNCTION | VARCHAR or NVARCHAR | The algorithm name, including:<br>• GEO;<br>• LOG;<br>• EXP;<br>• GLM;<br>• MLR;<br>• POL;<br>• DT;<br>• HGBT;<br>• MLP;<br>• RDT;<br>• SVM. |

Optional Parameters

The following parameters refer to partition processing only. See Parameter Tables [page 2030]for details about parameters for a specific algorithm.

| Name | Data Type | Default Value | Description | Dependency |
| --- | --- | --- | --- | --- |
| PARTITION_METHOD | INTEGER | 0 | Data set Parition via different algorithm:<br>• 0: Do not use partition;<br>• 1: Data set partition with specific flag;<br>• 2: Data set partition with random. | If the parameter is set to be 2, the 4th column in data table is invalid. |
| PARTITION_TRAINING_PERCENT | DOUBLE | 0.8 | Specific scale value to data partition with random.<br>Value: 0-1. | Valid only for partition_method=2. |
| PARTITION_RANDOM_SEED | INTEGER | 0 | Indicates the seed used to initialize the random number generator.<br>• 0: Uses the system time;<br>• Not 0: Uses the specific seed. | Valid only for partition_method=2. |
| OUTPUT_PARTITION_RESULT | INTEGER | 0 | Output partition result or not:<br>• 0: Do not output result;<br>• 1: Output result. | |

### Output Tables

| Table | Column | Column Name | Column Data Type |
|---|---|---|---|
| MODEL | 1st column | ROW_INDEX | INTEGER |
| | 2st column | PART_INDEX | INTEGER |
| | 3rd column | MODEL_CONTENT | VARCHAR, NVARCHAR, NCLOB |
| STATISTICS | 1st column | STAT_NAME | VARCHAR, NVARCHAR |
| | 2nd column | STAT_VALUE | VARCHAR, NVARCHAR |
| OPTIMAL_PARAMETER | 1st column | PARAM_NAME | VARCHAR, NVARCHAR |
| | 2nd column | INT_VALUE | INTEGER |
| | 3rd column | DOUBLE_VALUE | DOUBLE |
| | 4th column | STRING_VALUE | VARCHAR, NVARCHAR |
| PARTITION_TYPE | 1st column | ID column name | ID column data type |
| | 2nd column | TYPE | INTEGER |
| PLCAE_HOLDER_1 | 1st column | OBJECT | VARCHAR, NVARCHAR |
| | 2nd column | KEY | VARCHAR, NVARCHAR |
| | 3rd column | VALUE | VARCHAR, NVARCHAR |
| PLACE_HOLDER_2 | 1st column | OJBECT | VARCHAR, NVARCHAR |
| | 2nd column | KEY | VARCHAR, NVARCHAR |
| | 3rd column | VALUE | VARCHAR, NVARCHAR |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

Example:

```
SET SCHEMA DM_PAL;
DROP TABLE META_DATA_TAB;
CREATE COLUMN TABLE META_DATA_TAB(
    "VARIABLE_NAME" NVARCHAR(50),
    "VARIABLE_TYPE" NVARCHAR(50)
);
INSERT INTO META_DATA_TAB VALUES ('X1',    'CONTINUOUS');
INSERT INTO META_DATA_TAB VALUES ('Y' ,    'TARGET');
INSERT INTO META_DATA_TAB VALUES ('X2',    'CATEGORICAL');
INSERT INTO META_DATA_TAB VALUES ('X3',    'CONTINUOUS');
DROP TABLE PAL_DATA_TBL;
CREATE COLUMN TABLE PAL_DATA_TBL(
    "ID" INT,
    "VARIABLE_NAME" NVARCHAR(50),
    "VALUE" NVARCHAR(50),
    "PURPOSE" INT
);
INSERT INTO PAL_DATA_TBL VALUES (0,    'Y',    '-6.879',   1);
```

```
INSERT INTO PAL_DATA_TBL VALUES (0,    'X1',   '0.00',     1);
INSERT INTO PAL_DATA_TBL VALUES (0,    'X2',   'A',        1);
INSERT INTO PAL_DATA_TBL VALUES (0,    'X3',   '1',        1);
INSERT INTO PAL_DATA_TBL VALUES (1,    'Y',    '-3.449',   1);
INSERT INTO PAL_DATA_TBL VALUES (1,    'X1',   '0.50',     1);
INSERT INTO PAL_DATA_TBL VALUES (1,    'X2',   'A',        1);
INSERT INTO PAL_DATA_TBL VALUES (1,    'X3',   '1',        1);
INSERT INTO PAL_DATA_TBL VALUES (2,    'Y',    '6.635',    1);
INSERT INTO PAL_DATA_TBL VALUES (2,    'X1',   '0.54',     1);
INSERT INTO PAL_DATA_TBL VALUES (2,    'X2',   'B',        1);
INSERT INTO PAL_DATA_TBL VALUES (2,    'X3',   '1',        1);
INSERT INTO PAL_DATA_TBL VALUES (3,    'Y',    '11.844',   2);
INSERT INTO PAL_DATA_TBL VALUES (3,    'X1',   '1.04',     2);
INSERT INTO PAL_DATA_TBL VALUES (3,    'X2',   'B',        2);
INSERT INTO PAL_DATA_TBL VALUES (3,    'X3',   '1',        2);
INSERT INTO PAL_DATA_TBL VALUES (4,    'Y',    '2.786',    1);
INSERT INTO PAL_DATA_TBL VALUES (4,    'X1',   '1.50',     1);
INSERT INTO PAL_DATA_TBL VALUES (4,    'X2',   'A',        1);
INSERT INTO PAL_DATA_TBL VALUES (4,    'X3',   '1',        1);
INSERT INTO PAL_DATA_TBL VALUES (5,    'Y',    '2.389',    1);
INSERT INTO PAL_DATA_TBL VALUES (5,    'X1',   '0.04',     1);
INSERT INTO PAL_DATA_TBL VALUES (5,    'X2',   'B',        1);
INSERT INTO PAL_DATA_TBL VALUES (5,    'X3',   '2',        1);
INSERT INTO PAL_DATA_TBL VALUES (6,    'Y',    '-0.011',   2);
INSERT INTO PAL_DATA_TBL VALUES (6,    'X1',   '2.00',     2);
INSERT INTO PAL_DATA_TBL VALUES (6,    'X2',   'A',        2);
INSERT INTO PAL_DATA_TBL VALUES (6,    'X3',   '2',        2);
INSERT INTO PAL_DATA_TBL VALUES (7,    'Y',    '8.839',    1);
INSERT INTO PAL_DATA_TBL VALUES (7,    'X1',   '2.04',     1);
INSERT INTO PAL_DATA_TBL VALUES (7,    'X2',   'B',        1);
INSERT INTO PAL_DATA_TBL VALUES (7,    'X3',   '2',        1);
INSERT INTO PAL_DATA_TBL VALUES (8,    'Y',    '4.689',    1);
INSERT INTO PAL_DATA_TBL VALUES (8,    'X1',   '1.54',     1);
INSERT INTO PAL_DATA_TBL VALUES (8,    'X2',   'B',        1);
INSERT INTO PAL_DATA_TBL VALUES (8,    'X3',   '1',        1);
INSERT INTO PAL_DATA_TBL VALUES (9,    'Y',    '-5.507',   1);
INSERT INTO PAL_DATA_TBL VALUES (9,    'X1',   '1.00',     1);
INSERT INTO PAL_DATA_TBL VALUES (9,    'X2',   'A',        1);
INSERT INTO PAL_DATA_TBL VALUES (9,    'X3',   '2',        1);
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO PAL_PARAMETER_TBL VALUES ('FUNCTION',null,null,'MLR');
INSERT INTO PAL_PARAMETER_TBL VALUES ('PARTITION_METHOD',1,null,null);
INSERT INTO PAL_PARAMETER_TBL VALUES ('OUTPUT_PARTITION_RESULT',1,null,null);
INSERT INTO PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', null,0.5,null);
--CALL _SYS_AFL.PAL_UNIFIED_REGRESSION_PIVOT(META_DATA_TAB, PAL_DATA_TBL,
PAL_PARAMETER_TBL, ?, ?, ?, ?, ?, ?);
DROP TABLE PAL_DATA_MODEL_TBL;
CREATE COLUMN TABLE PAL_DATA_MODEL_TBL(
     "ROW_INDEX" INTEGER,
     "PART_INDEX" INTEGER,
     "MODEL_CONTENT" VARCHAR (5000)
);
DO BEGIN
  meta_data_tab = SELECT * FROM META_DATA_TAB;
  data_tab = SELECT * FROM PAL_DATA_TBL;
  control_tab = SELECT * FROM PAL_PARAMETER_TBL;
  CALL
_SYS_AFL.PAL_UNIFIED_REGRESSION_PIVOT(:meta_data_tab, :data_tab, :control_tab,
model_tab, stats_tab, optimal_parameter_tab, partition_result,
place_holder_1_tab, place_holder_2_tab);
  INSERT into PAL_DATA_MODEL_TBL SELECT * FROM :model_tab;
END
```

**Expected Result**:

MODEL:

| | ROW_INDEX | PART_INDEX | MODEL_CONTENT |
|---|---|---|---|
| 1 | 1 | ? | <PMML version="4.0" xmlns="http://www.dmg.org/PMML-4_0" xmlns:xsi="http://www.... |
| 2 | -1 | ? | MLR |
| 3 | -2 | ? | {"feature":["X1","X2","X3","Y"],"featureNum":4,"index":[0,1,2,3],"type":["CONTINUOUS","C... |

STATISTIC:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | TEST_EVAR | 0.8267792311153285 |
| 2 | TEST_MAE | 2.8427835999999997 |
| 3 | TEST_MAPE | 17.30461011789628 |
| 4 | TEST_MAX_ERROR | 5.309797199999999 |
| 5 | TEST_MSE | 14.167574699013917 |
| 6 | TEST_R2 | 0.5967704177761926 |
| 7 | TEST_RMSE | 3.7639838866570505 |
| 8 | TEST_WMAPE | 0.48048400236626376 |

OPTIMAL_PARAMETER:

PARTITION_TYPE:

| | ID | TYPE |
|---|---|---|
| 1 | 0 | 1 |
| 2 | 1 | 1 |
| 3 | 2 | 1 |
| 4 | 4 | 1 |
| 5 | 5 | 1 |
| 6 | 7 | 1 |
| 7 | 8 | 1 |
| 8 | 9 | 1 |
| 9 | 3 | 2 |
| 10 | 6 | 2 |

## _SYS_AFL.PAL_UNIFIED_REGRESSION_PREDICT_PIVOT

This function uses the trained model to make calculated predictions or predicted values.

**Overview of All Tables**

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA Table>` | IN |

| Position | Table Name | Parameter Type |
|---|---|---|
| 2 | `<MODEL Table>` | IN |
| 3 | `<PARAMETER Table>` | IN |
| 4 | `<RESULT Table>` | OUT |
| 5 | `<PLACE_HOLDER_1 Table>` | OUT |

### Input Table

| Table | Column | Data Type | Description |
|---|---|---|---|
| DATA | 1st column | INTEGER, VRACHAR or NVARCHAR | Record ID. |
| | 2nd column | VARCHAR or NVARCHAR | Variable name. |
| | 3rd column | VARCHAR or NVARCHAR | Variable value. |

### Model Table

| Table | Column | Data Type | Description |
|---|---|---|---|
| MODEL | 1st column | INTEGER | |
| | 2nd column | INTEGER | |
| | 3rd column | VARCHAR, NVARCHAR, or NCLOB | |

### Parameter Table

See more details in Parameter Tables [page 2030].

Mandatory parameters

None.

### Output Tables

| Table | Column | Column Name | Column Data | Description |
|---|---|---|---|---|
| RESULT | 1st column | ID column name | ID column data type | |
| | 2nd column | SCORE | DOUBLE | |
| | 3rd column | LOWER_BOUND | DOUBLE | |
| | 4th column | UPPER_BOUND | DOUBLE | |
| | 5th column | REASON | NCLOB | |
| PLACE_HOLDER_1 | 1st column | OBJECT | VARCHAR, NVARCHAR | |
| | 2nd column | KEY | VARCHAR, NVARCHAR | |
| | 3rd column | VALUE | VARCHAR, NVARCHAR | |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA DM_PAL;
DROP TABLE META_DATA_TAB;
CREATE COLUMN TABLE META_DATA_TAB(
    "VARIABLE_NAME" NVARCHAR(50),
    "VARIABLE_TYPE" NVARCHAR(50)
);
INSERT INTO META_DATA_TAB VALUES ('X1',    'CONTINUOUS');
INSERT INTO META_DATA_TAB VALUES ('Y' ,    'TARGET');
INSERT INTO META_DATA_TAB VALUES ('X2',    'CATEGORICAL');
INSERT INTO META_DATA_TAB VALUES ('X3',    'CONTINUOUS');
DROP TABLE PAL_DATA_TBL;
CREATE COLUMN TABLE PAL_DATA_TBL(
    "ID" INT,
    "VARIABLE_NAME" NVARCHAR(50),
    "VALUE" NVARCHAR(50),
    "PURPOSE" INT
);
INSERT INTO PAL_DATA_TBL VALUES (0,    'Y',    '-6.879',   1);
INSERT INTO PAL_DATA_TBL VALUES (0,    'X1',   '0.00',     1);
INSERT INTO PAL_DATA_TBL VALUES (0,    'X2',   'A',        1);
INSERT INTO PAL_DATA_TBL VALUES (0,    'X3',   '1',        1);
INSERT INTO PAL_DATA_TBL VALUES (1,    'Y',    '-3.449',   1);
INSERT INTO PAL_DATA_TBL VALUES (1,    'X1',   '0.50',     1);
INSERT INTO PAL_DATA_TBL VALUES (1,    'X2',   'A',        1);
INSERT INTO PAL_DATA_TBL VALUES (1,    'X3',   '1',        1);
INSERT INTO PAL_DATA_TBL VALUES (2,    'Y',    '6.635',    1);
INSERT INTO PAL_DATA_TBL VALUES (2,    'X1',   '0.54',     1);
INSERT INTO PAL_DATA_TBL VALUES (2,    'X2',   'B',        1);
INSERT INTO PAL_DATA_TBL VALUES (2,    'X3',   '1',        1);
INSERT INTO PAL_DATA_TBL VALUES (3,    'Y',    '11.844',   2);
INSERT INTO PAL_DATA_TBL VALUES (3,    'X1',   '1.04',     2);
INSERT INTO PAL_DATA_TBL VALUES (3,    'X2',   'B',        2);
INSERT INTO PAL_DATA_TBL VALUES (3,    'X3',   '1',        2);
INSERT INTO PAL_DATA_TBL VALUES (4,    'Y',    '2.786',    1);
INSERT INTO PAL_DATA_TBL VALUES (4,    'X1',   '1.50',     1);
INSERT INTO PAL_DATA_TBL VALUES (4,    'X2',   'A',        1);
INSERT INTO PAL_DATA_TBL VALUES (4,    'X3',   '1',        1);
INSERT INTO PAL_DATA_TBL VALUES (5,    'Y',    '2.389',    1);
INSERT INTO PAL_DATA_TBL VALUES (5,    'X1',   '0.04',     1);
INSERT INTO PAL_DATA_TBL VALUES (5,    'X2',   'B',        1);
INSERT INTO PAL_DATA_TBL VALUES (5,    'X3',   '2',        1);
INSERT INTO PAL_DATA_TBL VALUES (6,    'Y',    '-0.011',   2);
INSERT INTO PAL_DATA_TBL VALUES (6,    'X1',   '2.00',     2);
INSERT INTO PAL_DATA_TBL VALUES (6,    'X2',   'A',        2);
INSERT INTO PAL_DATA_TBL VALUES (6,    'X3',   '2',        2);
INSERT INTO PAL_DATA_TBL VALUES (7,    'Y',    '8.839',    1);
INSERT INTO PAL_DATA_TBL VALUES (7,    'X1',   '2.04',     1);
INSERT INTO PAL_DATA_TBL VALUES (7,    'X2',   'B',        1);
INSERT INTO PAL_DATA_TBL VALUES (7,    'X3',   '2',        1);
INSERT INTO PAL_DATA_TBL VALUES (8,    'Y',    '4.689',    1);
INSERT INTO PAL_DATA_TBL VALUES (8,    'X1',   '1.54',     1);
INSERT INTO PAL_DATA_TBL VALUES (8,    'X2',   'B',        1);
INSERT INTO PAL_DATA_TBL VALUES (8,    'X3',   '1',        1);
INSERT INTO PAL_DATA_TBL VALUES (9,    'Y',    '-5.507',   1);
INSERT INTO PAL_DATA_TBL VALUES (9,    'X1',   '1.00',     1);
INSERT INTO PAL_DATA_TBL VALUES (9,    'X2',   'A',        1);
INSERT INTO PAL_DATA_TBL VALUES (9,    'X3',   '2',        1);
DROP TABLE PAL_PARAMETER_TBL;
```

```
CREATE COLUMN TABLE PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO PAL_PARAMETER_TBL VALUES ('FUNCTION',null,null,'MLR');
INSERT INTO PAL_PARAMETER_TBL VALUES ('PARTITION_METHOD',1,null,null);
INSERT INTO PAL_PARAMETER_TBL VALUES ('OUTPUT_PARTITION_RESULT',1,null,null);
INSERT INTO PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', null,0.5,null);
--CALL _SYS_AFL.PAL_UNIFIED_REGRESSION_PIVOT(META_DATA_TAB, PAL_DATA_TBL,
PAL_PARAMETER_TBL, ?, ?, ?, ?, ?, ?);
DROP TABLE PAL_DATA_MODEL_TBL;
CREATE COLUMN TABLE PAL_DATA_MODEL_TBL(
     "ROW_INDEX" INTEGER,
     "PART_INDEX" INTEGER,
     "MODEL_CONTENT" VARCHAR (5000)
);
DO BEGIN
  meta_data_tab = SELECT * FROM META_DATA_TAB;
  data_tab = SELECT * FROM PAL_DATA_TBL;
  control_tab = SELECT * FROM PAL_PARAMETER_TBL;
  CALL
_SYS_AFL.PAL_UNIFIED_REGRESSION_PIVOT(:meta_data_tab, :data_tab, :control_tab,
model_tab, stats_tab, optimal_parameter_tab, partition_result,
place_holder_1_tab, place_holder_2_tab);
  INSERT into PAL_DATA_MODEL_TBL SELECT * FROM :model_tab;
END
-------------------------------Predict------------------------------
DROP TABLE PAL_DATA_TBL_PRE;
CREATE COLUMN TABLE PAL_DATA_TBL_PRE(
    "ID" INT,
    "VARIABLE_NAME" NVARCHAR(50),
    "VALUE" NVARCHAR(50)
);
INSERT INTO PAL_DATA_TBL_PRE VALUES (0,    'X1',    '1.69');
INSERT INTO PAL_DATA_TBL_PRE VALUES (0,    'X2',    'B');
INSERT INTO PAL_DATA_TBL_PRE VALUES (0,    'X3',    '1');
INSERT INTO PAL_DATA_TBL_PRE VALUES (1,    'X1',    '0.054');
INSERT INTO PAL_DATA_TBL_PRE VALUES (1,    'X2',    'B');
INSERT INTO PAL_DATA_TBL_PRE VALUES (1,    'X3',    '2');
INSERT INTO PAL_DATA_TBL_PRE VALUES (2,    'X1',    '0.123');
INSERT INTO PAL_DATA_TBL_PRE VALUES (2,    'X2',    'A');
INSERT INTO PAL_DATA_TBL_PRE VALUES (2,    'X3',    '2');
INSERT INTO PAL_DATA_TBL_PRE VALUES (3,    'X1',    '1.980');
INSERT INTO PAL_DATA_TBL_PRE VALUES (3,    'X2',    'A');
INSERT INTO PAL_DATA_TBL_PRE VALUES (3,    'X3',    '1');
INSERT INTO PAL_DATA_TBL_PRE VALUES (4,    'X1',    '0.563');
INSERT INTO PAL_DATA_TBL_PRE VALUES (4,    'X2',    'A');
INSERT INTO PAL_DATA_TBL_PRE VALUES (4,    'X3',    '1');
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
CALL _SYS_AFL.PAL_UNIFIED_REGRESSION_PREDICT_PIVOT(PAL_DATA_TBL_PRE,
PAL_DATA_MODEL_TBL, #PAL_PARAMETER_TBL, ?,?);
```

**Expected Result**:

RESULT:

| | ID | SCORE | UPPER_BOUND | LOWER_BOUND | REASON |
|---|---|---|---|---|---|
| 1 | 0 | 8.7284858 | ? | ? | ? |
| 2 | 1 | 1.4132442800000002 | ? | ? | ? |
| 3 | 2 | -6.72318414 | ? | ? | ? |
| 4 | 3 | 1.3381136000000005 | ? | ? | ? |
| 5 | 4 | -3.44542334 | ? | ? | ? |

## _SYS_AFL.PAL_UNIFIED_REGRESSION_SCORE_PIVOT

This function makes calculated predictions, and provides the evaluation statistics for the trained model.

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA Table>` | IN |
| 2 | `<MODEL Table>` | IN |
| 3 | `<PARAMETER Table>` | IN |
| 4 | `<RESULT Table>` | OUT |
| 5 | `<STATISTIC Table>` | OUT |

### Input Table

| Table | Column | Data Type | Description |
|---|---|---|---|
| DATA | 1st column | INTEGER, VRACHAR or NVARCHAR | Record ID. |
| | 2nd column | VARCHAR or NVARCHAR | Variable name. |
| | 3rd column | VARCHAR or NVARCHAR | Variable value. |

### Model Table

| Table | Column | Data Type | Description |
|---|---|---|---|
| MODEL | 1st column | INTEGER | |
| | 2nd column | INTEGER | |
| | 3rd column | VARCHAR, NVARCHAR, or NCLOB | |

### Parameter Table

See details in Parameter Tables [page 2030].

Mandatory parameters

None.

### Output Tables

| Table | Column | Column Name | Column Data Type | Description |
|---|---|---|---|---|
| RESULT | 1st column | ID column name | ID column data type | |
| | 2nd column | SCORE | DOUBLE | |
| | 3rd column | LOWER_BOUND | DOUBLE | |
| | 4th column | UPPER_BOUND | DOUBLE | |
| | 5th column | REASON | NCLOB | |
| STATISTICS | 1st column | STAT_NAME | NVARCHAR | |
| | 2nd column | STAT_VALUE | NVARCHAR | |

### Example

```
SET SCHEMA DM_PAL;
DROP TABLE META_DATA_TAB;
CREATE COLUMN TABLE META_DATA_TAB(
    "VARIABLE_NAME" NVARCHAR(50),
    "VARIABLE_TYPE" NVARCHAR(50)
);
INSERT INTO META_DATA_TAB VALUES ('X1',    'CONTINUOUS');
INSERT INTO META_DATA_TAB VALUES ('Y' ,    'TARGET');
INSERT INTO META_DATA_TAB VALUES ('X2',    'CATEGORICAL');
INSERT INTO META_DATA_TAB VALUES ('X3',    'CONTINUOUS');
DROP TABLE PAL_DATA_TBL;
CREATE COLUMN TABLE PAL_DATA_TBL(
    "ID" INT,
    "VARIABLE_NAME" NVARCHAR(50),
    "VALUE" NVARCHAR(50),
    "PURPOSE" INT
);
INSERT INTO PAL_DATA_TBL VALUES (0,    'Y',    '-6.879',   1);
INSERT INTO PAL_DATA_TBL VALUES (0,    'X1',   '0.00',     1);
INSERT INTO PAL_DATA_TBL VALUES (0,    'X2',   'A',        1);
INSERT INTO PAL_DATA_TBL VALUES (0,    'X3',   '1',        1);
INSERT INTO PAL_DATA_TBL VALUES (1,    'Y',    '-3.449',   1);
INSERT INTO PAL_DATA_TBL VALUES (1,    'X1',   '0.50',     1);
INSERT INTO PAL_DATA_TBL VALUES (1,    'X2',   'A',        1);
INSERT INTO PAL_DATA_TBL VALUES (1,    'X3',   '1',        1);
INSERT INTO PAL_DATA_TBL VALUES (2,    'Y',    '6.635',    1);
INSERT INTO PAL_DATA_TBL VALUES (2,    'X1',   '0.54',     1);
INSERT INTO PAL_DATA_TBL VALUES (2,    'X2',   'B',        1);
INSERT INTO PAL_DATA_TBL VALUES (2,    'X3',   '1',        1);
INSERT INTO PAL_DATA_TBL VALUES (3,    'Y',    '11.844',   2);
INSERT INTO PAL_DATA_TBL VALUES (3,    'X1',   '1.04',     2);
INSERT INTO PAL_DATA_TBL VALUES (3,    'X2',   'B',        2);
INSERT INTO PAL_DATA_TBL VALUES (3,    'X3',   '1',        2);
INSERT INTO PAL_DATA_TBL VALUES (4,    'Y',    '2.786',    1);
INSERT INTO PAL_DATA_TBL VALUES (4,    'X1',   '1.50',     1);
INSERT INTO PAL_DATA_TBL VALUES (4,    'X2',   'A',        1);
INSERT INTO PAL_DATA_TBL VALUES (4,    'X3',   '1',        1);
INSERT INTO PAL_DATA_TBL VALUES (5,    'Y',    '2.389',    1);
INSERT INTO PAL_DATA_TBL VALUES (5,    'X1',   '0.04',     1);
INSERT INTO PAL_DATA_TBL VALUES (5,    'X2',   'B',        1);
INSERT INTO PAL_DATA_TBL VALUES (5,    'X3',   '2',        1);
INSERT INTO PAL_DATA_TBL VALUES (6,    'Y',    '-0.011',   2);
INSERT INTO PAL_DATA_TBL VALUES (6,    'X1',   '2.00',     2);
INSERT INTO PAL_DATA_TBL VALUES (6,    'X2',   'A',        2);
INSERT INTO PAL_DATA_TBL VALUES (6,    'X3',   '2',        2);
```

```
INSERT INTO PAL_DATA_TBL VALUES (7,    'Y',    '8.839',    1);
INSERT INTO PAL_DATA_TBL VALUES (7,    'X1',   '2.04',     1);
INSERT INTO PAL_DATA_TBL VALUES (7,    'X2',   'B',        1);
INSERT INTO PAL_DATA_TBL VALUES (7,    'X3',   '2',        1);
INSERT INTO PAL_DATA_TBL VALUES (8,    'Y',    '4.689',    1);
INSERT INTO PAL_DATA_TBL VALUES (8,    'X1',   '1.54',     1);
INSERT INTO PAL_DATA_TBL VALUES (8,    'X2',   'B',        1);
INSERT INTO PAL_DATA_TBL VALUES (8,    'X3',   '1',        1);
INSERT INTO PAL_DATA_TBL VALUES (9,    'Y',    '-5.507',   1);
INSERT INTO PAL_DATA_TBL VALUES (9,    'X1',   '1.00',     1);
INSERT INTO PAL_DATA_TBL VALUES (9,    'X2',   'A',        1);
INSERT INTO PAL_DATA_TBL VALUES (9,    'X3',   '2',        1);
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO PAL_PARAMETER_TBL VALUES ('FUNCTION',null,null,'MLR');
INSERT INTO PAL_PARAMETER_TBL VALUES ('PARTITION_METHOD',1,null,null);
INSERT INTO PAL_PARAMETER_TBL VALUES ('OUTPUT_PARTITION_RESULT',1,null,null);
INSERT INTO PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', null,0.5,null);
--CALL _SYS_AFL.PAL_UNIFIED_REGRESSION_PIVOT(META_DATA_TAB, PAL_DATA_TBL,
PAL_PARAMETER_TBL, ?, ?, ?, ?, ?, ?);
DROP TABLE PAL_DATA_MODEL_TBL;
CREATE COLUMN TABLE PAL_DATA_MODEL_TBL(
     "ROW_INDEX" INTEGER,
     "PART_INDEX" INTEGER,
     "MODEL_CONTENT" VARCHAR (5000)
);
DO BEGIN
  meta_data_tab = SELECT * FROM META_DATA_TAB;
  data_tab = SELECT * FROM PAL_DATA_TBL;
  control_tab = SELECT * FROM PAL_PARAMETER_TBL;
  CALL
_SYS_AFL.PAL_UNIFIED_REGRESSION_PIVOT(:meta_data_tab, :data_tab, :control_tab,
model_tab, stats_tab, optimal_parameter_tab, partition_result,
place_holder_1_tab, place_holder_2_tab);
  INSERT into PAL_DATA_MODEL_TBL SELECT * FROM :model_tab;
END
-------------------------------Score-------------------------------
DROP TABLE PAL_DATA_TBL_SCORE;
CREATE COLUMN TABLE PAL_DATA_TBL_SCORE(
    "ID" INT,
    "VARIABLE_NAME" NVARCHAR(50),
    "VALUE" NVARCHAR(50)
);
INSERT INTO PAL_DATA_TBL_SCORE VALUES (0,    'Y',    '4');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (0,    'X1',    '1.69');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (0,    'X2',    'B');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (0,    'X3',    '1');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (1,    'Y',    '3');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (1,    'X1',    '0.054');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (1,    'X2',    'B');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (1,    'X3',    '2');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (2,    'Y',    '7');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (2,    'X1',    '0.123');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (2,    'X2',    'A');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (2,    'X3',    '2');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (3,    'Y',    '9');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (3,    'X1',    '1.980');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (3,    'X2',    'A');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (3,    'X3',    '1');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (4,    'Y',    '0.5');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (4,    'X1',    '0.563');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (4,    'X2',    'A');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (4,    'X3',    '1');
```

```
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
CALL _SYS_AFL.PAL_UNIFIED_REGRESSION_SCORE_PIVOT(PAL_DATA_TBL_SCORE,
PAL_DATA_MODEL_TBL, #PAL_PARAMETER_TBL, ?,?);
```

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

**Expected Result**

RESULT:

| | ID | SCORE | UPPER_BOUND | LOWER_BOUND | REASON |
|---|---|---|---|---|---|
| 1 | 0 | 8.7284858 | ? | ? | ? |
| 2 | 1 | 1.4132442800000002 | ? | ? | ? |
| 3 | 2 | -6.72318414 | ? | ? | ? |
| 4 | 3 | 1.3381136000000005 | ? | ? | ? |
| 5 | 4 | -3.44542334 | ? | ? | ? |

STATISTICS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | EVAR | -3.2188565928568202 |
| 2 | MAE | 6.32914708 |
| 3 | MAPE | 2.48273245831746 |
| 4 | MAX_ERROR | 13.72318414 |
| 5 | MSE | 57.49460463088472 |
| 6 | R2 | -5.416808552554099 |
| 7 | RMSE | 7.582519675601556 |
| 8 | WMAPE | 1.3466270382978724 |

## 5.13.10.1 Parameters Specific to Pivoted Unified Regression

This topic lists mapping parameters between the pivoted unified regression procedures and original procedures.

| Algorithm | New Name | Original Name |
|---|---|---|
| GEO | DECOMPOSITION | ALG |
| | FEATURES | SELECTED_FEATURES |

| Algorithm | New Name | Original Name |
|---|---|---|
| LOG | DECOMPOSITION | ALG |
| | FEATURES | SELECTED_FEATURES |
| EXP | DECOMPOSITION | ALG |
| | FEATURES | SELECTED_FEATURES |
| GEO | DECOMPOSITION | ALG |
| | FEATURES | SELECTED_FEATURES |
| GLM | TOL | CONVERGENCE_CRITERION |
| | ALPHA | ENET_ALPHA |
| | ALPHA_RANGE | ENET_ALPHA_RANGE |
| | ALPHA_VALUES | ENET_ALPHA_VALUES |
| | LAMBDA | ENET_LAMBDA |
| | LAMBDA_RANGE | ENET_LAMBDA_RANGE |
| | LAMBDA_VALUES | ENET_LAMBDA_VALUES |
| | NUM_LAMBDA | ENET_NUM_LAMBDA |
| | HANDLE_MISSING_FIT | HANDLE_MISSING |
| | MAX_ITER | MAX_ITERATION |
| | QUASILIKELIHOOD | QUASI |
| MLR | SOLVER | ALG |
| | ALPHA | ENET_ALPHA |
| | ALPHA_RANGE | ENET_ALPHA_RANGE |
| | ALPHA_VALUES | ENET_ALPHA_VALUES |
| | LAMBDA | ENET_LAMBDA |
| | LAMBDA_RANGE | ENET_LAMBDA_RANGE |
| | LAMBDA_VALUES | ENET_LAMBDA_VALUES |
| | MAX_ITER | MAX_ITERATION |
| | INTERCEPT | NO_INTERCEPT |
| | TOL | THRESHOLD |
| | VAR_SELECT | VARIABLE_SELECTION |
| DT | ALLOW_MISSING_LABEL | ALLOW_MISSING_DEPENDENT |
| | MIN_RECORDS_LEAF | MIN_RECORDS_OF_LEAF |
| | MIN_RECORDS_LEAF_RANGE | MIN_RECORDS_OF_LEAF_RANGE |
| | MIN_RECORDS_LEAF_VALUES | MIN_RECORDS_OF_LEAF_VALUES |
| | MIN_RECORDS_PARENT | MIN_RECORDS_OF_PARENT |
| | MIN_RECORDS_PARENT_RANGE | MIN_RECORDS_OF_ PARENT_RANGE |

| Algorithm | New Name | Original Name |
|---|---|---|
| | MIN_RECORDS_PARENT_VALUES | MIN_RECORDS_OF_PARENT_VALUES |
| POL | DEGREE | POLYNOMIAL_NUM |
| | DECOMPOSITION | ALG |
| | LABEL | DEPENDENT_VARIABLE |
| | DEGREE_RANGE | POLYNOMIAL_NUM _RANGE |
| | DEGREE_VALUES | POLYNOMIAL_NUM _VALUES |
| | FEATURES | SELECTED_FEATURES |
| HGBT | COL_SUBSAMPLE_SPLIT | COL_SAMPLE_RATE_BYSPLIT |
| | COL_SUBSAMPLE_SPLIT_RANGE | COL_SAMPLE_RATE_BYSPLIT_RANGE |
| | COL_SUBSAMPLE_SPLIT_VALUES | COL_SAMPLE_RATE_BYSPLIT_VAL-UES |
| | COL_SUBSAMPLE_TREE | COL_SAMPLE_RATE_BYTREE |
| | COL_SUBSAMPLE_TREE_RANGE | COL_SAMPLE_RATE_BYTREE_RANGE |
| | COL_SUBSAMPLE_TREE_VALUES | COL_SAMPLE_RATE_BYTREE_VALUES |
| | LEARNING_RATE | ETA |
| | LEARNING_RATE_RANGE | ETA_RANGE |
| | LEARNING_RATE_VALUES | ETA_VALUES |
| | SPLIT_THRESHOLD | GAMMA |
| | SPLIT_THRESHOLD_RANGE | GAMMA_RANGE |
| | SPLIT_THRESHOLD_VALUES | GAMMA_VALUES |
| | N_ESTIMATORS | ITER_NUM |
| | N_ESTIMATORS_RANGE | ITER_NUM_RANGE |
| | N_ESTIMATORS_VALUES | ITER_NUM_VALUES |
| | LAMB | LAMBDA |
| | LAMB_RANGE | LAMBDA_RANGE |
| | LAMB_VALUES | LAMBDA_VALUES |
| | MIN_SAMPLES_WEIGHT_LEAF | MIN_CHILD_HESSIAN |
| | MIN_SAM-PLES_WEIGHT_LEAF_RANGE | MIN_CHILD_HESSIAN_RANGE |
| | MIN_SAMPLES_WEIGHT_LEAF_VAL-UES | MIN_CHILD_HESSIAN_VALUES |
| | MIN_SAMPLES_LEAF | NODE_SIZE |
| | MIN_SAMPLES_LEAF_RANGE | NODE_SIZE_RANGE |
| | MIN_SAMPLES_LEAF_VALUES | NODE_SIZE_VALUES |
| | MAX_W_IN_SPLIT | NODE_WEIGHT_CONSTRAINT |
| | MAX_W_IN_SPLIT_RANGE | NODE_WEIGHT_CONSTRAINT_RANGE |

| Algorithm | New Name | Original Name |
|---|---|---|
| | MAX_W_IN_SPLIT_VALUES | NODE_WEIGHT_CONSTRAINT_VALUES |
| | SUBSAMPLE | ROW_SAMPLE_RATE |
| | SUBSAMPLE_RANGE | ROW_SAMPLE_RATE_RANGE |
| | SUBSAMPLE_VALUES | ROW_SAMPLE_RATE_VALUES |
| MLP | ACTIVATION | HIDDEN_LAYER_ACTIVE_FUNC |
| | OUTPUT_ACTIVATION | OUTPUT_LAYER_ACTIVE_FUNC |
| | ACTIVATION_OPTIONS | HIDDEN_LAYER_ACTIVE_FUNC_VALUES |
| | HIDDEN_LAYER_SIZE_OPTIONS | HIDDEN_LAYER_SIZE_VALUES |
| | MAX_ITER | MAX_ITERATION |
| | BATCH_SIZE | MINI_BATCH_SIZE |
| | BATCH_SIZE_RANGE | MINI_BATCH_SIZE_RANGE |
| | BATCH_SIZE_VALUES | MINI_BATCH_SIZE_VALUES |
| | MOMENTUM | MOMENTUM_FACTOR |
| | MOMENTUM_RANGE | MOMENTUM_FACTOR_RANGE |
| | MOMENTUM_VALUES | MOMENTUM_FACTOR_VALUES |
| | OUTPUT_ACTIVATION_OPTIONS | OUTPUT_LAYER_ACTIVE_FUNC_VALUES |
| RDT | ALLOW_MISSING_LABEL | ALLOW_MISSING_DEPENDENT |
| | MIN_SAMPLES_LEAF | NODE_SIZE |
| | N_ESTIMATORS | TREES_NUM |
| | MAX_FEATURES | TRY_NUM |
| SVM | TOL | ERROR_TOL |

## 5.13.10.2 Parameter Tables

This topic lists all parameters for unified regression with pivoted input data.

### _SYS_AFL.PAL_UNIFIED_REGRESSION_PIVOT

#### UNIFIED REGRESSION

Mandatory parameters

| Name | Data Type | Description |
| --- | --- | --- |
| FUNCTION | VARCHAR or NVARCHAR | The algorithm name, including:<br>• GEO;<br>• LOG;<br>• EXP;<br>• GLM;<br>• MLR;<br>• POL;<br>• DT;<br>• HGBT;<br>• MLP;<br>• RDT;<br>• SVM |

Optional parameters

| Name | Data Type | Default Value | Description | Dependency |
| --- | --- | --- | --- | --- |
| PARTITION_METHOD | INTEGER | 0 | Data set partition via different algorithm:<br>• 0: Do not use partition;<br>• 1: Data set partition with specific flag;<br>• 2: Data set partition with random. | If the parameter is set to be 2, the 4th column in data table is invalid. |
| PARTITION_TRAINING_PERCENT | DOUBLE | 0.8 | Specific scale value to data partition with random.<br>Value: 0-1 | Valid only for partition_method=2. |
| PARTITION_RANDOM_SEED | INTEGER | 0 | Indicates the seed used to initialize the random number generator.<br>• 0: Uses the system time;<br>• Not 0: Uses the specific seed. | Valid only for partition_method=2. |
| OUTPUT_PARTITION_RESULT | INTEGER | 0 | Output partition result or not:<br>• 0: Does not output result;<br>• 1: Outputs result. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| IGNORE_ZERO | INTEGER | 0 | Valid values are:<br>• 0: Uses zero values in the actual dataset when calculating MPE or MAPE;<br>• 1: Ignores zero values in the actual dataset when calculating MPE or MAPE. | Only valid when MEASURE_NAME is MPE or MAPE. |
| HANDLE_MISSING_VALUE | INTEGER | 0 | Whether to handle missing values:<br>• 0: No<br>• 1: Yes | |
| IMPUTATION_TYPE | INTEGER | 1 | Chooses the overall default imputation type for all columns:<br>• 0: NONE<br>• 1: MODE_MEAN<br>• 2: MODE_MEDIAN<br>• 3: MODE_ALLZERO<br>• 4: MODE_ALS<br>• 5: DELETE<br>NONE and DELETE: set all columns to imputation type NONE/DELETE.<br>MODE_MEAN, MODE_MEDIAN, MODE_ALLZERO, and MODE_ALS: set categorical columns to the first imputation type (the type before “”); set numerical columns to the second imputation type (the type after “”). ALLZERO means to fill all missing values in numerical columns with 0. | Only valid when HANDLE_MISSING_VALUE is 1. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| `<column name>`_IMPUTATION_TYPE | INTEGER | No default value | Sets imputation type of a specified column. This value overwrites the default imputation type deduced by the IMPUTATION_TYPE parameter:<br>• 0: NONE<br>• 1: DELETE<br>• 100: MODE<br>• 101: SPECIFIED_CATEGORICAL_VALUE<br>• 200: MEAN<br>• 201: MEDIAN<br>• 203:SPECIFIED_NUMERICAL_VALUE<br>• 204: ALS<br>For type 101 (SPECIFIED_CATEGORICAL_VALUE), use the following syntax to specify a categorical value to be filled in a column V0:<br>INSERT INTO #PAL_PARAMETER_TBL VALUES('V0_IMPUTATION_TYPE', 101, NULL, '`<CATEGORICAL_VALUE>`');<br>For type 203 (SPECIFIED_NUMERICAL_VALUE), use the following syntax to specify a numerical value to be filled in a column V1:<br>INSERT INTO #PAL_PARAMETER_TBL VALUES('V1_IMPUTA- | Only valid when HANDLE_MISSING_VALUE is 1. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| | | | TION_TYPE', 203, `<NUMERICAL_VALUE>`, NULL);<br><br>**Note**<br>A column of integer type can be set to type 101 (SPECIFIED_CATEGORICAL_VALUE) or type 203 (SPECIFIED_NUMERICAL_VALUE):<br>• when setting to type 101 (SPECIFIED_CATEGORICAL_VALUE), the string must contain a valid integer value, otherwise an error will be thrown;<br>• when setting to type 203 (SPECIFIED_NUMERICAL_VALUE), the double value will be converted to integer. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| ALS_FACTOR_NUMBER | INTEGER | 3 | Length of factor vectors, that is, the f in ALS model.<br>**Note**<br>ALS_FACTOR_NUMBER should be less than the number of numerical columns to get a meaningful result. | |
| ALS_REGULARIZATION | DOUBLE | 1e-2 | L2 regularization of the factors, that is, the λ in ALS model. | |
| ALS_MAX_ITERATION | INTEGER | 20 | Maximum number of iterations when training ALS model. | |
| ALS_SEED | INTEGER | 0 | Specifies the seed of the random number generator used in the training of ALS model:<br>• 0: Uses the current time as the seed<br>• Others: Uses the specified value as the seed. | |
| ALS_EXIT_THRESHOLD | DOUBLE | 0.0 | The training of ALS model exits if the value of cost function is not decreased more than the value of ALS_EXIT_THRESHOLD since the last check.<br>If ALS_EXIT_THRESHOLD is set to 0, there is no check and the training only exits on reaching the maximum number of iterations. Evaluations of cost function require additional calculations. You can set this parameter to 0 to avoid it. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| ALS_EXIT_INTERVAL | INTEGER | 5 | During the training of ALS model, it will calculate cost function and check the exit criteria every `<ALS_EXIT_INTERVAL>` iterations. Evaluations of cost function require additional calculations. | |
| ALS_LINEAR_SYSTEM_SOLVER | INTEGER | 0 | • 0: cholesky solver<br>• 1: cg solver (recommended when ALS_FACTOR_NUMBER is large) | |
| ALS_CG_MAX_ITERATION | INTEGER | 3 | Maximum number of iteration of cg solver. | |
| ALS_CENTERING | INTEGER | 1 | Whether to center the data by column before training ALS model.<br>• 0: no centering<br>• 1: centering. | |
| ALS_SCALING | INTEGER | 1 | Whether to scale the data by column before training ALS model.<br>• 0: no scaling<br>• 1: scaling | |
| SIGNIFICANCE_LEVEL | DOUBLE | 0.05 | The significance level for the confidence interval and the prediction interval. Valid range is (0, 1). | |
| INTERVAL | INTEGER | 0 | Specifies the type of interval to output.<br>• 0: No interval is calculated and output;<br>• 1: Confidence Interval;<br>• 2: Prediction Interval. | |

## GEO

R

Mandatory parameters

None

Optional parameters

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| DECOMPOSITION | INTEGER | 1 | Matrix factorization type to use. Case-insensitive.<br>• 0: LU Decomposition;<br>• 1: QR Decomposition;<br>• 2: SVD;<br>• 5: Cholesky Decomposition |
| ADJUSTED_R2 | INTEGER | 0 | • 0: Does not output adjusted R square;<br>• 1: Outputs adjusted R square. |
| FEATURES | VARCHAR or NVARCHAR | No default value | A string to specify the features that will be processed. The pattern is "$X_1,...,X_n$", where $X_i$ is the corresponding column name in the data table. If this parameter is not specified, all the features will be processed. |

**LOG**

Mandatory parameters

None

Optional parameters

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| DECOMPOSITION | INTEGER | 1 | Matrix factorization type to use. Case-insensitive.<br>• 0: LU Decomposition;<br>• 1: QR Decomposition;<br>• 2: SVD;<br>• 5: Cholesky Decomposition |
| ADJUSTED_R2 | INTEGER | 0 | • 0: Does not output adjusted R square;<br>• 1: Outputs adjusted R square. |

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| FEATURES | VARCHAR or NVARCHAR | No default value | A string to specify the features that will be processed. The pattern is "$X_1,...,X_n$", where $X_i$ is the corresponding column name in the data table. If this parameter is not specified, all the features will be processed. |

**EXP**

Mandatory parameters

None

Optional parameters

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| DECOMPOSITION | INTEGER | 1 | Matrix factorization type to use. Case-insensitive.<br>• 0: LU Decomposition;<br>• 1: QR Decomposition;<br>• 2: SVD;<br>• 5: Cholesky Decomposition |
| ADJUSTED_R2 | INTEGER | 0 | • 0: Does not output adjusted R square;<br>• 1: Outputs adjusted R square. |
| FEATURES | VARCHAR or NVARCHAR | No default value | A string to specify the features that will be processed. The pattern is "$X_1,...,X_n$", where $X_i$ is the corresponding column name in the data table. If this parameter is not specified, all the features will be processed. |

**GLM**

Mandatory parameters

None

Optional parameters

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| SOLVER | VARCHAR or NVARCHAR | irls | Specifies the optimisation algorithm.<br>• irls: iterative reweighted least square;<br>• cd: coordinate descent, solving ENET problem. | |
| FAMILY | VARCHAR or NVARCHAR | gaussian | Specifies the distribution:<br>• gaussian, normal;<br>• poisson;<br>• binomial;<br>• gamma;<br>• inversegaussian;<br>• negativebinomial. | |
| LINK | VARCHAR or NVARCHAR | • gaussian: identity;<br>• poisson: log;<br>• binomial: logit;<br>• gamma: reciprocal;<br>• inversegaussian: nversesquare;<br>• negativebinomial: log. | Specifies the link function:<br>• identity;<br>• log;<br>• logit;<br>• probit;<br>• comploglog;<br>• reciprocal, inverse;<br>• inversesquare;<br>• sqrt. | The supported link functions for each distribution are:<br>• gaussian, normal: identity, log, reciprocal, inverse;<br>• poisson: identity, log;<br>• binomial: logit, probit, comploglog, log;<br>• gamma: identity, reciprocal, inverse, log;<br>• inversegaussian: inversesquare, identity, reciprocal, inverse, log;<br>• negativebinomial: {identity, log, sqrt}. |
| HANDLE_MISSING_FIT | INTEGER | 1 | Specifies how to handle missing covariates:<br>• 0: throw error;<br>• 1: remove missing rows;<br>• 2: replace missing value with 0. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| QUASILIKELIHOOD | INTEGER | 0 | Indicates whether to use the quasi-likelihood to estimate over-dispersion.<br>• 0: Does not consider over-dispersion;<br>• 1: Considers over-dispersion. | Valid only for Poisson and Negative Binomial distribution. |
| GROUP_RESPONSE | INTEGER | 0 | Indicates whether the response consists of two columns.<br>• 0: Consists of only one column (the target column);<br>• 1: Consists of two columns(two target columns) | Valid only for Binomial distribution, n,k for $\binom{n}{k}$, respectively.<br>• Define 'n' data to target in metadata table;<br>• Must define the K column at last in metadata table. |
| MAX_ITER | INTEGER | • 100 for IRLS;<br>• 1e5 for coordinate descent | The maximum number of iterations for numeric optimization. | |
| TOL | DOUBLE | • 1e-8 for IRLS;<br>• 1e-7 for coordinate descent. | The convergence criterion of coefficients for numeric optimization. | IRLS and coordinate descent evaluate the relative change of objective. |
| SIGNIFICANCE_LEVEL | DOUBLE | 0.05 | The significance level for the confidence interval of estimated coefficients. | Not for coordinate descent. |
| ALPHA | DOUBLE | 1.0 | The elastic net mixing parameter. The value range is between 0 and 1 inclusively. | Valid only for coordinate descent. |
| LAMBDA | DOUBLE | No default value | Coefficient for L1 and L2 regularization. | Valid only for coordinate descent. |
| NUM_LAMBDA | INTEGER | 100 | The number of lambda values. | Valid only for coordinate descent. Takes effect only when LAMBDA is not given explicitly. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| LAMBDA_MIN_RATIO | DOUBLE | 1e-2 or 1e-4 | The smallest value of lambda, as a fraction of the maximum lambda, where $\lambda max$ is the smallest value for which all coefficients are zero. | Valid only for coordinate descent.<br>Takes effect only when LAMBDA is not given explicitly.<br>When the number of observations is smaller than the number of covariates, LAMBDA_MIN_RATIO defaults to 1e-2; Otherwise, it defaults to 1e-4. |
| CATEGORICAL_VARIABLE | VARCHAR, or NVARCHAR | Detected from input data | Indicates which variables are cheated as categorical. The default behavior is:<br>• string: categorical;<br>• integer and double: continuous | Valid only for integer or string column. |
| EXCLUDED_FEATURE | VARCHAR or NVARCHAR | No default value | Excludes the indicated feature column. | Cannot be the ID column name and the column name of dependent variable. |

Model Evaluation and Parameter Selection

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| RESAMPLING_METHOD | VARCHAR or NVARCHAR | No default value | Specifies the resampling method for model evaluation or parameter selection.<br>• cv;<br>• bootstrap.<br>If no value is specified for this parameter, neither model evaluation nor parameter selection is activated. | |
| EVALUATION_METRIC | VARCHAR or NVARCHAR | No default value | Specifies the evaluation metric for model evaluation or parameter selection.<br>• RMSE;<br>• MAE. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| FOLD_NUM | INTEGER | No default value | Specifies the fold number for a cross validation method. | Mandatory and valid only when RESAMPLING_METHOD is set to cv. |
| REPEAT_TIMES | INTEGER | 1 | Specifies the number of repeat times for resampling. | |
| PARAM_SEARCH_STRATEGY | VARCHAR or NVARCHAR | No default value | Specifies the method to activate parameter selection:<br>• grid;<br>• random. | |
| RANDOM_SEARCH_TIMES | INTEGER | No default value | Specifies the number of times to randomly select candidate parameters for selection. | Mandatory and valid when PARAM_SEARCH_STRATEGY is set to random. |
| SEED | INTEGER | 0 | Specifies the seed for random generation. Use system time when 0 is specified. | |
| TIMEOUT | INTEGER | 0 | Specifies maximum running time for model evaluation or parameter selection, in seconds. No timeout when 0 is specified. | |
| PROGRESS_INDICATOR_ID | VARCHAR or NVARCHAR | No default value | Sets an ID of progress indicator for model evaluation or parameter selection. No progress indicator is active if no value is provided. | |
| LINK_VALUES | VARCHAR or NVARCHAR | No default value | The discrete set for candidate of link functions. | |
| LAMBDA_VALUES | VARCHAR or NVARCHAR | No default value | Specifies values of LAMBDA to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| LAMBDA_RANGE | VARCHAR or NVARCHAR | No default value | Specifies the range of LAMBDA to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| ALPHA_VALUES | VARCHAR or NVARCHAR | No default value | Specifies values of ALPHA to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| ALPHA_RANGE | VARCHAR or NVARCHAR | No default value | Specifies the range of parameter ALPHA to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |

**MLR**

Mandatory Parameters

None.

Optional Parameters

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| THREAD_RATIO | DOUBLE | -1.0 | The ratio of available threads for multi-thread task:<br>• 0: single thread<br>• 0–1: uses the specified percentage of available threads. PAL uses all available threads if the number is 1.0.<br>• -1: uses all of available threads. | Only valid when SLOVER is 1,4,5, or 6. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| SOLVER | INTEGER | 1 | Specifies algorithms for solving the least square problem:<br>• 1: QR decomposition (numerically stable, but fails when A is rank-deficient);<br>• 2: SVD (numerically stable and can handle rank deficiency but computationally expensive);<br>• 4: Cyclical coordinate descent method to solve elastic net regularized multiple linear regression;<br>• 5: Cholesky decomposition (fast but numerically unstable);<br>• 6: Alternating direction method of multipliers (ADMM) to solve elastic net regularized multiple linear regression. | The value 4 and 6 are supported only when VAR_SELECT is 0. |
| CATEGORICAL_VARIABLE | VARCHAR or NVARCHAR | No default value | Indicates by the column name whether or not a column data is actually corresponding to a category variable even the data type of this column is INTEGER.<br>By default, 'VARCHAR' or 'NVARCHAR' is category variable, and 'INTEGER' or 'DOUBLE' is continuous variable. | The data type of the column specified by this parameter must be INTEGER. |
| MODEL_FORMAT | INTEGER | 1 | Specifies the tree model format for store:<br>• 1: JSON<br>• 2: PMML | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| VAR_SELECT | INTEGER | 0 | • 0: All variables are included;<br>• 1: Forward selection;<br>• 2: Backward selection;<br>• 3: Stepwise selection. | The value 1 or 2 or 3 are supported only when SLOVER is not 4 or 6. |
| MANDATORY_FEATURE | VARCHAR or NVARCHAR | No default value | Specifies the column name that needs to be included in the final training model when executing the variable selection.<br>This parameter can be specified multiple times, each time with one column name as feature.<br>**Note**<br>This parameter is a hint. There are exceptional cases that specified mandatory feature is excluded in the final model, for instance, some mandatory features can be represented as a linear combination of other features, among which some are also mandatory features. | Only valid when VAR_SELECT is not 0. |
| INTERCEPT | INTEGER | 0 | Specifies whether the intercept term should be ignored in the model.<br>• 0: Does not ignore;<br>• 1: Ignore. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| ALPHA_TO_ENTER | DOUBLE | 0.05 or 0.15 | P-value for forward selection and stepwise selection. | • When VAR_SELECT is 1, default value is 0.05;<br>• when VAR_SELECT is 3, default value is 0.15. |
| ALPHA_TO_REMOVE | DOUBLE | 0.1 or 0.15 | P-value for backward selection and stepwise selection. | • When VAR_SELECT is 2, default value is 0.1;<br>• when VAR_SELECT is 3, default value is 0.15. |
| LAMBDA | DOUBLE | No default value | Penalized weight.<br>The value should be equal to or greater than 0. | Only valid when SLOVER is 4 or 6. |
| ALPHA | DOUBLE | 1.0 | The elastic net mixing parameter.<br>The value range is between 0 and 1 inclusively.<br>• 0: Ridge penalty;<br>• 1: LASSO penalty. | Only valid when SLOVER is 4 or 6. |
| HANDLE_MISSING | INTEGER | 1 | Whether to handle missing value:<br>• 0: No;<br>• 1: Yes. | |
| MAX_ITER | INTEGER | 1e5 | Maximum number of passes over training data. If convergence is not reached after the specified number of iterations, an error will be generated. | Only valid when SLOVER is 4 or 6. |
| TOL | DOUBLE | 1.0e-7 | Convergence threshold for coordinate descent. | Only valid when SLOVER is 4. |
| PHO | DOUBLE | 1.8 | Step size for ADMM. Generally, the value should be greater than 1. | Only valid when SLOVER is 6. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| STAT_INF | INTEGER | 0 | Specifies whether to output t-value and Pr(>\|t\|) of coefficients in the Result table or not.<br>• 0: No;<br>• 1: Yes. | |
| ADJUSTED_R2 | INTEGER | 0 | Specifies whether to output adjusted R square or not.<br>• 0: No;<br>• 1: Yes. | |
| DW_TEST | INTEGER | 0 | Specifies whether to do Durbin-Watson test under the null hypothesis that the errors do not follow a first order autoregressive process.<br>• 0: No;<br>• 1: Yes. | Not available if elastic net regularization is enabled or INTERCEPT = 1. |
| RESET_TEST | INTEGER | 1 | Specifies the order of Ramsey RESET test. Choosing 1 means this test will not be conducted.<br>If you specify an INTEGER larger than 1, then the MLR function will run Ramsey RESET test with power of variables ranging from 2 to the value you specify. | Not available if elastic net regularization is enabled or INTERCEPT = 1. |
| BP_TEST | INTEGER | 0 | Specifies whether or not to do Breusch-Pagan test under the null hypothesis that homoscedasticity is satisfied.<br>• 0: No;<br>• 1: Yes. | Not available if elastic net regularization is enabled or INTERCEPT = 1. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| KS_TEST | INTEGER | 0 | Specifies whether or not to do Kolmogorov-Smirnov normality test under the null hypothesis that if errors of MLR follow a normal distribution.<br>• 0: No;<br>• 1: Yes. | Not available if elastic net regularization is enabled or INTERCEPT = 1. |
| SELECTED_FEATURES | VARCHAR or NVARCHAR | No default value | A string to specify the features that will be processed. The pattern is "$X_1$,...,$X_n$", where $X_i$ is the corresponding column name in the data table. If this parameter is not specified, all the features will be processed. | |
| OUTPUT_COEFCOV | INTEGER | 0 | Specifies whether to output coefficient covariance information.<br>• 0: Doesn't output related information;<br>• 1: Outputs related information. | Valid only when JSON_EXPORT=1. |

Model Evaluation and Parameter Selection

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| RESAMPLING_METHOD | VARCHAR or NVARCHAR | No default value | Specifies the resampling method for model evaluation or parameter selection.<br>• cv;<br>• cv_sha<br>• cv_hyperband<br>• bootstrap.<br>• bootstrap_sha<br>• bootstrap_hyperband<br>If no value is specified for this parameter, neither model evaluation nor parameter selection is activated. | |
| RESOURCE | NVARCHAR | No default value | Resource type used in successive-halving and hyperband algorithm for parameter selection.<br>• MAX_ITER | Mandatory and valid only when RESAMPLING_METHOD parameter is cv_sha, cv_hyperband, bootstrap_sha, or bootstrap_hyperband. |
| MAX_RESOURCE | INTEGER | No default value | Maximum resource budget for single hyper-parameter candidate. Valid value must be larger than 0. | Mandatory and valid only when RESOURCE parameter is set. |
| MIN_RESOURCE_RATE | DOUBLE | 0.0 | Minimum required resource budget comparing to maximum resource for single hyper-parameter candidate.<br>Valid value should be within [0, 1). | Valid only when RESOURCE parameter is set. |
| REDUCTION_RATE | DOUBLE | 3.0 | This parameter specifies the reduction rate of available size of hyper-parameter candidates. For each round, the available parameter candidate size will be divided by value of this parameter. Thus valid value for this parameter must be larger than 1.0. | Valid only when RESOURCE parameter is set. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| AGGRESSIVE_ELIMINATION | INTEGER | 0 | Boolean value indicates whether aggressive elimination will be performed for successive-halving algorithm. When set to 1, it will eliminate more parameter candidates than expected, which is defined by REDUCTION_RATE parameter; This can enhance the run-time performance but may result in sub-optimal hyper-parameter candidate.<br>• 0: Does not perform aggressive elimination;<br>• 1: Performs aggressive elimiination | Valid only when RESAMPLING_METHOD parameter is cv_sha or bootstrap_sha. |
| PS_VERBOSE | INTEGER | 1 | Boolean value indicates whether optimal hyper-parameter and all evaluation statistics of related hyper-parameter candidates will be output in Statistics output table.<br>• 0: Does not output evaluation statistics<br>• 1: Outputs evaluation statistics | Valid only when RESAMPLING_METHOD is set. |
| EVALUATION_METRIC | VARCHAR or NVARCHAR | No default value | Specifies the evaluation metric for model evaluation or parameter selection.<br>• RMSE;<br>• MAE. | |
| FOLD_NUM | INTEGER | No default value | Specifies the fold number for a cross validation method. | Mandatory and valid only when RESAMPLING_METHOD is set to cv, cv_sha, or cv_hyperband. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| REPEAT_TIMES | INTEGER | 1 | Specifies the number of repeat times for re-sampling. | |
| PARAM_SEARCH_STRATEGY | VARCHAR or NVARCHAR | No default value | Specifies the method to activate parameter selection.<br>• grid;<br>• random. | This parameter is only valid when RESAMPLING_METHOD is not cv_hyperband nor bootstrap_hyperband; Otherwise, this parameter is always considered to have value `random`. |
| RANDOM_SEARCH_TIMES | INTEGER | No default value | Specifies the number of times to randomly select candidate parameters for selection. | Mandatory and valid when PARAM_SEARCH_STRATEGY is set to random. |
| SEED | INTEGER | 0 | Specifies the seed for random generation. Use system time when 0 is specified. | |
| TIMEOUT | INTEGER | 0 | Specifies maximum running time for model evaluation or parameter selection, in seconds. No timeout when 0 is specified. | |
| PROGRESS_INDICATOR_ID | VARCHAR or NVARCHAR | No default value | Sets an ID of progress indicator for model evaluation or parameter selection. No progress indicator is active if no value is provided. | |
| LAMBDA_VALUES | VARCHAR or NVARCHAR | No default value | Specifies values of LAMBDA to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| LAMBDA_RANGE | VARCHAR or NVARCHAR | No default value | Specifies the range of LAMBDA to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| ALPHA_VALUES | VARCHAR or NVARCHAR | No default value | Specifies values of ALPHA to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| ALPHA_RANGE | VARCHAR or NVARCHAR | No default value | Specifies the range of parameter ALPHA to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| PRECOM-PUTE_LMS_SKETCH | INTEGER | 1 | • 0: Do not perform LMS sketch<br>• 1: Performs LMS sketch | LMS sketch will only perform when RESAM-PLING_METHOD is set and Data Row Number is larger than Data Fea-ture Number. |
| STABLE_SKETCH_ALG | INTEGER | 1 | When computing LMS sketch, there are two algorithms to choose. One algorithm is more numerical stable than the other one, but is slower.<br>• 0: Do not use sta-ble algorithm<br>• 1: Uses stable al-gorithm | Only valid when LMS sketch is performed and SPARSE_SKETCH_AL G is 0. |
| SPARSE_SKETCH_AL G | INTEGER | 0 | There is a specific LMS sketch algorithm to cope with sparse data.<br>• Do not use sparse LMS sketch<br>• Uses sparse LMS sketch algorithm<br>**Note**<br>Use this option with caution. If the provided data are not sparse, turning on this option will cause LMS Sketch to be extremely slow! | Only valid when LMS sketch is performed. |

## POL

Mandatory Parameter

| Name | Data Type | Description |
|---|---|---|
| DEGREE | INTEGER | Degree of the polynomial model. |

Optional Parameters

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| DECOMPOSITION | INTEGER | 1 | Matrix factorization type to use. Case-insensitive.<br>• 0: LU Decomposition;<br>• 1: QR Decomposition;<br>• 2: SVD;<br>• 5: Cholesky Decomposition |
| ADJUSTED_R2 | INTEGER | 0 | • 0: Does not output adjusted R square;<br>• 1: Outputs adjusted R square |
| FEATURES | VARCHAR or NVARCHAR | No default value | A string to specify the features that will be processed. The pattern is "$X_1,...,X_n$", where $X_i$ is the corresponding column name in the data table. If this parameter is not specified, all the features will be processed. |

Model Evaluation and Parameter Selection

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| RESAMPLING_METHOD | VARCHAR or NVARCHAR | No default value | Specifies the resampling method for model evaluation or parameter selection.<br>• cv;<br>• bootstrap.<br>If no value is specified for this parameter, neither model evaluation nor parameter selection is activated. | |
| EVALUATION_METRIC | VARCHAR or NVARCHAR | No default value | Specifies the evaluation metric for model evaluation or parameter selection.<br>• RMSE;<br>• MAE. | |
| FOLD_NUM | INTEGER | No default value | Specifies the fold number for a cross validation method. | Mandatory and valid only when RESAMPLING_METHOD is set to cv. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| REPEAT_TIMES | INTEGER | 1 | Specifies the number of repeat times for re-sampling. | |
| PARAM_SEARCH_STRATEGY | VARCHAR or NVARCHAR | No default value | Specifies the method to activate parameter selection.<br>• grid;<br>• random. | |
| RANDOM_SEARCH_TIMES | INTEGER | No default value | Specifies the number of times to randomly select candidate parameters for selection. | Mandatory and valid when PARAM_SEARCH_STRATEGY is set to random. |
| SEED | INTEGER | 0 | Specifies the seed for random generation. Use system time when 0 is specified. | |
| TIMEOUT | INTEGER | 0 | Specifies maximum running time for model evaluation or parameter selection, in seconds. No timeout when 0 is specified. | |
| PROGRESS_INDICATOR_ID | VARCHAR or NVARCHAR | No default value | Sets an ID of progress indicator for model evaluation or parameter selection. No progress indicator is active if no value is provided. | |
| DEGREE_VALUES | VARCHAR or NVARCHAR | No default value | Specifies values of DEGREE to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| DEGREE_RANGE | VARCHAR or NVARCHAR | No default value | Specifies the range of DEGREE to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |

**DT**

Mandatory Parameters

None

Optional Parameters

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| THREAD_RATIO | DOUBLE | -1.0 | The ratio of available threads for multi-thread task:<br>• 0: single thread<br>• 0–1: uses the specified percentage of available threads. PAL uses all available threads if the number is 1.0.<br>• -1: uses all of available threads. | |
| ALLOW_MISSING_LABEL | INTEGER | 1 | Specifies if missing target value is allowed.<br>• 0: Not allowed. If missing target presents, an error occurs.<br>• 1: Allowed. The datum with missing target is removed. | |
| PERCENTAGE | DOUBLE | 1.0 | Specifies the percentage of the input data that will be used to build the tree model.<br>For example, if you set this parameter to 0.7, then 70% of the training data will be used to build the tree model and 30% will be used to prune the tree model. | |
| MIN_RECORDS_PARENT | INTEGER | 2 | Specifies the stop condition: if the number of records in one node is less than the specified value, the algorithm stops splitting. | |
| MIN_RECORDS_LEAF | INTEGER | 1 | Promises the minimum number of records in a leaf. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| MAX_DEPTH | INTEGER | Unlimited | Specifies the stop condition: if the depth of the tree model is greater than the specified value, the algorithm stops splitting. | |
| CATEGORICAL_VARIABLE | VARCHAR or NVARCHAR | Detected from input data | Indicates which variables are treated as categorical. The default behavior is:<br>• STRING: categorical;<br>• INTEGER and DOUBLE: continuous. | Valid only for INTEGER variables; omitted otherwise. |
| SPLIT_THRESHOLD | DOUBLE | 1e-5 | Specifies the stop condition for a node. The reduction of Gini index or relative MSE of the best split is less than this value.<br>The smaller the SPLIT_THRESHOLD value is, the larger a CART tree grows. | |
| USE_SURROGATE | INTEGER | 1 | Indicates whether to use surrogate split when NULL values are encountered.<br>• 0: Does not use surrogate split.<br>• 1: Uses surrogate split. | |
| MODEL_FORMAT | INTEGER | 1 | Specifies the tree model format for store:<br>• 1: JSON;<br>• 2: PMML. | |

Parameters for Model Evaluation and Parameter Search

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| RESAMPLING_METHOD | VARCHAR or NVARCHAR | No default value | Specifies the resampling method for model evaluation or parameter selection.<br>• cv;<br>• bootstrap;<br>If no value is specified for this parameter, neither model evaluation nor parameter selection is activated. | |
| EVALUATION_METRIC | VARCHAR or NVARCHAR | No default value | Specifies the evaluation metric for model evaluation or parameter selection.<br>• RMSE;<br>• MAE. | |
| FOLD_NUM | INTEGER | No default value | Specifies the fold number for a cross validation method. | Valid only for cross validation; mandatory for cross validation. |
| REPEAT_TIMES | INTEGER | 1 | The number of repeated times for model evaluation or parameter search. | |
| PARAM_SEARCH_STRATEGY | VARCHAR or NVARCHAR | No default value | The search strategy for parameters:<br>• random: random search;<br>• grid: grid search | If not specified, only model evaluation is carried out. |
| RANDOM_SEARCH_TIMES | INTEGER | No default value | Specifies the number of search times for random search. | Mandatory for random search. |
| TIMEOUT | INTEGER | 0 | The time allocated (seconds) for program running. 0 indicates unlimited. | |
| PROGRESS_INDICATOR_ID | VARCHAR or NVARCHAR | No default value | Sets an ID of progress indicator for model evaluation or parameter selection. | No progress indicator is active if no value is provided. |
| MIN_RECORDS_LEAF_VALUES | VARCHAR or NVARCHAR | No default value | The discrete set for the MIN_RECORDS_LEAF candidate. | |
| MIN_RECORDS_LEAF_RANGE | VARCHAR or NVARCHAR | No default value | The range for the MIN_RECORDS_LEAF candidate. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| MIN_RECORDS_PARENT_VALUES | VARCHAR or NVARCHAR | No default value | The discrete set for the MIN_RECORDS_PARENT candidate. | |
| MIN_RECORDS_PARENT_RANGE | VARCHAR or NVARCHAR | No default value | The range for the MIN_RECORDS_PARENT candidate. | |
| MAX_DEPTH_VALUES | VARCHAR or NVARCHAR | No default value | The discrete set for the MAX_DEPTH candidate. | |
| MAX_DEPTH_RANGE | VARCHAR, or NVARCHAR | No default value | The range for the MAX_DEPTH candidate. | |
| SPLIT_THRESHOLD_VALUES | VARCHAR or NVARCHAR | No default value | The discrete set for the SPLIT_THRESHOLD candidate. | |
| SPLIT_THRESHOLD_RANGE | VARCHAR or NVARCHAR | No default value | The range for the SPLIT_THRESHOLD candidate. | |

### HGBT

Mandatory Parameters

None.

Optional Parameters

| Name | Data Type | Default Value | Description | Denpendency |
|---|---|---|---|---|
| THREAD_RATIO | DOUBLE | -1.0 | The ratio of available threads for multi-thread task:<br>• 0: single thread<br>• 0–1: uses the specified percentage of available threads. PAL uses all available threads if the number is 1.0.<br>• -1: uses all of available threads. | |
| SEED | INTEGER | 0 | The seed for random number generating.<br>• 0: current time;<br>• Others: the seed. | |

| Name | Data Type | Default Value | Description | Denpendency |
|---|---|---|---|---|
| SPLIT_METHOD | VARCHAR or NVARCHAR | histogram | The method to find split point for numeric feature.<br>• exact: the exact method, trying all possible points;<br>• sketch: the sketch method, accounting for the distribution of the sum of hessian;<br>• sampling: samples the split point randomly;<br>• histogram:builds histogram upon data and uses it as split point. | The exact method comparably has the highest test accuracy, but costs more time. The other three methods have relative higher computational efficiency but might lead to lower test accuracy, and are considered to be adopted as the training data set is huge.<br>Valid only for numeric features. |
| SKETCH_EPS | DOUBLE | 0.1 | The epsilon of the sketch method. It indicates that the sum of hessian between two split points is not larger than this value. That is, the number of bins is approximately 1/eps. | The less is this value, the more split points are tried. |
| MAX_BIN_NUM | INTEGER | 256 | The maximum bin number for histogram method. Decreasing this number gains better performance in terms of running time at a cost of accuracy. | Only valid when SPLIT_METHOD is set to histogram. |
| N_ESTIMATORS | INTEGER | 10 | Total iteration number, which is equivalent to the number of trees in the final model. | |
| MAX_DEPTH | INTEGER | 6 | The maximum depth of each tree. | |
| MIN_SAMPLES_LEAF | INTEGER | 1 | The minimum number of data in a leaf. | |
| LEARNING_RATE | DOUBLE | 0.3 | Learning rate of each iteration.<br>Range: (0, 1] | |

| Name | Data Type | Default Value | Description | Denpendency |
|---|---|---|---|---|
| SPLIT_THRESHOLD | DOUBLE | 0.0 | The minimum loss change value to make a split in tree growth (gamma in the equation). | |
| MIN_SAMPLES_WEIGHT_LEAF | DOUBLE | 1.0 | The minimum summation of sample weights(hessian) in the leaf node. | |
| MAX_W_IN_SPLIT | DOUBLE | 0.0 | The maximum weight constraint assigned to each tree node. 0 for no constraint. | |
| SUBSAMPLE | DOUBLE | 1.0 | The sample rate of row (data points).<br>Range: (0, 1] | |
| COL_SUBSAMPLE_SPLIT | DOUBLE | 1.0 | The sample rate of feature set in each split.<br>Range: (0, 1] | |
| COL_SUBSAMPLE_TREE | DOUBLE | 1.0 | The sample rate of feature set in each tree growth.<br>Range: (0, 1] | |
| LAMB | DOUBLE | 1.0 | L2 regularization.<br>Range: [0, 1] | |
| ALPHA | DOUBLE | 0.0 | L1 regularization.<br>Range: [0, 1] | |
| BASE_SCORE | DOUBLE | 0.0 | Initial prediction score of all instances.<br>Global bias for sufficient number of iterations. | |
| START_FROM_AVERAGE | INTEGER | 0 | Indicates whether to adopt the prior distribution as the initial point. To be specific, use average value if it is a regression problem.<br>• 0: no;<br>• 1: yes. | BASE_SCORE is ignored if this parameter is set to 1. |

| Name | Data Type | Default Value | Description | Denpendency |
|---|---|---|---|---|
| OBJ_FUNC | INTEGER | • 0 for regression problem<br>• 5 for binary classification<br>• 7 for multiple classification | For regression:<br>• 0: Squared error<br>• 1: Squared log error<br>• 2: Pseudo Huber error<br>• 3: Gamma<br>• 4: Tweedie<br>For binary classification:<br>• 5: Logistic<br>• 6: Hinge<br>For multiple classification:<br>• 7: Softmax | |
| TWEEDIE_POWER | DOUBLE | 1.5 | For Tweedie object function; must be in range [1.0, 2.0]. | Only valid when OBJ_FUNC equals 4. |
| REPLACE_MISSING | INTEGER | 1 | Replaces the missing value by another value in the feature.<br>If it is a continuous feature, the value is the mean value;<br>If it is a categorical feature, the value is the most frequent one.<br>• 0: no<br>• 1: yes | |
| DEFAULT_MISSING_DIRECTION | INTEGER | 1 | Defines the default direction where the missing value goes to while splitting.<br>• 0: left<br>• 1: right | |

| Name | Data Type | Default Value | Description | Denpendency |
|---|---|---|---|---|
| CATEGORICAL_VARIABLE | VARCHAR or NVARCHAR | Detected from input data | Indicates which variables are treated as categorical. The default behavior is:<br>• STRING: categorical;<br>• INTEGER and DOUBLE: continuous. | Valid only for INTEGER variables; omitted otherwise. |
| CALCULATE_IMPORTANCE | INTEGER | 1 | Determines whether to calculate variable importance:<br>• 0: no;<br>• 1: yes. | |

Early Stop

See Early Stop of Hybrid Gradient Boosting Tree [page 231] for details.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| VALIDATION_SET_RATE | DOUBLE | 0.0 | Specifies the rate of validation set that is sampled from the data set. If 0.0 is set, then no early stop will be applied. | |
| STRATIFIED_VALIDATION_SET | INTEGER | 0 | Indicates weather or not to apply the stratified method while sampling validation set. | Valid only when VALIDATION_SET_RATE is greater than 0. |
| TOLERANT_ITER_NUM | INTEGER | 10 | Indicates how many consecutive deteriorated iterations are observed before applying early stop. | Valid only when VALIDATION_SET_RATE is greater than 0. |

Feature Grouping

See Feature Grouping for Hybrid Gradient Boosting Tree [page 232] for details.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| FEATURE_GROUPING | INTEGER | 1 | Applies feature grouping by grouping sparse features that only contain one significant value in each row.<br>• 0: no<br>• 1: yes | |
| TOLERANT_RATE | DOUBLE | 0.0001 | While applying feature grouping, features are still merged when there are rows containing more than one significant value. This parameter specifies the rate of such rows that are allowed. | Only available when FEATURE_GROUPING is set to 1. |
| FG_MIN_ZERO_RATE | DOUBLE | 0.5 | Specifies the minimum zero rate that is used to indicate sparse columns, only which may take part in feature grouping. | Only available when FEATURE_GROUPING is set to 1. |

Model Evaluation and Parameter Selection

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| RESAMPLING_METHOD | VARCHAR or NVARCHAR | No default value | Specifies the resampling method for model evaluation or parameter selection.<br>• cv<br>• cv_sha<br>• cv_hyperband<br>• bootstrap<br>• bootstrap_sha<br>• bootstrap_hyperband<br>If no value is specified for this parameter, neither model evaluation nor parameter selection is activated. | Options like *_sha and *_hyperband can only be used in parameter selection, can not be used in model evaluation. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| EVALUATION_METRIC | VARCHAR or NVARCHAR | No default value | Specify evaluation metric for model evaluation or parameter selection:<br>• RMSE;<br>• MAE. | It is mandatory if RESAMPLING_METHOD is set. |
| REF_METRIC | VARCHAR or NVARCHAR | No default value | The metric(s) for reference. Could be of multi-lines, for example, more than one REF_METRIC. | The metric that is identical to EVALUATION_METRIC will be omitted. |
| FOLD_NUM | INTEGER | No default value | Specifies the fold number for a cross validation method. | Mandatory and valid only when RESAMPLING_METHOD is set to cv. |
| REPEAT_TIMES | INTEGER | 1 | Specifies the number of repeat times for resampling. | |
| PARAM_SEARCH_STRATEGY | VARCHAR or NVARCHAR | No default value | Specify value to this parameter to active parameter selection.<br>• grid;<br>• random. | If this parameter is not set, then only model evaluation is activated. |
| RANDOM_SEARCH_TIMES | INTEGER | No default value | Specifies the number of times to randomly select candidate parameters for selection. | Mandatory and valid when PARAM_SEARCH_STRATEGY is set to random. |
| REDUCTION_RATE | DOUBLE | 3.0 | Specifies reduction rate in SHA or Hyperband method. | Only valid when RESAMPLING_METHOD is specified as *_sha or *_hyperband. |
| RESOURCE | NVARCHAR | No default value | Specifies resource definition for SHA or Hyperband method.<br>• Not specified, size of data set is used as resource by default.<br>• 'N_ESTIMATORS' | Only valid when RESAMPLING_METHOD is specified as *_sha or *_hyperband. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| MIN_RESOURCE_RATE | DOUBLE | Algorithm specified | Specifies the minimum resource rate that should be used in SHA or Hyperband iteration. | Only valid when RESAMPLING_METHOD is specified as *_sha or *_hyperband.<br>Default value 0.0 for default RESOURCE and default value 1/MAX_RESOURCE for RESOURCE = N_ESTIMATORS. |
| MAX_RESOURCE | INTEGER | No default value | Specifies the maximum number of N_ESTIMATORS that should be used in SHA or Hyperband method. | Only valid when RESAMPLING_METHOD is specified as *_sha or *_hyperband. It is valid and mandatory if RESOURCE is set. |
| AGGRESSIVE_ELIMINATION | INTEGER | 0 | Specifies whether to apply aggressive elimination while using SHA method. Aggressive elimination happens when the data size and parameters size to be searched does not match and there are still bunch of parameters to be searched while data size reaches its upper limits. If aggressive elimination is applied, lower bound of limit of data size will be used multiple times first to reduce number of parameters.<br>• 0: doe not apply aggressive elimination<br>• 1: applies aggressive elimination | Only valid when RESAMPLING_METHOD is specified as *_sha. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| TIMEOUT | INTEGER | 0 | Specifies maximum running time for model evaluation or parameter selection, in seconds.<br>No timeout when 0 is specified. | |
| PROGRESS_INDICATOR_ID | VARCHAR or NVARCHAR | No default value | Sets an ID of progress indicator for model evaluation or parameter selection. No progress indicator is active if no value is provided. | |
| N_ESTIMATORS_VALUES | VARCHAR or NVARCHAR | No default value | Specifies values of parameter N_ESTIMATORS to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| N_ESTIMATORS_RANGE | VARCHAR or NVARCHAR | No default value | Specifies range of parameter N_ESTIMATORS to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| MAX_DEPTH_VALUES | VARCHAR or NVARCHAR | No default value | Specifies values of parameter MAX_DEPTH to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| MAX_DEPTH_RANGE | | No default value | Specifies range of parameter MAX_DEPTH to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| MIN_SAMPLES_LEAF_VALUES | VARCHAR or NVARCHAR | No default value | Specifies values of parameter MIN_SAMPLES_LEAF to be selected.VARCHAR or NVARCHAR | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| MIN_SAMPLES_LEAF_RANGE | VARCHAR or NVARCHAR | No default value | Specifies range of parameter MIN_SAMPLES_LEAF to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| LEARNING_RATE_VALUES | VARCHAR or NVARCHAR | No default value | Specifies values of parameter LEARNING_RATE to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| LEARNING_RATE_RANGE | VARCHAR or NVARCHAR | No default value | Specifies range of parameter LEARNING_RATE to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| SPLIT_THRESHOLD_VALUES | VARCHAR or NVARCHAR | No default value | VARCHAR or NVARCHSpecifies values of parameter SPLIT_THRESHOLD to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| SPLIT_THRESH-OLD_RANGE | VARCHAR or NVARCHAR | No default value | Specifies range of parameter SPLIT_THRESHOLD to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| MIN_SAM-PLES_WEIGHT_LEAF_VALUES | VARCHAR or NVARCHAR | No default value | Specifies values of parameter MIN_SAM-PLES_WEIGHT_LEAF to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| MIN_SAM-PLES_WEIGHT_LEAF_RANGE | VARCHAR or NVARCHAR | No default value | Specifies range of parameter MIN_SAM-PLES_WEIGHT_LEAF to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| MAX_W_IN_SPLIT_VALUES | VARCHAR or NVARCHAR | No default value | Specifies values of parameter MAX_W_IN_SPLIT to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| MAX_W_IN_SPLIT_RANGE | VARCHAR or NVARCHAR | No default value | Specifies range of parameter MAX_W_IN_SPLIT to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| COL_SUBSAM-PLE_SPLIT_VALUES | VARCHAR or NVARCHAR | No default value | Specifies values of parameter COL_SUB-SAMPLE_SPLIT to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| COL_SUBSAM-PLE_SPLIT_RANGE | VARCHAR or NVARCHAR | No default value | Specifies range of parameter COL_SUB-SAMPLE_SPLIT to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| COL_SUBSAM-PLE_TREE_VALUES | VARCHAR or NVARCHAR | No default value | Specifies values of parameter COL_SUB-SAMPLE_TREE to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| COL_SUBSAM-PLE_TREE_RANGE | VARCHAR or NVARCHAR | No default value | Specifies range of parameter COL_SUB-SAMPLE_TREE to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| LAMB_VALUES | VARCHAR or NVARCHAR | No default value | Specifies values of parameter LAMB to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| LAMB_RANGE | VARCHAR or NVARCHAR | No default value | Specifies range of parameter LAMB to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| ALPHA_VALUES | VARCHAR or NVARCHAR | No default value | Specifies values of parameter ALPHA to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| ALPHA_RANGE | VARCHAR or NVARCHAR | No default value | Specifies range of parameter ALPHA to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |

> **Note**
>
> - To active parameter selection or model evaluation, parameter RESAMPLING_METHOD must be set. Then, to active parameter selection, parameter PARAM_SEARCH_STRATEGY must be set, or only model evaluation is carried out. For parameters like *_VALUES, its format is {val1, val2, ..., valn}. For parameters like *_RANGE, its format is [RANGE_START, STEP, RANGE_END]. STEP can be omitted if using random search, in which case its format is [RANGE_START ,, RANGE_END]. Commas are still needed.
> - To make this function back compatible, old cross validation fashion is still supported. In this case, RESAMPLING_METHOD is not set and FOLD_NUM is set to a value greater than 1. Parameter EVALUATION_METRIC and REF_METRIC has default value RMSE for regression. And parameters for searching purpose are of name RANGE_* and format like [RANGE_START, RANGE_END, NUM], of which the first and second ones are of integer or double types, whereas the last one is of integer type (not less than 2), indicating that the number of candidate values by equal space constrained by range of [RANGE_START, RANGE_END].

**MLP**

Mandatory Parameters

| Name | Data Type | Description |
|---|---|---|
| ACTIVATION | INTEGER | Active function code for the hidden layer.<br>• TANH = 1;<br>• LINEAR = 2;<br>• SIGMOID_ASYMMETRIC = 3;<br>• SIGMOID_SYMMETRIC = 4;<br>• GAUSSIAN_ASYMMETRIC = 5;<br>• GAUSSIAN_SYMMETRIC = 6;<br>• ELLIOT_ASYMMETRIC = 7;<br>• ELLIOT_SYMMETRIC = 8;<br>• SIN_ASYMMETRIC = 9;<br>• SIN_SYMMETRIC = 10;<br>• COS_ASYMMETRIC = 11;<br>• COS_SYMMETRIC = 12;<br>• ReLU = 13. |
| OUTPUT_ACTIVATION | INTEGER | Active function code for the output layer. The code is the same as ACTIVATION. |

| Name | Data Type | Description |
| --- | --- | --- |
| HIDDEN_LAYER_SIZE | VARCHAR or NVARCHAR | Specifies the size of each hidden layer in the format of “2, 3, 4”. The value 0 will be ignored, for example, “2, 0, 3” is equal to “2, 3”. |

Optional Parameters

| Name | Data Type | Default Value | Description | Dependency |
| --- | --- | --- | --- | --- |
| THREAD_RATIO | DOUBLE | -1.0 | The ratio of available threads.<br>• 0: single thread<br>• 0–1: uses the specified percentage of available threads. PAL uses all available threads if the number is 1.0.<br>• -1: uses all of available threads. | |
| MAX_ITER | INTEGER | 100 | Maximum number of iterations. | |
| TRAINING_STYLE | INTEGER | 1 | Specifies the training style:<br>• 0: Batch;<br>• 1: Stochastic. | |
| LEARNING_RATE | DOUBLE | No default value | Specifies the learning rate. | Mandatory and valid only when TRAINING_STYLE = 1. |
| MOMENTUM | DOUBLE | No default value | Specifies the momentum factor. | Mandatory and valid only when TRAINING_STYLE = 1. |
| BATCH_SIZE | INTEGER | 1 | Specifies the size of mini batch. | Valid only when TRAINING_STYLE = 1. |
| NORMALIZATION | INTEGER | 0 | Specifies the normalization type:<br>• 0: None;<br>• 1: Z-transform;<br>• 2: Scalar. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| WEIGHT_INIT | INTEGER | 0 | Specifies the weight initial value:<br>• 0: All zeros;<br>• 1: Normal distribution;<br>• 2: Uniform distribution in range (0, 1);<br>• 3: Variance scale normal;<br>• 4: Variance scale uniform. | |
| CATEGORICAL_VARIABLE | VARCHAR or NVARCHAR | No default value | Indicates whether or not a column data is actually corresponding to a category variable even the data type of this column is INTEGER.<br>By default, 'VARCHAR' or 'NVARCHAR' is category variable, and 'INTEGER' or 'DOUBLE' is continuous variable. | |

Model Evaluation and Parameter Selection

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| RESAMPLING_METHOD | VARCHAR or NVARCHAR | No default value | Specifies the resampling method for model evaluation or parameter selection.<br>• cv;<br>• bootstrap;<br>If no value is specified for this parameter, neither model evaluation nor parameter selection is activated. | |
| EVALUATION_METRIC | VARCHAR or NVARCHAR | No default value | Specifies the evaluation metric for model evaluation or parameter selection.<br>• RMSE. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| FOLD_NUM | INTEGER | No default value | Specifies the fold number for a cross validation method. | Mandatory and valid only when RESAMPLING_METHOD is set to cv. |
| REPEAT_TIMES | INTEGER | 1 | Specifies the number of repeat times for resampling. | |
| PARAM_SEARCH_STRATEGY | VARCHAR or NVARCHAR | No default value | Specifies the method to activate parameter selection.<br>• grid;<br>• random. | |
| RANDOM_SEARCH_TIMES | INTEGER | No default value | Specifies the number of times to randomly select candidate parameters for selection. | Mandatory and valid when PARAM_SEARCH_STRATEGY is set to random. |
| SEED | INTEGER | 0 | Specifies the seed for random generation. Use system time when 0 is specified. | |
| TIMEOUT | INTEGER | 0 | Specifies maximum running time for model evaluation or parameter selection, in seconds. No timeout when 0 is specified. | |
| PROGRESS_INDICATOR_ID | VARCHAR or NVARCHAR | No default value | Sets an ID of progress indicator for model evaluation or parameter selection. No progress indicator is active if no value is provided. | |
| ACTIVATION_OPTIONS | VARCHAR or NVARCHAR | No default value | Specifies values of ACTIVATION to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| OUTPUT_ACTIVATION_OPTIONS | VARCHAR or NVARCHAR | No default value | Specifies values of OUTPUT_ACTIVATION to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| HIDDEN_LAYER_SIZE_OPTIONS | VARCHAR or NVARCHAR | No default value | Specifies values of HIDDEN_LAYER_SIZE to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| LEARNING_RATE_VALUES | VARCHAR or NVARCHAR | No default value | Specifies values of LEARNING_RATE to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| LEARNING_RATE_RANGE | VARCHAR or NVARCHAR | No default value | Specifies range of LEARNING_RATE to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| MOMENTUM_VALUES | VARCHAR or NVARCHAR | No default value | Specifies values of MOMENTUM to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| MOMENTUM_RANGE | VARCHAR or NVARCHAR | No default value | Specifies range of MOMENTUM to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| BATCH_SIZE_VALUES | VARCHAR or NVARCHAR | No default value | Specifies values of BATCH_SIZE to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| BATCH_SIZE_RANGE | VARCHAR or NVARCHAR | No default value | Specifies range of BATCH_SIZE to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |

**RDT**

Mandatory Parameters

None

Optional Parameters

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| THREAD_RATIO | DOUBLE | -1.0 | The ratio of available threads.<br>• 0: single thread<br>• 0–1: uses the specified percentage of available threads. PAL uses all available threads if the number is 1.0.<br>• -1: uses all of available threads. | |
| N_ESTIMATORS | INTEGER | 100 | Specifies the number of trees to grow. | |
| MAX_FEATURES | INTEGER | p/3, where p is the number of valid independent variables. | Specifies the number of randomly selected variables for splitting. | Should not larger than the valid number of features. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| ALLOW_MISSING_LABEL | INTEGER | 1 | Specifies if missing target value is allowed.<br>• 0: Not allowed. An error occurs if missing dependent presents.<br>• 1: Allowed. The datum with missing dependent is removed. | |
| CATEGORICAL_VARIABLE | VARCHAR or NVARCHAR | Detected from input data | Indicates which variables are treated as categorical. The default behavior is:<br>• STRING: categorical;<br>• INTEGER and DOUBLE: continuous. | VALID only for INTEGER variables; omitted otherwise. |
| SEED | INTEGER | 0 | Specifies the seed for random number generator.<br>• 0: Uses the current time (in second) as seed.<br>• Others: Uses the specified value as seed. | |
| MIN_SAMPLES_LEAF | INTEGER | 5 | Specifies the minimum number of records in a leaf. | |
| SPLIT_THRESHOLD | DOUBLE | 1e-5 | Specifies the stop condition: if the improvement value of the best split is less than this value, the tree stops growing. | |
| MAX_DEPTH | INTEGER | 56 | The maximum depth of a tree. | If MAX_DEPTH is set to -1, there are no restrictions on the depth of the tree. Otherwise, MAX_DEPTH cannot be larger than 56. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| SAMPLE_FRACTION | DOUBLE | 1 | The fraction of data used for training.<br>Assume there are n pieces of data, SAMPLE_FRACTION is r, then n*r data is selected for training. | In the range of [0,1].<br>If n*r is less than the number of classes, the sample size is restored to the number of classes. |
| COMPRESSION | INTEGER | 0 | Specifies if the model is stored in compressed format.<br>• 0: No;<br>• 1: Yes. | |
| MAX_BITS | INTEGER | 12 | Maximal number of bits to quantize continuous features. Equivalent to use $2^{MAX_BITS}$ bins. | Must be less than 31. |
| QUANTIZE_RATE | DOUBLE | 0.005 | Quantizes a categorical feature if the largest class frequency of the feature is less than QUANTIZE_RATE. | Valid only when COMPRESSION is equal to 1. |
| OUTPUT_LEAF_VALUES | INTEGER | 0 | Controls whether to save target values in each leaf node.<br>• 0: Doesn't output target values in each leaf node;<br>• 1: Outputs target values in each leaf node. | Valid only when JSON_EXPORT = 1 or COMPRESSION = 1. |
| FITTINGS_QUANTIZATION | INTEGER | 0 | Indicates whether to quantize fitting values.<br>• 0: No;<br>• 1: Yes. | Valid only for regression when COMPRESSION is equal to 1. |
| MODEL_FORMAT | INTEGER | 1 | Specifies the tree model format for store:<br>• 1: JSON<br>• 2: PMML | If both MODEL_FORMAT and COMPRESSION are specified, COMPRESSION takes precedence. |

### SVM

Mandatory Parameter

None

Optional Parameters

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| KERNEL_TYPE | INTEGER | 2 | Kernel type:<br>• 0: LINEAR KERNEL;<br>• 1: POLY KERNEL;<br>• 2: RBF KERNEL;<br>• 3: SIGMOID KERNEL. | |
| THREAD_RATIO | DOUBLE | -1.0 | The ratio of available threads.<br>• 0: single thread<br>• 0–1: uses the specified percentage of available threads. PAL uses all available threads if the number is 1.0.<br>• -1: uses all of available threads. | |
| POLY_DEGREE | INTEGER | 3 | Coefficient for the PLOY KERNEL type.<br>Value range: ≥ 1 | Only valid when KERNEL_TYPE is 1. |
| RBF_GAMMA | DOUBLE | 1.0/number of features in the dataset | Coefficient for the RBF KERNEL type.<br>Value range: > 0 | Only valid when KERNEL_TYPE is 2. |
| COEF_LIN | DOUBLE | 0.0 | Coefficient for the POLY/SIGMOID KERNEL type. | Only valid when KERNEL_TYPE is 1 or 3. |
| COEF_CONST | DOUBLE | 0.0 | Coefficient for the POLY/SIGMOID KERNEL type. | Only valid when KERNEL_TYPE is 1 or 3. |
| SVM_C | DOUBLE | 100.0 | Trade-off between training error and margin.<br>Value range: > 0 | |
| REGRESSION_EPS | DOUBLE | 0.1 | Epsilon width of tube for regression.<br>Value range: > 0 | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| SCALE_INFO | INTEGER | 1 | • 0: No scale.<br>• 1: Standardization.The algorithm transforms the data to have zero mean and unit variance. The general formula is given as:<br>$x_{new}=\frac{x-\mu}{\sigma}$,<br>where x is the origin feature vector, μ is the mean of that feature vector, and σ is its standard deviation.<br>• 2: Rescaling. The algorithm rescales the range of the features to scale the range in [-1,1]. The general formula is given as:<br>$x_{new}=-1+\frac{x-\min(x)}{\max(x)-\min(x)}*2$,<br>where x is the origin feature vector. | |
| SCALE_LABEL | INTEGER | 1 | Controls whether to standardize the label.<br>• 0: No;<br>• 1: Yes. | Only valid when SCALE_INFO is 1. |
| SHRINK | INTEGER | 1 | Decides whether to use shrink strategy or not:<br>• 0: Does not use shrink strategy.<br>• 1: Uses shrink strategy. | Using shrink strategy may accelerate the training process. |
| HANDLE_MISSING | INTEGER | 1 | Whether to handle missing values:<br>• 0: No;<br>• 1: Yes. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| CATEGORICAL_VARIABLE | VARCHAR or NVARCHAR | No default value | Indicates whether or not a column data is actually corresponding to a category variable even the data type of this column is INTEGER.<br>By default, 'VARCHAR' or 'NVARCHAR' is category variable, and 'INTEGER' or 'DOUBLE' is continuous variable. | |
| CATEGORY_WEIGHT | DOUBLE | 0.707 | Represents the weight of category attributes (γ). The value must be greater than 0. | |
| TOL | DOUBLE | 0.001 | Specifies the error tolerance in the training process. The value must be greater than 0. | |
| COMPRESSION | INTEGER | 1 | Specifies if the model is stored in compressed format.<br>• 0 : no;<br>• 1: yes. | |
| MAX_BITS | INTEGER | 12 | Maximal number of bits to quantize continuous features. Equivalent to use $2^{MAX_BITS}$ bins. | Must be less than 31. |
| MAX_QUANTIZATION_ITER | DOUBLE | 1000 | The maximum iteration steps for quantization. | Valid only when COMPRESSION is equal to 1. |

Model Evaluation and Parameter Selection

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| RESAMPLING_METHOD | VARCHAR or NVARCHAR | No default value | Specifies the resampling method for model evaluation or parameter selection.<br>• cv<br>• cv_sha<br>• cv_hyperband<br>• bootstrap<br>• bootstrap_sha<br>• bootstrap_hyperband<br>If no value is specified for this parameter, neither model evaluation nor parameter selection is activated. | |
| EVALUATION_METRIC | VARCHAR or NVARCHAR | No default value | Specifies the evaluation metric for model evaluation or parameter selection.<br>• RMSE. | |
| FOLD_NUM | INTEGER | No default value | Specifies the fold number for a cross validation method. | Mandatory and valid only when RESAMPLING_METHOD is set to cv. |
| REPEAT_TIMES | INTEGER | 1 | Specifies the number of repeat times for resampling. | |
| PARAM_SEARCH_STRATEGY | VARCHAR or NVARCHAR | No default value | Specifies the method to activate parameter selection.<br>• grid;<br>• random. | When *_sha is specified in RESAMPLING_METHOD, PARAM_SEARCH_STRATEGY is mandatory. When *_hyperband is specified in RESAMPLING_METHOD, PARAM_SEARCH_STRATEGY is set to random whatever other value is provided. |
| RANDOM_SEARCH_TIMES | INTEGER | No default value | Specifies the number of times to randomly select candidate parameters for selection. | Mandatory and valid when PARAM_SEARCH_STRATEGY is set to random. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| REDUCTION_RATE | DOUBLE | 3.0 | Specifies reduction rate in SHA or Hyperband method. | Only valid when RESAMPLING_METHOD is specified as *_sha or *_hyperband. |
| AGGRESSIVE_ELIMINATION | INTEGER | 0 | Specifies whether to apply aggressive elimination while using SHA method. Aggressive elimination happens when the data size and parameters size to be searched does not match and there are still bunch of parameters to be searched while data size reaches its upper limits. If aggressive elimination is applied, lower bound of limit of data size will be used multiple times first to reduce number of parameters.<br>• 0: doe not apply aggressive elimination<br>• 1: applies aggressive elimination | Only valid when RESAMPLING_METHOD is specified as *_sha. |
| SEED | INTEGER | 0 | Specifies the seed for random generation. Use system time when 0 is specified. | |
| TIMEOUT | INTEGER | 0 | Specifies maximum running time for model evaluation or parameter selection, in seconds. No timeout when 0 is specified. | |
| PROGRESS_INDICATOR_ID | VARCHAR or NVARCHAR | No default value | Sets an ID of progress indicator for model evaluation or parameter selection. No progress indicator is active if no value is provided. | |
| SVM_C_VALUES | VARCHAR or NVARCHAR | No default value | Specifies values of SVM_C to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| SVM_C_RANGE | VARCHAR or NVARCHAR | No default value | Specifies range of SVM_C to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| RBF_GAMMA_VALUES | VARCHAR or NVARCHAR | No default value | Specifies values of RBF_GAMMA to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| RBF_GAMMA_RANGE | VARCHAR or NVARCHAR | No default value | Specifies range of RBF_GAMMA to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| POLY_DEGREE_VALUES | VARCHAR or NVARCHAR | No default value | Specifies values of POLY_DEGREE to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| POLY_DEGREE_RANGE | VARCHAR or NVARCHAR | No default value | Specifies range of POLY_DEGREE to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| COEF_LIN_VALUES | VARCHAR or NVARCHAR | No default value | Specifies values of COEF_LIN to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| COEF_LIN_RANGE | VARCHAR or NVARCHAR | No default value | Specifies range of COEF_LIN to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| COEF_CONST_VALUES | VARCHAR or NVARCHAR | No default value | Specifies values of COEF_CONST to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |
| COEF_CONST_RANGE | VARCHAR or NVARCHAR | No default value | Specifies range of COEF_CONST to be selected. | Only valid when PARAM_SEARCH_STRATEGY is specified. |

## _SYS_AFL.PAL_UNIFIED_REGRESSION_PREDICT_PIVOT

### PIVOTED UNIFIED REGRESSION PREDICT

Mandatory parameters

None.

Optional parameters

| Name | Data Type | Default Value | Description | Constraint |
|---|---|---|---|---|
| THREAD_RATIO | DOUBLE | -1.0 | The ratio of available threads for multi-thread task:<br>• 0: single thread<br>• 0–1: uses the specified percentage of available threads. PAL uses all available threads if the number is 1.0.<br>• -1: uses all of available threads. | Excludes: GLM |
| HANDLE_MISSING_VALUE | INTEGER | 0 | Whether to handle missing values:<br>• 0: No<br>• 1: Yes | |
| IMPUTATION_TYPE | INTEGER | 1 | Chooses the overall default imputation type for all columns:<br>• 0: NONE<br>• 1: MODE_MEAN<br>• 2: MODE_MEDIAN<br>• 3: MODE_ALLZERO<br>• 4: MODE_ALS<br>• 5: DELETE<br>NONE and DELETE: set all columns to imputation type NONE/DELETE.<br>MODE_MEAN, MODE_MEDIAN, MODE_ALLZERO, and MODE_ALS: set categorical columns to the first imputation type (the type before “”); set numerical columns to the second imputation type (the type after “”). ALLZERO means to fill all missing values in numerical columns with 0. | Only valid when HANDLE_MISSING_VALUE is 1. |

| Name | Data Type | Default Value | Description | Constraint |
|---|---|---|---|---|
| <column name>_IMPUTATION_TYPE | INTEGER | No default value | Sets imputation type of a specified column. This value overwrites the default imputation type deduced by the IMPUTATION_TYPE parameter:<br>• 0: NONE<br>• 1: DELETE<br>• 100: MODE<br>• 101: SPECIFIED_CATEGORICAL_VALUE<br>• 200: MEAN<br>• 201: MEDIAN<br>• 203:SPECIFIED_NUMERICAL_VALUE<br>• 204: ALS<br>For type 101 (SPECIFIED_CATEGORICAL_VALUE), use the following syntax to specify a categorical value to be filled in a column V0:<br>INSERT INTO #PAL_PARAMETER_TBL VALUES('V0_IMPUTATION_TYPE', 101, NULL, '<CATEGORICAL_VALUE>');<br>For type 203 (SPECIFIED_NUMERICAL_VALUE), use the following syntax to specify a numerical value to be filled in a column V1:<br>INSERT INTO #PAL_PARAMETER_TBL VALUES('V1_IMPUTA- | Only valid when HANDLE_MISSING_VALUE is 1. |

| Name | Data Type | Default Value | Description | Constraint |
|---|---|---|---|---|
| | | | TION_TYPE', 203, `<NUMERICAL_VALUE>`, NULL);<br><br>**Note**<br>A column of integer type can be set to type 101 (SPECIFIED_CATEGORICAL_VALUE) or type 203 (SPECIFIED_NUMERICAL_VALUE):<br>• when setting to type 101 (SPECIFIED_CATEGORICAL_VALUE), the string must contain a valid integer value, otherwise an error will be thrown;<br>• when setting to type 203 (SPECIFIED_NUMERICAL_VALUE), the double value will be converted to integer. | |

| Name | Data Type | Default Value | Description | Constraint |
|---|---|---|---|---|
| ALS_FACTOR_NUMBER | INTEGER | 3 | Length of factor vectors, that is, the f in ALS model.<br>**Note**<br>ALS_FACTOR_NUMBER should be less than the number of numerical columns to get a meaningful result. | |
| ALS_REGULARIZATION | DOUBLE | 1e-2 | L2 regularization of the factors, that is, the λ in ALS model. | |
| ALS_MAX_ITERATION | INTEGER | 20 | Maximum number of iterations when training ALS model. | |
| ALS_SEED | INTEGER | 0 | Specifies the seed of the random number generator used in the training of ALS model:<br>• 0: Uses the current time as the seed<br>• Others: Uses the specified value as the seed. | |
| ALS_EXIT_THRESHOLD | DOUBLE | 0.0 | The training of ALS model exits if the value of cost function is not decreased more than the value of ALS_EXIT_THRESHOLD since the last check.<br>If ALS_EXIT_THRESHOLD is set to 0, there is no check and the training only exits on reaching the maximum number of iterations. Evaluations of cost function require additional calculations. You can set this parameter to 0 to avoid it. | |

| Name | Data Type | Default Value | Description | Constraint |
|---|---|---|---|---|
| ALS_EXIT_INTERVAL | INTEGER | 5 | During the training of ALS model, it will calculate cost function and check the exit criteria every `<ALS_EXIT_INTERVAL>` iterations. Evaluations of cost function require additional calculations. | |
| ALS_LINEAR_SYSTEM_SOLVER | INTEGER | 0 | • 0: cholesky solver<br>• 1: cg solver (recommended when ALS_FACTOR_NUMBER is large) | |
| ALS_CG_MAX_ITERATION | INTEGER | 3 | Maximum number of iteration of cg solver. | |
| ALS_CENTERING | INTEGER | 1 | Whether to center the data by column before training ALS model.<br>• 0: no centering<br>• 1: centering. | |
| ALS_SCALING | INTEGER | 1 | Whether to scale the data by column before training ALS model.<br>• 0: no scaling<br>• 1: scaling | |
| SIGNIFICANCE_LEVEL | DOUBLE | 0.05 | The significance level for the confidence interval and prediction interval. Valid range is (0, 1). | |
| INTERVAL | INTEGER | 0 | Specifiecs the type of interval to output.<br>• 0: No interval will be caluclated and output;<br>• 1: Confidence Interval;<br>• 2: Prediction Interval. | |

### GLM

Mandatory parameters

None

Optional parameters

| Name | Data Type | Default Value | Description | Constraint |
|---|---|---|---|---|
| TYPE | VARCHAR or NVARCHAR | response | The prediction type:<br>• response: Predicts response μ;<br>• link: Predicts linear predictor η. | |
| SIGNIFICANCE_LEVEL | DOUBLE | 0.05 | The significance level for the confidence interval and prediction interval. | Valid only for IRLS. |
| HANDLE_MISSING | INTEGER | 2 | Specify how to handle missing covariates.<br>• 1: Remove missing rows;<br>• 2: Replace missing with 0. | |

**RDT**

Mandatory parameters

None

Optional parameters

| Name | Data Type | Default Value | Description | Constraint |
|---|---|---|---|---|
| BLOCK_SIZE | INTEGER | 0 | Specifies the number of data loaded per time during scoring.<br>• 0: load all data once;<br>• Others: the specified number. | This parameter is for reducing memory consumption, especially as the predict data is huge, or it consists of a large number of missing independent variables. However, you might lose some efficiency. |

## _SYS_AFL.PAL_UNIFIED_REGRESSION_SCORE_PIVOT

**PIVOTED UNIFIED REGRESSION SCORE**

Mandatory parameters

None.

Optional parameters

| Name | Data Type | Default Value | Description | Constraint |
|---|---|---|---|---|
| THREAD_RATIO | DOUBLE | -1.0 | The ratio of available threads for multi-thread task:<br>• 0: single thread<br>• 0–1: uses the specified percentage of available threads. PAL uses all available threads if the number is 1.0.<br>• -1: uses all of available threads. | Excludes: GLM |
| IGNORE_ZERO | INTEGER | 0 | Valid values are:<br>• 0: Uses zero values in the actual dataset when calculating MPE or MAPE;<br>• 1: Ignores zero values in the actual dataset when calculating MPE or MAPE. | Only valid when MEASURE_NAME is MPE or MAPE. |
| HANDLE_MISSING_VALUE | INTEGER | 0 | Whether to handle missing values:<br>• 0: No<br>• 1: Yes | |

| Name | Data Type | Default Value | Description | Constraint |
|---|---|---|---|---|
| IMPUTATION_TYPE | INTEGER | 1 | Chooses the overall default imputation type for all columns:<br>• 0: NONE<br>• 1: MODE_MEAN<br>• 2: MODE_MEDIAN<br>• 3: MODE_ALL-ZERO<br>• 4: MODE_ALS<br>• 5: DELETE<br>NONE and DELETE: set all columns to imputation type NONE/DELETE.<br>MODE_MEAN, MODE_MEDIAN, MODE_ALLZERO, and MODE_ALS: set categorical columns to the first imputation type (the type before “”); set numerical columns to the second imputation type (the type after “”). ALLZERO means to fill all missing values in numerical columns with 0. | Only valid when HANDLE_MISSING_VALUE is 1. |

| Name | Data Type | Default Value | Description | Constraint |
|---|---|---|---|---|
| `<column name>`_IMPUTATION_TYPE | INTEGER | No default value | Sets imputation type of a specified column. This value overwrites the default imputation type deduced by the IMPUTATION_TYPE parameter:<br>• 0: NONE<br>• 1: DELETE<br>• 100: MODE<br>• 101: SPECIFIED_CATEGORICAL_VALUE<br>• 200: MEAN<br>• 201: MEDIAN<br>• 203:SPECIFIED_NUMERICAL_VALUE<br>• 204: ALS<br>For type 101 (SPECIFIED_CATEGORICAL_VALUE), use the following syntax to specify a categorical value to be filled in a column V0:<br>INSERT INTO #PAL_PARAMETER_TBL VALUES('V0_IMPUTATION_TYPE', 101, NULL, '`<CATEGORICAL_VALUE>`');<br>For type 203 (SPECIFIED_NUMERICAL_VALUE), use the following syntax to specify a numerical value to be filled in a column V1:<br>INSERT INTO #PAL_PARAMETER_TBL VALUES('V1_IMPUTA- | Only valid when HANDLE_MISSING_VALUE is 1. |

| Name | Data Type | Default Value | Description | Constraint |
|---|---|---|---|---|
| | | | TION_TYPE', 203, `<NUMERICAL_VALUE>`, NULL);<br><br>**Note**<br>A column of integer type can be set to type 101 (SPECIFIED_CATEGORICAL_VALUE) or type 203 (SPECIFIED_NUMERICAL_VALUE):<br>• when setting to type 101 (SPECIFIED_CATEGORICAL_VALUE), the string must contain a valid integer value, otherwise an error will be thrown;<br>• when setting to type 203 (SPECIFIED_NUMERICAL_VALUE), the double value will be converted to integer. | |

| Name | Data Type | Default Value | Description | Constraint |
|---|---|---|---|---|
| ALS_FACTOR_NUMBER | INTEGER | 3 | Length of factor vectors, that is, the f in ALS model.<br>**Note**<br>ALS_FACTOR_NUMBER should be less than the number of numerical columns to get a meaningful result. | |
| ALS_REGULARIZATION | DOUBLE | 1e-2 | L2 regularization of the factors, that is, the λ in ALS model. | |
| ALS_MAX_ITERATION | INTEGER | 20 | Maximum number of iterations when training ALS model. | |
| ALS_SEED | INTEGER | 0 | Specifies the seed of the random number generator used in the training of ALS model:<br>• 0: Uses the current time as the seed<br>• Others: Uses the specified value as the seed. | |
| ALS_EXIT_THRESHOLD | DOUBLE | 0.0 | The training of ALS model exits if the value of cost function is not decreased more than the value of ALS_EXIT_THRESHOLD since the last check.<br>If ALS_EXIT_THRESHOLD is set to 0, there is no check and the training only exits on reaching the maximum number of iterations. Evaluations of cost function require additional calculations. You can set this parameter to 0 to avoid it. | |

| Name | Data Type | Default Value | Description | Constraint |
|---|---|---|---|---|
| ALS_EXIT_INTERVAL | INTEGER | 5 | During the training of ALS model, it will calculate cost function and check the exit criteria every `<ALS_EXIT_INTERVAL>` iterations. Evaluations of cost function require additional calculations. | |
| ALS_LINEAR_SYSTEM_SOLVER | INTEGER | 0 | • 0: cholesky solver<br>• 1: cg solver (recommended when ALS_FACTOR_NUMBER is large) | |
| ALS_CG_MAX_ITERATION | INTEGER | 3 | Maximum number of iteration of cg solver. | |
| ALS_CENTERING | INTEGER | 1 | Whether to center the data by column before training ALS model.<br>• 0: no centering<br>• 1: centering. | |
| ALS_SCALING | INTEGER | 1 | Whether to scale the data by column before training ALS model.<br>• 0: no scaling<br>• 1: scaling | |

### GLM

Mandatory parameters

None

Optional parameters

| Name | Data Type | Default Value | Description | Constraint |
|---|---|---|---|---|
| TYPE | VARCHAR or NVARCHAR | response | The prediction type:<br>• response: Predicts response $\mu$;<br>• link: Predicts linear predictor $\eta$. | |
| SIGNIFICANCE_LEVEL | DOUBLE | 0.05 | The significance level for the confidence interval and prediction interval. | Valid only for IRLS. |

| Name | Data Type | Default Value | Description | Constraint |
|---|---|---|---|---|
| HANDLE_MISSING | INTEGER | 2 | Specify how to handle missing covariates.<br>• 1: Remove missing rows.<br>• 2: Replace missing with 0. | |

**RDT**

Mandatory parameters

None

Optional parameters

| Name | Data Type | Default Value | Description | Constraint |
|---|---|---|---|---|
| BLOCK_SIZE | INTEGER | 0 | Specifies the number of data loaded per time during scoring.<br>• 0: load all data once;<br>• Others: the specified number. | This parameter is for reducing memory consumption, especially as the predict data is huge, or it consists of a large number of missing independent variables. However, you might lose some efficiency. |

## 5.13.10.3 Data Partition

A data partition process is provided in the train procedure implementation.

The input data can be divided into the train set and the validation set. The train set is used for generating the training model and the validation set is used to evaluate the quality of the model.

You can execute data partition by setting the following parameters.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| PARTITION_METHOD | INTEGER | 0 | Data set Parition via different algorithm:<br>• 0: Do not use partition;<br>• 1: Data set partition with specific flag;<br>• 2: Data set partition with random. | If the parameter is set to be 2, the 4th column in data table is invalid. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| PARTITION_TRAINING_PERCENT | DOUBLE | 0.8 | Specific scale value to data partition with random.<br>Value: 0-1 | Valid only for partition_method=2. |
| PARTITION_RANDOM_SEED | INTEGER | 0 | Indicates the seed used to initialize the random number generator.<br>• 0: Uses the system time;<br>• Not 0: Uses the specific seed. | Valid only for partition_method=2. |
| OUTPUT_PARTITION_RESULT | INTEGER | 0 | Output partition result or not:<br>• 0: Do not output result;<br>• 1: Output result. | |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

**Example 1: Data set partition with specific flag(PARTITION_METHOD=1)**

```
SET SCHEMA DM_PAL;
DROP TABLE META_DATA_TAB;
CREATE COLUMN TABLE META_DATA_TAB(
    "VARIABLE_NAME" NVARCHAR(50),
    "VARIABLE_TYPE" NVARCHAR(50)
);
INSERT INTO META_DATA_TAB VALUES ('OUTLOOK',  'CATEGORICAL');
INSERT INTO META_DATA_TAB VALUES ('CLASS',    'TARGET');
INSERT INTO META_DATA_TAB VALUES ('TEMP',     'CONTINUOUS');
INSERT INTO META_DATA_TAB VALUES ('HUMIDITY', 'CONTINUOUS');
INSERT INTO META_DATA_TAB VALUES ('WINDY',    'CATEGORICAL');
DROP TABLE PAL_DATA_TBL;
CREATE COLUMN TABLE PAL_DATA_TBL(
    "ID" INT,
    "VARIABLE_NAME" NVARCHAR(50),
    "VALUE" NVARCHAR(50),
    "PURPOSE" INT
);
INSERT INTO PAL_DATA_TBL VALUES (0,     'OUTLOOK',      'Sunny',    1);
INSERT INTO PAL_DATA_TBL VALUES (0,     'TEMP',         '75',       1);
INSERT INTO PAL_DATA_TBL VALUES (0,     'HUMIDITY',     '70',       1);
INSERT INTO PAL_DATA_TBL VALUES (0,     'WINDY',        'Yes',      1);
INSERT INTO PAL_DATA_TBL VALUES (0,     'CLASS',        '1',        1);
INSERT INTO PAL_DATA_TBL VALUES (1,     'OUTLOOK',      'Sunny',    1);
INSERT INTO PAL_DATA_TBL VALUES (1,     'TEMP',         '80',       1);
INSERT INTO PAL_DATA_TBL VALUES (1,     'HUMIDITY',     '90',       1);
INSERT INTO PAL_DATA_TBL VALUES (1,     'WINDY',        'Yes',      1);
```

```
INSERT INTO PAL_DATA_TBL VALUES (1,     'CLASS',        '0',        1);
INSERT INTO PAL_DATA_TBL VALUES (2,     'OUTLOOK',      'Sunny',      1);
INSERT INTO PAL_DATA_TBL VALUES (2,     'TEMP',         '85',        1);
INSERT INTO PAL_DATA_TBL VALUES (2,     'HUMIDITY',     '85',        1);
INSERT INTO PAL_DATA_TBL VALUES (2,     'WINDY',        'No',       1);
INSERT INTO PAL_DATA_TBL VALUES (2,     'CLASS',        '0',        1);
INSERT INTO PAL_DATA_TBL VALUES (3,     'OUTLOOK',      'Sunny',    2);
INSERT INTO PAL_DATA_TBL VALUES (3,     'TEMP',         '72',          2);
INSERT INTO PAL_DATA_TBL VALUES (3,     'HUMIDITY',     '95',          2);
INSERT INTO PAL_DATA_TBL VALUES (3,     'WINDY',        'No',          2);
INSERT INTO PAL_DATA_TBL VALUES (3,     'CLASS',        '0',           2);
INSERT INTO PAL_DATA_TBL VALUES (4,     'OUTLOOK',      'Sunny',      1);
INSERT INTO PAL_DATA_TBL VALUES (4,     'TEMP',         '69',       1);
INSERT INTO PAL_DATA_TBL VALUES (4,     'HUMIDITY',     '70',       1);
INSERT INTO PAL_DATA_TBL VALUES (4,     'WINDY',        'No',       1);
INSERT INTO PAL_DATA_TBL VALUES (4,     'CLASS',        '1',        1);
INSERT INTO PAL_DATA_TBL VALUES (5,     'OUTLOOK',      'Overcast', 1);
INSERT INTO PAL_DATA_TBL VALUES (5,     'TEMP',         '72',       1);
INSERT INTO PAL_DATA_TBL VALUES (5,     'HUMIDITY',     '90',       1);
INSERT INTO PAL_DATA_TBL VALUES (5,     'WINDY',        'Yes',      1);
INSERT INTO PAL_DATA_TBL VALUES (5,     'CLASS',        '1',        1);
INSERT INTO PAL_DATA_TBL VALUES (6,     'OUTLOOK',      'Overcast',    1);
INSERT INTO PAL_DATA_TBL VALUES (6,     'TEMP',         '83',       1);
INSERT INTO PAL_DATA_TBL VALUES (6,     'HUMIDITY',     '78',       1);
INSERT INTO PAL_DATA_TBL VALUES (6,     'WINDY',        'No',       1);
INSERT INTO PAL_DATA_TBL VALUES (6,     'CLASS',        '0',        1);
INSERT INTO PAL_DATA_TBL VALUES (7,     'OUTLOOK',      'Overcast',    1);
INSERT INTO PAL_DATA_TBL VALUES (7,     'TEMP',         '64',          1);
INSERT INTO PAL_DATA_TBL VALUES (7,     'HUMIDITY',     '65',          1);
INSERT INTO PAL_DATA_TBL VALUES (7,     'WINDY',        'Yes',         1);
INSERT INTO PAL_DATA_TBL VALUES (7,     'CLASS',        '1',           1);
INSERT INTO PAL_DATA_TBL VALUES (8,        'OUTLOOK',       'Overcast',1);
INSERT INTO PAL_DATA_TBL VALUES (8,     'TEMP',          '81',      1);
INSERT INTO PAL_DATA_TBL VALUES (8,     'HUMIDITY',      '75',      1);
INSERT INTO PAL_DATA_TBL VALUES (8,     'WINDY',         'No',      1);
INSERT INTO PAL_DATA_TBL VALUES (8,     'CLASS',         '1',       1);
INSERT INTO PAL_DATA_TBL VALUES (9,     'OUTLOOK',      'Rain',     2);
INSERT INTO PAL_DATA_TBL VALUES (9,     'TEMP',         '71',       2);
INSERT INTO PAL_DATA_TBL VALUES (9,     'HUMIDITY',     '80',       2);
INSERT INTO PAL_DATA_TBL VALUES (9,     'WINDY',        'Yes',      2);
INSERT INTO PAL_DATA_TBL VALUES (9,     'CLASS',        '0',        2);
INSERT INTO PAL_DATA_TBL VALUES (10,    'OUTLOOK',      'Rain',     1);
INSERT INTO PAL_DATA_TBL VALUES (10,    'TEMP',         '65',       1);
INSERT INTO PAL_DATA_TBL VALUES (10,    'HUMIDITY',     '70',       1);
INSERT INTO PAL_DATA_TBL VALUES (10,    'WINDY',        'Yes',      1);
INSERT INTO PAL_DATA_TBL VALUES (10,    'CLASS',        '0'    ,       1);
INSERT INTO PAL_DATA_TBL VALUES (11,    'OUTLOOK',      'Rain',     1);
INSERT INTO PAL_DATA_TBL VALUES (11,    'TEMP',         '75',       1);
INSERT INTO PAL_DATA_TBL VALUES (11,    'HUMIDITY',     '80',       1);
INSERT INTO PAL_DATA_TBL VALUES (11,    'WINDY',        'No',       1);
INSERT INTO PAL_DATA_TBL VALUES (11,    'CLASS',        '1',        1);
INSERT INTO PAL_DATA_TBL VALUES (12,    'OUTLOOK',      'Rain',     2);
INSERT INTO PAL_DATA_TBL VALUES (12,    'TEMP',         '68',       2);
INSERT INTO PAL_DATA_TBL VALUES (12,    'HUMIDITY',     '80',       2);
INSERT INTO PAL_DATA_TBL VALUES (12,    'WINDY',        'No',       2);
INSERT INTO PAL_DATA_TBL VALUES (12,    'CLASS',        '1',        2);
INSERT INTO PAL_DATA_TBL VALUES (13,    'OUTLOOK',      'Rain',     2);
INSERT INTO PAL_DATA_TBL VALUES (13,    'TEMP',         '70',       2);
INSERT INTO PAL_DATA_TBL VALUES (13,    'HUMIDITY',     '96',       2);
INSERT INTO PAL_DATA_TBL VALUES (13,    'WINDY',        'No',       2);
INSERT INTO PAL_DATA_TBL VALUES (13,    'CLASS',        '0',        2);
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
```

```
INSERT INTO PAL_PARAMETER_TBL VALUES ('FUNCTION',null,null,'DT');
INSERT INTO PAL_PARAMETER_TBL VALUES ('MODEL_FORMAT', 1, NULL, NULL); -- JSON
INSERT INTO PAL_PARAMETER_TBL VALUES ('SPLIT_THRESHOLD', NULL, 1e-5, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('MIN_RECORDS_PARENT', 2, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('MIN_RECORDS_LEAF', 1, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('PARTITION_METHOD',1,null,null);
INSERT INTO PAL_PARAMETER_TBL VALUES ('OUTPUT_PARTITION_RESULT',1,null,null);
INSERT INTO PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', null,0.5,null);
CALL _SYS_AFL.PAL_UNIFIED_REGRESSION_PIVOT(META_DATA_TAB, PAL_DATA_TBL,
PAL_PARAMETER_TBL, ?, ?, ?, ?, ?, ?);
```

**Excepted Result**

PARTITION RESULT

| | ID | TYPE |
|---|---|---|
| 1 | 0 | 1 |
| 2 | 1 | 1 |
| 3 | 2 | 1 |
| 4 | 4 | 1 |
| 5 | 5 | 1 |
| 6 | 6 | 1 |
| 7 | 7 | 1 |
| 8 | 8 | 1 |
| 9 | 10 | 1 |
| 10 | 11 | 1 |
| 11 | 3 | 2 |
| 12 | 9 | 2 |
| 13 | 12 | 2 |
| 14 | 13 | 2 |

**Example 2: Data set partition with random(PARTITION_METHOD=2)**

```
SET SCHEMA DM_PAL;
DROP TABLE META_DATA_TAB;
CREATE COLUMN TABLE META_DATA_TAB(
    "VARIABLE_NAME" NVARCHAR(50),
    "VARIABLE_TYPE" NVARCHAR(50)
);
INSERT INTO META_DATA_TAB VALUES ('OUTLOOK',  'CATEGORICAL');
INSERT INTO META_DATA_TAB VALUES ('CLASS',    'TARGET');
INSERT INTO META_DATA_TAB VALUES ('TEMP',     'CONTINUOUS');
INSERT INTO META_DATA_TAB VALUES ('HUMIDITY', 'CONTINUOUS');
INSERT INTO META_DATA_TAB VALUES ('WINDY',    'CATEGORICAL');
DROP TABLE PAL_DATA_TBL;
CREATE COLUMN TABLE PAL_DATA_TBL(
    "ID" INT,
    "VARIABLE_NAME" NVARCHAR(50),
    "VALUE" NVARCHAR(50),
    "PURPOSE" INT
);
INSERT INTO PAL_DATA_TBL VALUES (0,     'OUTLOOK',      'Sunny',    1);
INSERT INTO PAL_DATA_TBL VALUES (0,     'TEMP',         '75',       1);
INSERT INTO PAL_DATA_TBL VALUES (0,     'HUMIDITY',     '70',       1);
INSERT INTO PAL_DATA_TBL VALUES (0,     'WINDY',        'Yes',      1);
INSERT INTO PAL_DATA_TBL VALUES (0,     'CLASS',        '1',        1);
INSERT INTO PAL_DATA_TBL VALUES (1,     'OUTLOOK',      'Sunny',    1);
INSERT INTO PAL_DATA_TBL VALUES (1,     'TEMP',         '80',       1);
INSERT INTO PAL_DATA_TBL VALUES (1,     'HUMIDITY',     '90',       1);
```

```
INSERT INTO PAL_DATA_TBL VALUES (1,     'WINDY',        'Yes',      1);
INSERT INTO PAL_DATA_TBL VALUES (1,     'CLASS',        '0',        1);
INSERT INTO PAL_DATA_TBL VALUES (2,     'OUTLOOK',      'Sunny',      1);
INSERT INTO PAL_DATA_TBL VALUES (2,     'TEMP',         '85',        1);
INSERT INTO PAL_DATA_TBL VALUES (2,     'HUMIDITY',     '85',        1);
INSERT INTO PAL_DATA_TBL VALUES (2,     'WINDY',        'No',       1);
INSERT INTO PAL_DATA_TBL VALUES (2,     'CLASS',        '0',        1);
INSERT INTO PAL_DATA_TBL VALUES (3,     'OUTLOOK',      'Sunny',    2);
INSERT INTO PAL_DATA_TBL VALUES (3,     'TEMP',         '72',          2);
INSERT INTO PAL_DATA_TBL VALUES (3,     'HUMIDITY',     '95',          2);
INSERT INTO PAL_DATA_TBL VALUES (3,     'WINDY',        'No',          2);
INSERT INTO PAL_DATA_TBL VALUES (3,     'CLASS',        '0',           2);
INSERT INTO PAL_DATA_TBL VALUES (4,     'OUTLOOK',      'Sunny',      1);
INSERT INTO PAL_DATA_TBL VALUES (4,     'TEMP',         '69',       1);
INSERT INTO PAL_DATA_TBL VALUES (4,     'HUMIDITY',     '70',       1);
INSERT INTO PAL_DATA_TBL VALUES (4,     'WINDY',        'No',       1);
INSERT INTO PAL_DATA_TBL VALUES (4,     'CLASS',        '1',        1);
INSERT INTO PAL_DATA_TBL VALUES (5,     'OUTLOOK',      'Overcast', 1);
INSERT INTO PAL_DATA_TBL VALUES (5,     'TEMP',         '72',       1);
INSERT INTO PAL_DATA_TBL VALUES (5,     'HUMIDITY',     '90',       1);
INSERT INTO PAL_DATA_TBL VALUES (5,     'WINDY',        'Yes',      1);
INSERT INTO PAL_DATA_TBL VALUES (5,     'CLASS',        '1',        1);
INSERT INTO PAL_DATA_TBL VALUES (6,     'OUTLOOK',      'Overcast',    1);
INSERT INTO PAL_DATA_TBL VALUES (6,     'TEMP',         '83',       1);
INSERT INTO PAL_DATA_TBL VALUES (6,     'HUMIDITY',     '78',       1);
INSERT INTO PAL_DATA_TBL VALUES (6,     'WINDY',        'No',       1);
INSERT INTO PAL_DATA_TBL VALUES (6,     'CLASS',        '0',        1);
INSERT INTO PAL_DATA_TBL VALUES (7,     'OUTLOOK',      'Overcast',    1);
INSERT INTO PAL_DATA_TBL VALUES (7,     'TEMP',         '64',          1);
INSERT INTO PAL_DATA_TBL VALUES (7,     'HUMIDITY',     '65',          1);
INSERT INTO PAL_DATA_TBL VALUES (7,     'WINDY',        'Yes',         1);
INSERT INTO PAL_DATA_TBL VALUES (7,     'CLASS',        '1',           1);
INSERT INTO PAL_DATA_TBL VALUES (8,        'OUTLOOK',       'Overcast',1);
INSERT INTO PAL_DATA_TBL VALUES (8,     'TEMP',          '81',      1);
INSERT INTO PAL_DATA_TBL VALUES (8,     'HUMIDITY',      '75',      1);
INSERT INTO PAL_DATA_TBL VALUES (8,     'WINDY',         'No',      1);
INSERT INTO PAL_DATA_TBL VALUES (8,     'CLASS',         '1',       1);
INSERT INTO PAL_DATA_TBL VALUES (9,     'OUTLOOK',      'Rain',     2);
INSERT INTO PAL_DATA_TBL VALUES (9,     'TEMP',         '71',       2);
INSERT INTO PAL_DATA_TBL VALUES (9,     'HUMIDITY',     '80',       2);
INSERT INTO PAL_DATA_TBL VALUES (9,     'WINDY',        'Yes',      2);
INSERT INTO PAL_DATA_TBL VALUES (9,     'CLASS',        '0',        2);
INSERT INTO PAL_DATA_TBL VALUES (10,    'OUTLOOK',      'Rain',     1);
INSERT INTO PAL_DATA_TBL VALUES (10,    'TEMP',         '65',       1);
INSERT INTO PAL_DATA_TBL VALUES (10,    'HUMIDITY',     '70',       1);
INSERT INTO PAL_DATA_TBL VALUES (10,    'WINDY',        'Yes',      1);
INSERT INTO PAL_DATA_TBL VALUES (10,    'CLASS',        '0'    ,       1);
INSERT INTO PAL_DATA_TBL VALUES (11,    'OUTLOOK',      'Rain',     1);
INSERT INTO PAL_DATA_TBL VALUES (11,    'TEMP',         '75',       1);
INSERT INTO PAL_DATA_TBL VALUES (11,    'HUMIDITY',     '80',       1);
INSERT INTO PAL_DATA_TBL VALUES (11,    'WINDY',        'No',       1);
INSERT INTO PAL_DATA_TBL VALUES (11,    'CLASS',        '1',        1);
INSERT INTO PAL_DATA_TBL VALUES (12,    'OUTLOOK',      'Rain',     2);
INSERT INTO PAL_DATA_TBL VALUES (12,    'TEMP',         '68',       2);
INSERT INTO PAL_DATA_TBL VALUES (12,    'HUMIDITY',     '80',       2);
INSERT INTO PAL_DATA_TBL VALUES (12,    'WINDY',        'No',       2);
INSERT INTO PAL_DATA_TBL VALUES (12,    'CLASS',        '1',        2);
INSERT INTO PAL_DATA_TBL VALUES (13,    'OUTLOOK',      'Rain',     2);
INSERT INTO PAL_DATA_TBL VALUES (13,    'TEMP',         '70',       2);
INSERT INTO PAL_DATA_TBL VALUES (13,    'HUMIDITY',     '96',       2);
INSERT INTO PAL_DATA_TBL VALUES (13,    'WINDY',        'No',       2);
INSERT INTO PAL_DATA_TBL VALUES (13,    'CLASS',        '0',        2);
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
```

```
);
INSERT INTO PAL_PARAMETER_TBL VALUES ('FUNCTION',null,null,'DT');
INSERT INTO PAL_PARAMETER_TBL VALUES ('MODEL_FORMAT', 1, NULL, NULL); -- JSON
INSERT INTO PAL_PARAMETER_TBL VALUES ('SPLIT_THRESHOLD', NULL, 1e-5, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('MIN_RECORDS_PARENT', 2, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('MIN_RECORDS_LEAF', 1, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('PARTITION_METHOD',2,null,null);
INSERT INTO PAL_PARAMETER_TBL VALUES ('OUTPUT_PARTITION_RESULT',1,null,null);
INSERT INTO PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', null,0.5,null);
CALL _SYS_AFL.PAL_UNIFIED_REGRESSION_PIVOT(META_DATA_TAB, PAL_DATA_TBL,
PAL_PARAMETER_TBL, ?, ?, ?, ?, ?, ?);
```

**Excepted Result**

PARTITION RESULT

| | ID | TYPE |
|---|---|---|
| 1 | 0 | 1 |
| 2 | 1 | 1 |
| 3 | 2 | 1 |
| 4 | 3 | 1 |
| 5 | 4 | 1 |
| 6 | 8 | 1 |
| 7 | 9 | 1 |
| 8 | 10 | 1 |
| 9 | 11 | 1 |
| 10 | 12 | 1 |
| 11 | 13 | 1 |
| 12 | 5 | 2 |
| 13 | 6 | 2 |
| 14 | 7 | 2 |

## 5.13.10.4 Model Evaluation

Unified regression with pivoted input data provides one output tables as the evaluating results of the training model: statistics table.

It is generated in both the train and scoring procedures. In the training procedure, the model evaluation works only if the PARTITION_METHOD is 1 or 2.

The statistic without "TEST_" is calculated by calling the original algorithm for all training data; while the statistic with "TEST_" is calculated by the unified function of the partitioned test data. For model evaluation, it is better to use the latter.

## Statistics Table

In the statistics table, the following statistics are supported:

- R2: R-squared (R2) is a statistical measure that represents the proportion of the variance for a dependent variable that's explained by an independent variable or variables in a regression model. R2 is calculated by the following equation:

  $$R^2 = 1 - \frac{RSS}{TSS} = 1 - \frac{\sum_i (y_i - y_i')^2}{\sum_i (y_i - \bar{y})^2}$$

  Where $y_i$ is an observed value, $y'_i$ is an expected value and $\bar{y}$ is a mean observed value.
- MSE: See Mean squared error for more details;
- RMSE: See Root mean squared error for more details;
- MAE: Mean Absolute Error (MAE) is a measure of errors between paired observations expressing the same phenomenon. MAE is calculated by the following equation:

  $$MAE = \frac{1}{m}\sum_i^m |y_i - y_i'|$$

  Where $y_i$ is an observed value and $y'_i$ is an expected value.
- MAPE: See Mean absolute percentage error for more details;
- WMAPE: See Weighted mean absolute percentage error for more details;
- EVar: Explained Variance(EVar) measures the proportion to which a mathematical model accounts for the variation (dispersion) of a given data set. EVar is calculated by the following equation:

  $$EVar = 1 - \frac{Var\{y-y'\}}{Var\{y\}}$$

  Where `Var` is Variance and calculated by the following equation:

  $$Var = \frac{\sum (x_i - \bar{x})^2}{n-1}.$$
- Max Error: MAx Error represents the maximum absolute difference between the observed value and the expected value. Max Error is calculated by the following equation:
  `Max_Error=MAX(|y-y'|)`
  Where `y` is an observed value and y' is an expected value.
- MSLE: Mean squared logarithmic error(MSLE) computes a risk metric corresponding to the expected value of the squared logarithmic (quadratic) error or loss. MSLE is calculated by the following equation:

  $$MSLE = \frac{1}{n}\sum_i^{n-1} (\log_e(1+y_i) - \log_e(1+y_i'))^2$$

  Where `y` is the observed value, `y'` is the expected value. It's only for exponential regression.

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA DM_PAL;
DROP TABLE META_DATA_TAB;
CREATE COLUMN TABLE META_DATA_TAB(
    "VARIABLE_NAME" NVARCHAR(50),
    "VARIABLE_TYPE" NVARCHAR(50)
);
INSERT INTO META_DATA_TAB VALUES ('OUTLOOK',  'CATEGORICAL');
INSERT INTO META_DATA_TAB VALUES ('CLASS',    'TARGET');
```

```
INSERT INTO META_DATA_TAB VALUES ('TEMP',     'CONTINUOUS');
INSERT INTO META_DATA_TAB VALUES ('HUMIDITY', 'CONTINUOUS');
INSERT INTO META_DATA_TAB VALUES ('WINDY',    'CATEGORICAL');
DROP TABLE PAL_DATA_TBL;
CREATE COLUMN TABLE PAL_DATA_TBL(
    "ID" INT,
    "VARIABLE_NAME" NVARCHAR(50),
    "VALUE" NVARCHAR(50),
    "PURPOSE" INT
);
INSERT INTO PAL_DATA_TBL VALUES (0,     'OUTLOOK',      'Sunny',    1);
INSERT INTO PAL_DATA_TBL VALUES (0,     'TEMP',         '75',       1);
INSERT INTO PAL_DATA_TBL VALUES (0,     'HUMIDITY',     '70',       1);
INSERT INTO PAL_DATA_TBL VALUES (0,     'WINDY',        'Yes',      1);
INSERT INTO PAL_DATA_TBL VALUES (0,     'CLASS',        '1',        1);
INSERT INTO PAL_DATA_TBL VALUES (1,     'OUTLOOK',      'Sunny',    1);
INSERT INTO PAL_DATA_TBL VALUES (1,     'TEMP',         '80',       1);
INSERT INTO PAL_DATA_TBL VALUES (1,     'HUMIDITY',     '90',       1);
INSERT INTO PAL_DATA_TBL VALUES (1,     'WINDY',        'Yes',      1);
INSERT INTO PAL_DATA_TBL VALUES (1,     'CLASS',        '0',        1);
INSERT INTO PAL_DATA_TBL VALUES (2,     'OUTLOOK',      'Sunny',      1);
INSERT INTO PAL_DATA_TBL VALUES (2,     'TEMP',         '85',        1);
INSERT INTO PAL_DATA_TBL VALUES (2,     'HUMIDITY',     '85',        1);
INSERT INTO PAL_DATA_TBL VALUES (2,     'WINDY',        'No',       1);
INSERT INTO PAL_DATA_TBL VALUES (2,     'CLASS',        '0',        1);
INSERT INTO PAL_DATA_TBL VALUES (3,     'OUTLOOK',      'Sunny',    2);
INSERT INTO PAL_DATA_TBL VALUES (3,     'TEMP',         '72',          2);
INSERT INTO PAL_DATA_TBL VALUES (3,     'HUMIDITY',     '95',          2);
INSERT INTO PAL_DATA_TBL VALUES (3,     'WINDY',        'No',          2);
INSERT INTO PAL_DATA_TBL VALUES (3,     'CLASS',        '0',           2);
INSERT INTO PAL_DATA_TBL VALUES (4,     'OUTLOOK',      'Sunny',      1);
INSERT INTO PAL_DATA_TBL VALUES (4,     'TEMP',         '69',       1);
INSERT INTO PAL_DATA_TBL VALUES (4,     'HUMIDITY',     '70',       1);
INSERT INTO PAL_DATA_TBL VALUES (4,     'WINDY',        'No',       1);
INSERT INTO PAL_DATA_TBL VALUES (4,     'CLASS',        '1',        1);
INSERT INTO PAL_DATA_TBL VALUES (5,     'OUTLOOK',      'Overcast', 1);
INSERT INTO PAL_DATA_TBL VALUES (5,     'TEMP',         '72',       1);
INSERT INTO PAL_DATA_TBL VALUES (5,     'HUMIDITY',     '90',       1);
INSERT INTO PAL_DATA_TBL VALUES (5,     'WINDY',        'Yes',      1);
INSERT INTO PAL_DATA_TBL VALUES (5,     'CLASS',        '1',        1);
INSERT INTO PAL_DATA_TBL VALUES (6,     'OUTLOOK',      'Overcast',    1);
INSERT INTO PAL_DATA_TBL VALUES (6,     'TEMP',         '83',       1);
INSERT INTO PAL_DATA_TBL VALUES (6,     'HUMIDITY',     '78',       1);
INSERT INTO PAL_DATA_TBL VALUES (6,     'WINDY',        'No',       1);
INSERT INTO PAL_DATA_TBL VALUES (6,     'CLASS',        '0',        1);
INSERT INTO PAL_DATA_TBL VALUES (7,     'OUTLOOK',      'Overcast',    1);
INSERT INTO PAL_DATA_TBL VALUES (7,     'TEMP',         '64',          1);
INSERT INTO PAL_DATA_TBL VALUES (7,     'HUMIDITY',     '65',          1);
INSERT INTO PAL_DATA_TBL VALUES (7,     'WINDY',        'Yes',         1);
INSERT INTO PAL_DATA_TBL VALUES (7,     'CLASS',        '1',           1);
INSERT INTO PAL_DATA_TBL VALUES (8,        'OUTLOOK',       'Overcast',1);
INSERT INTO PAL_DATA_TBL VALUES (8,     'TEMP',          '81',      1);
INSERT INTO PAL_DATA_TBL VALUES (8,     'HUMIDITY',      '75',      1);
INSERT INTO PAL_DATA_TBL VALUES (8,     'WINDY',         'No',      1);
INSERT INTO PAL_DATA_TBL VALUES (8,     'CLASS',         '1',       1);
INSERT INTO PAL_DATA_TBL VALUES (9,     'OUTLOOK',      'Rain',     2);
INSERT INTO PAL_DATA_TBL VALUES (9,     'TEMP',         '71',       2);
INSERT INTO PAL_DATA_TBL VALUES (9,     'HUMIDITY',     '80',       2);
INSERT INTO PAL_DATA_TBL VALUES (9,     'WINDY',        'Yes',      2);
INSERT INTO PAL_DATA_TBL VALUES (9,     'CLASS',        '0',        2);
INSERT INTO PAL_DATA_TBL VALUES (10,    'OUTLOOK',      'Rain',     1);
INSERT INTO PAL_DATA_TBL VALUES (10,    'TEMP',         '65',       1);
INSERT INTO PAL_DATA_TBL VALUES (10,    'HUMIDITY',     '70',       1);
INSERT INTO PAL_DATA_TBL VALUES (10,    'WINDY',        'Yes',      1);
INSERT INTO PAL_DATA_TBL VALUES (10,    'CLASS',        '0'    ,       1);
INSERT INTO PAL_DATA_TBL VALUES (11,    'OUTLOOK',      'Rain',     1);
INSERT INTO PAL_DATA_TBL VALUES (11,    'TEMP',         '75',       1);
INSERT INTO PAL_DATA_TBL VALUES (11,    'HUMIDITY',     '80',       1);
```

```
INSERT INTO PAL_DATA_TBL VALUES (11,    'WINDY',        'No',       1);
INSERT INTO PAL_DATA_TBL VALUES (11,    'CLASS',        '1',        1);
INSERT INTO PAL_DATA_TBL VALUES (12,    'OUTLOOK',      'Rain',     2);
INSERT INTO PAL_DATA_TBL VALUES (12,    'TEMP',         '68',       2);
INSERT INTO PAL_DATA_TBL VALUES (12,    'HUMIDITY',     '80',       2);
INSERT INTO PAL_DATA_TBL VALUES (12,    'WINDY',        'No',       2);
INSERT INTO PAL_DATA_TBL VALUES (12,    'CLASS',        '1',        2);
INSERT INTO PAL_DATA_TBL VALUES (13,    'OUTLOOK',      'Rain',     2);
INSERT INTO PAL_DATA_TBL VALUES (13,    'TEMP',         '70',       2);
INSERT INTO PAL_DATA_TBL VALUES (13,    'HUMIDITY',     '96',       2);
INSERT INTO PAL_DATA_TBL VALUES (13,    'WINDY',        'No',       2);
INSERT INTO PAL_DATA_TBL VALUES (13,    'CLASS',        '0',        2);
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO PAL_PARAMETER_TBL VALUES ('FUNCTION',null,null,'DT');
INSERT INTO PAL_PARAMETER_TBL VALUES ('MODEL_FORMAT', 1, NULL, NULL); -- JSON
INSERT INTO PAL_PARAMETER_TBL VALUES ('SPLIT_THRESHOLD', NULL, 1e-5, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('MIN_RECORDS_PARENT', 2, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('MIN_RECORDS_LEAF', 1, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('PARTITION_METHOD',1,null,null);
INSERT INTO PAL_PARAMETER_TBL VALUES ('OUTPUT_PARTITION_RESULT',1,null,null);
INSERT INTO PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', null,0.5,null);
--INSERT INTO PAL_PARAMETER_TBL VALUES ('LABEL', null,null,'OUTLOOK');--Throw
error
CALL _SYS_AFL.PAL_UNIFIED_REGRESSION_PIVOT(META_DATA_TAB, PAL_DATA_TBL,
PAL_PARAMETER_TBL, ?, ?, ?, ?, ?, ?);
```

**Excepted Result**

STATISTIC

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | TEST_EVAR | -0.33333333333333326 |
| 2 | TEST_MAE | 0.5 |
| 3 | TEST_MAPE | 0.5 |
| 4 | TEST_MAX_ERROR | 1 |
| 5 | TEST_MSE | 0.5 |
| 6 | TEST_R2 | -1.6666666666666665 |
| 7 | TEST_RMSE | 0.7071067811865476 |
| 8 | TEST_WMAPE | 2 |

## 5.13.10.5 Model Compression

The training process may cost a large size of memory. Model compression aims to reduce the size of the model with minimum loss of the accuracy.

Model compression is provided for Support Vector, Random Decision Trees, and Hybrid Gradient Boosting Tree. The following tables list the parameters for using model compression in Support Vector Machine, Random Decision Trees, and Hybrid Gradient Boosting Tree respectively.

**Support Vector Machine**

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| COMPRESSION | INTEGER | 1 | Specifies if the model is stored in compressed format.<br>• 0: no;<br>• 1: yes. | |
| MAX_BITS | INTEGER | 12 | The maximum number of bits to quantize continuous features. Equivalent to use $2^{MAX_BITS}$ bins.<br>Must be less than 31. | Valid only when the value of COMPRESSION is 1. |
| MAX_QUANTIZATION_ITER | INTEGER | 1000 | The maximum iteration steps for quantization. | Valid only when the value of COMPRESSION is 1. |

### Random Decision Trees

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| COMPRESSION | INTEGER | 0 | Specifies if the model is stored in compressed format.<br>• 0: no;<br>• 1: yes. | |
| MAX_BITS | INTEGER | 12 | The maximum number of bits to quantize continuous features. Equivalent to use $2^{MAX_BITS}$ bins.<br>Must be less than 31. | Valid only when the value of COMPRESSION is 1. |
| QUANTIZE_RATE | DOUBLE | 0.005 | Quantize a feature if the largest class frequency of which is less than QUANTIZE_RATE. | Valid only when the value of COMPRESSION is 1. |
| FITTINGS_QUANTIZATION | INTEGER | 0 | Indicate whether to quantize the fitting values.<br>• 0: no;<br>• 1: yes. | |

### Hybrid Gradient Boosting Tree

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| COMPRESSION | INTEGER | 0 | Indicates whether result model should be compressed:<br>• 0. no;<br>• 1. yes. | |
| MAX_BITS | INTEGER | 12 | Maximum number of bits to quantize continuous features. Equivalent to use $2^{MAX_BITS}$ bins. | Must be less than 31. |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA DM_PAL;
DROP TABLE META_DATA_TAB;
CREATE COLUMN TABLE META_DATA_TAB(
    "VARIABLE_NAME" NVARCHAR(50),
    "VARIABLE_TYPE" NVARCHAR(50)
);
INSERT INTO META_DATA_TAB VALUES ('OUTLOOK',  'CATEGORICAL');
INSERT INTO META_DATA_TAB VALUES ('CLASS',    'TARGET');
INSERT INTO META_DATA_TAB VALUES ('TEMP',     'CONTINUOUS');
INSERT INTO META_DATA_TAB VALUES ('HUMIDITY', 'CONTINUOUS');
INSERT INTO META_DATA_TAB VALUES ('WINDY',    'CATEGORICAL');
DROP TABLE PAL_DATA_TBL;
CREATE COLUMN TABLE PAL_DATA_TBL(
    "ID" INT,
    "VARIABLE_NAME" NVARCHAR(50),
    "VALUE" NVARCHAR(50),
    "PURPOSE" INT
);
INSERT INTO PAL_DATA_TBL VALUES (0,     'OUTLOOK',      'Sunny',    1);
INSERT INTO PAL_DATA_TBL VALUES (0,     'TEMP',         '75',       1);
INSERT INTO PAL_DATA_TBL VALUES (0,     'HUMIDITY',     '70',       1);
INSERT INTO PAL_DATA_TBL VALUES (0,     'WINDY',        'Yes',      1);
INSERT INTO PAL_DATA_TBL VALUES (0,     'CLASS',        '1',        1);
INSERT INTO PAL_DATA_TBL VALUES (1,     'OUTLOOK',      'Sunny',    1);
INSERT INTO PAL_DATA_TBL VALUES (1,     'TEMP',         'null',       1);
INSERT INTO PAL_DATA_TBL VALUES (1,     'HUMIDITY',     '90',       1);
INSERT INTO PAL_DATA_TBL VALUES (1,     'WINDY',        'Yes',      1);
INSERT INTO PAL_DATA_TBL VALUES (1,     'CLASS',        '0',        1);
INSERT INTO PAL_DATA_TBL VALUES (2,     'OUTLOOK',      'Sunny',      1);
INSERT INTO PAL_DATA_TBL VALUES (2,     'TEMP',         '85',        1);
INSERT INTO PAL_DATA_TBL VALUES (2,     'HUMIDITY',     'null',        1);
INSERT INTO PAL_DATA_TBL VALUES (2,     'WINDY',        'No',       1);
INSERT INTO PAL_DATA_TBL VALUES (2,     'CLASS',        '0',        1);
INSERT INTO PAL_DATA_TBL VALUES (3,     'OUTLOOK',      'Sunny',    2);
INSERT INTO PAL_DATA_TBL VALUES (3,     'TEMP',         '72',          2);
INSERT INTO PAL_DATA_TBL VALUES (3,     'HUMIDITY',     '95',          2);
INSERT INTO PAL_DATA_TBL VALUES (3,     'WINDY',        'No',          2);
INSERT INTO PAL_DATA_TBL VALUES (3,     'CLASS',        '0',           2);
INSERT INTO PAL_DATA_TBL VALUES (4,     'OUTLOOK',      'null',      1);
INSERT INTO PAL_DATA_TBL VALUES (4,     'TEMP',         'null',       1);
INSERT INTO PAL_DATA_TBL VALUES (4,     'HUMIDITY',     '70',       1);
```

```
INSERT INTO PAL_DATA_TBL VALUES (4,     'WINDY',        'null',       1);
INSERT INTO PAL_DATA_TBL VALUES (4,     'CLASS',        '1',        1);
INSERT INTO PAL_DATA_TBL VALUES (5,     'OUTLOOK',      'Overcast', 1);
INSERT INTO PAL_DATA_TBL VALUES (5,     'TEMP',         '72',       1);
INSERT INTO PAL_DATA_TBL VALUES (5,     'HUMIDITY',     '90',       1);
INSERT INTO PAL_DATA_TBL VALUES (5,     'WINDY',        'Yes',      1);
INSERT INTO PAL_DATA_TBL VALUES (5,     'CLASS',        '1',        1);
INSERT INTO PAL_DATA_TBL VALUES (6,     'OUTLOOK',      'Overcast',    1);
INSERT INTO PAL_DATA_TBL VALUES (6,     'TEMP',         '83',       1);
INSERT INTO PAL_DATA_TBL VALUES (6,     'HUMIDITY',     '78',       1);
INSERT INTO PAL_DATA_TBL VALUES (6,     'WINDY',        'No',       1);
INSERT INTO PAL_DATA_TBL VALUES (6,     'CLASS',        '0',        1);
INSERT INTO PAL_DATA_TBL VALUES (7,     'OUTLOOK',      'Overcast',    1);
INSERT INTO PAL_DATA_TBL VALUES (7,     'TEMP',         '64',          1);
INSERT INTO PAL_DATA_TBL VALUES (7,     'HUMIDITY',     '65',          1);
INSERT INTO PAL_DATA_TBL VALUES (7,     'WINDY',        'Yes',         1);
INSERT INTO PAL_DATA_TBL VALUES (7,     'CLASS',        '1',           1);
INSERT INTO PAL_DATA_TBL VALUES (8,        'OUTLOOK',       'Overcast',1);
INSERT INTO PAL_DATA_TBL VALUES (8,     'TEMP',          '81',      1);
INSERT INTO PAL_DATA_TBL VALUES (8,     'HUMIDITY',      '75',      1);
INSERT INTO PAL_DATA_TBL VALUES (8,     'WINDY',         'No',      1);
INSERT INTO PAL_DATA_TBL VALUES (8,     'CLASS',         '1',       1);
INSERT INTO PAL_DATA_TBL VALUES (9,     'OUTLOOK',      'null',     2);
INSERT INTO PAL_DATA_TBL VALUES (9,     'TEMP',         '71',       2);
INSERT INTO PAL_DATA_TBL VALUES (9,     'HUMIDITY',     '80',       2);
INSERT INTO PAL_DATA_TBL VALUES (9,     'WINDY',        'Yes',      2);
INSERT INTO PAL_DATA_TBL VALUES (9,     'CLASS',        '0',        2);
INSERT INTO PAL_DATA_TBL VALUES (10,    'OUTLOOK',      'Rain',     1);
INSERT INTO PAL_DATA_TBL VALUES (10,    'TEMP',         '65',       1);
INSERT INTO PAL_DATA_TBL VALUES (10,    'HUMIDITY',     '70',       1);
INSERT INTO PAL_DATA_TBL VALUES (10,    'WINDY',        'Yes',      1);
INSERT INTO PAL_DATA_TBL VALUES (10,    'CLASS',        '0'    ,       1);
INSERT INTO PAL_DATA_TBL VALUES (11,    'OUTLOOK',      'Rain',     1);
INSERT INTO PAL_DATA_TBL VALUES (11,    'TEMP',         '75',       1);
INSERT INTO PAL_DATA_TBL VALUES (11,    'HUMIDITY',     '80',       1);
INSERT INTO PAL_DATA_TBL VALUES (11,    'WINDY',        'No',       1);
INSERT INTO PAL_DATA_TBL VALUES (11,    'CLASS',        '1',        1);
INSERT INTO PAL_DATA_TBL VALUES (12,    'OUTLOOK',      'Rain',     2);
INSERT INTO PAL_DATA_TBL VALUES (12,    'TEMP',         '68',       2);
INSERT INTO PAL_DATA_TBL VALUES (12,    'HUMIDITY',     '80',       2);
INSERT INTO PAL_DATA_TBL VALUES (12,    'WINDY',        'No',       2);
INSERT INTO PAL_DATA_TBL VALUES (12,    'CLASS',        '1',        2);
INSERT INTO PAL_DATA_TBL VALUES (13,    'OUTLOOK',      'Rain',     2);
INSERT INTO PAL_DATA_TBL VALUES (13,    'TEMP',         '70',       2);
INSERT INTO PAL_DATA_TBL VALUES (13,    'HUMIDITY',     '96',       2);
INSERT INTO PAL_DATA_TBL VALUES (13,    'WINDY',        'No',       2);
INSERT INTO PAL_DATA_TBL VALUES (13,    'CLASS',        '0',        2);
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO PAL_PARAMETER_TBL VALUES ('FUNCTION', NULL, NULL, 'RDT');
INSERT INTO PAL_PARAMETER_TBL VALUES ('SEED', 2, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('N_ESTIMATORS', 200, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('MAX_FEATURES', 3, NULL, NULL);
INSERT INTO PAL_PARAMETER_TBL VALUES ('COMPRESSION', 1, NULL,
NULL);                      ------parameter for model compression
--CALL _SYS_AFL.PAL_UNIFIED_REGRESSION_PIVOT(META_DATA_TAB, PAL_DATA_TBL,
PAL_PARAMETER_TBL, ?, ?, ?, ?, ?, ?);
DROP TABLE PAL_DATA_MODEL_TBL;
CREATE COLUMN TABLE PAL_DATA_MODEL_TBL(
     "ROW_INDEX" INTEGER,
     "PART_INDEX" INTEGER,
     "MODEL_CONTENT" VARCHAR (5000)
);
```

```
DO BEGIN
  meta_data_tab = SELECT * FROM META_DATA_TAB;
  data_tab = SELECT * FROM PAL_DATA_TBL;
  control_tab = SELECT * FROM PAL_PARAMETER_TBL;
  CALL
_SYS_AFL.PAL_UNIFIED_REGRESSION_PIVOT(:meta_data_tab, :data_tab, :control_tab,
model_tab, stats_tab, optimal_parameter_tab, partition_result,
place_holder_1_tab, place_holder_2_tab);
  INSERT into PAL_DATA_MODEL_TBL SELECT * FROM :model_tab;
END
SELECT * FROM PAL_DATA_MODEL_TBL;
```

**Excepted Result**

MODEL

| | ROW_INDEX | PART_INDEX | MODEL_CONTENT |
|---|---|---|---|
| 1 | 0 | -1 | {"atFields":[{"dataType":"string","name":"OUTLOOK","opType":"cat","values":["Sunny"]},{"... |
| 2 | 1 | 0 | {"cw":"e\u0019%EF1L)&FF\u0018ppppddq'\n1FL4bRRRRJ8SE\u0015\u0015\u0015\u0015\u0015\u... |
| 3 | 2 | 1 | {"cw":"/&. /&. \u001c:\u0002\u000e\u001d\u0001\u0006\u0005\u0004?\u0003\u0002B\u0... |
| 4 | 3 | 2 | {"cw":"/2\r\u0006*\n\u000bIC!JBBtvTX^\u000bSZRbxG\u001c#\u001e6dG\u001c#\u001e6... |
| 5 | 4 | 2 | 1\u000e\u0004\n\u0006\u0004\u0003\u0001\u000e\u0005\t\u0006\u0003\u0001\u0002\u0... |
| 6 | 5 | 2 | u0002\u0001\u0007\u0002\u0001\u0001\u0005\u0004\u000e\u0007\u0006\u0001\u0007\u... |
| 7 | 6 | 3 | {"at":0,"cvs":["\u0000","\u0003\u0002\u0001","\u0002","\u0001","\u0003\u0001\u0000","\u... |
| 8 | 7 | 3 | 00b\u0006\u0007\u0006\u0007\u0006\u0007\u0006\u0007\n\u000b\n\u000b\n\u000b\n\... |
| 9 | 8 | 4 | {"at":1,"cvs":["73.5","82","79","73","71","66.5","71.5","70","64.5","69","77.5","68.5","69.5","80... |
| 10 | 9 | 4 | \u0080\u0081\u0088\u0089\u008a\u008b\u008e\u008f\u0096\u0097\u0098\u0099\u009a\... |
| 11 | 10 | 5 | {"at":2,"cvs":["76.5","93","92.5","82.5","88","75","87.5","67.5","77.5","72.5","74","85","79"],"di... |
| 12 | 11 | 5 | 2\u0002","trees":"67<=vw"},{"cw":"\u0004\u0000","idict":4,"nodes":"\u0007\b\u0007\b","n... |
| 13 | 12 | 6 | {"at":3,"cvs":["\u0000","\u0002\u0001","\u0001","\u0002\u0000"],"dicts":[[{"p":0.3125,"v":0}... |
| 14 | -1 | ? | RDT |
| 15 | -2 | ? | {"feature":["OUTLOOK","TEMP","HUMIDITY","WINDY","CLASS"],"featureNum":5,"index":[0... |

## 5.13.10.6 Local Interpretability of Models

In the process of machine learning predictive modeling, you can also view why the predicative results are made.

Based on the Shapley Values in game theory, Shapley Additive Explanations (SHAP) was proposed to explain individual predictions.

When it comes to specified machine learning model, the implementations of SHAP values have variant versions.

For tree based machine learning models, there are two explainers for the tree model: tree SHAP and Saabas. Tree SHAP explainer can exactly compute the Shapley values in polynomial time and satisfy the interpretation consistency. Saabas explainer is also implemented to calculate the attribution values, which takes only the single decision path into account and could run in a logarithmic time (over the number of nodes).

For linear based models, for example, Multiple Linear Regression, a linear SHAP expainer(linear SHAP) is provided. Linear SHAP uses samplings to estimate the SHAP values. The samplings called background data are sampled from the train dataset in the training process and stored in the model table.

For nonlinear and non-tree based models, for example, Support Vector Machine, a kernel SHAP explainer (kernel SHAP) is provided. Kernel SHAP does the same thing as linear SHAP, but using a kernel sampling. It is also be sampled from the train dataset in the training process and stored in the model table.

As mentioned in Exponential Regression [page 448], `y'` and `x1...xn` is a linear relationship and can be solved using the linear regression method. Therefore, the `y'(ln(y))` is explained via the linear SHAP of EXP, not the original output y.

As mentioned in Generalised Linear Models [page 456] (GLM), compared with the classical linear regression, GLM regresses a linear predictor η instead of the response itself. Therefore, the η is explained via the linear SHAP of GLM, not the original output `y`. For example, `η=log(μ)` when the link function is Log.

The following table gives an overview of the supported SHAP versions for different methods.

| Function Name | Method |
|---|---|
| DT | • tree SHAP<br>• Saabas |
| RDT | • tree SHAP<br>• Saabas |
| HGBT | • tree SHAP<br>• Saabas |
| EXP | linear SHAP |
| GLM | linear SHAP |
| MLR | linear SHAP |
| MLP | kernel SHAP |
| SVM | kernel SHAP |

## _SYS_AFL.PAL_UNIFIED_REGRESSION

### Parameter Table

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| BACKGROUND_SIZE | INTEGER | 0 | The row size of background data. | It should not be larger than the row size of train data. Valid for the functions:<br>• EXP<br>• GLM<br>• MLR<br>• MLp<br>• SVM |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| BACKGROUND_SAMPLING_SEED | INTEGER | 0 | Specified the seed for random number generator in the background sampling:<br>• 0: Uses the current time(in second) as seed;<br>• Others: Uses the specified value as seed. | Valid for the functions:<br>• EXP<br>• GLM<br>• MLR<br>• MLP<br>• SVM |

### Examples

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

#### Example 1: tree SHAP(DT)

```
SET SCHEMA DM_PAL;
DROP TABLE META_DATA_TAB;
CREATE COLUMN TABLE META_DATA_TAB(
    "VARIABLE_NAME" NVARCHAR(50),
    "VARIABLE_TYPE" NVARCHAR(50)
);
INSERT INTO META_DATA_TAB VALUES ('OUTLOOK',  'CATEGORICAL');
INSERT INTO META_DATA_TAB VALUES ('CLASS',    'TARGET');
INSERT INTO META_DATA_TAB VALUES ('TEMP',     'CONTINUOUS');
INSERT INTO META_DATA_TAB VALUES ('HUMIDITY', 'CONTINUOUS');
INSERT INTO META_DATA_TAB VALUES ('WINDY',    'CATEGORICAL');
DROP TABLE PAL_DATA_TBL;
CREATE COLUMN TABLE PAL_DATA_TBL(
    "ID" INT,
    "VARIABLE_NAME" NVARCHAR(50),
    "VALUE" NVARCHAR(50),
    "PURPOSE" INT
);
INSERT INTO PAL_DATA_TBL VALUES (0,     'OUTLOOK',      'Sunny',    1);
INSERT INTO PAL_DATA_TBL VALUES (0,     'TEMP',         '75',       1);
INSERT INTO PAL_DATA_TBL VALUES (0,     'HUMIDITY',     '70',       1);
INSERT INTO PAL_DATA_TBL VALUES (0,     'WINDY',        'Yes',      1);
INSERT INTO PAL_DATA_TBL VALUES (0,     'CLASS',        '1',        1);
INSERT INTO PAL_DATA_TBL VALUES (1,     'OUTLOOK',      'Sunny',    1);
INSERT INTO PAL_DATA_TBL VALUES (1,     'TEMP',         '80',       1);
INSERT INTO PAL_DATA_TBL VALUES (1,     'HUMIDITY',     '90',       1);
INSERT INTO PAL_DATA_TBL VALUES (1,     'WINDY',        'Yes',      1);
INSERT INTO PAL_DATA_TBL VALUES (1,     'CLASS',        '0',        1);
INSERT INTO PAL_DATA_TBL VALUES (2,     'OUTLOOK',      'Sunny',      1);
INSERT INTO PAL_DATA_TBL VALUES (2,     'TEMP',         '85',        1);
INSERT INTO PAL_DATA_TBL VALUES (2,     'HUMIDITY',     '85',        1);
INSERT INTO PAL_DATA_TBL VALUES (2,     'WINDY',        'No',       1);
INSERT INTO PAL_DATA_TBL VALUES (2,     'CLASS',        '0',        1);
INSERT INTO PAL_DATA_TBL VALUES (3,     'OUTLOOK',      'Sunny',    2);
INSERT INTO PAL_DATA_TBL VALUES (3,     'TEMP',         '72',          2);
INSERT INTO PAL_DATA_TBL VALUES (3,     'HUMIDITY',     '95',          2);
INSERT INTO PAL_DATA_TBL VALUES (3,     'WINDY',        'No',          2);
INSERT INTO PAL_DATA_TBL VALUES (3,     'CLASS',        '0',          2);
INSERT INTO PAL_DATA_TBL VALUES (4,     'OUTLOOK',      'Sunny',      1);
```

```
INSERT INTO PAL_DATA_TBL VALUES (4,     'TEMP',         '69',       1);
INSERT INTO PAL_DATA_TBL VALUES (4,     'HUMIDITY',     '70',       1);
INSERT INTO PAL_DATA_TBL VALUES (4,     'WINDY',        'No',       1);
INSERT INTO PAL_DATA_TBL VALUES (4,     'CLASS',        '1',        1);
INSERT INTO PAL_DATA_TBL VALUES (5,     'OUTLOOK',      'Overcast', 1);
INSERT INTO PAL_DATA_TBL VALUES (5,     'TEMP',         '72',       1);
INSERT INTO PAL_DATA_TBL VALUES (5,     'HUMIDITY',     '90',       1);
INSERT INTO PAL_DATA_TBL VALUES (5,     'WINDY',        'Yes',      1);
INSERT INTO PAL_DATA_TBL VALUES (5,     'CLASS',        '1',        1);
INSERT INTO PAL_DATA_TBL VALUES (6,     'OUTLOOK',      'Overcast',    1);
INSERT INTO PAL_DATA_TBL VALUES (6,     'TEMP',         '83',       1);
INSERT INTO PAL_DATA_TBL VALUES (6,     'HUMIDITY',     '78',       1);
INSERT INTO PAL_DATA_TBL VALUES (6,     'WINDY',        'No',       1);
INSERT INTO PAL_DATA_TBL VALUES (6,     'CLASS',        '0',        1);
INSERT INTO PAL_DATA_TBL VALUES (7,     'OUTLOOK',      'Overcast',    1);
INSERT INTO PAL_DATA_TBL VALUES (7,     'TEMP',         '64',          1);
INSERT INTO PAL_DATA_TBL VALUES (7,     'HUMIDITY',     '65',          1);
INSERT INTO PAL_DATA_TBL VALUES (7,     'WINDY',        'Yes',         1);
INSERT INTO PAL_DATA_TBL VALUES (7,     'CLASS',        '1',           1);
INSERT INTO PAL_DATA_TBL VALUES (8,        'OUTLOOK',       'Overcast',1);
INSERT INTO PAL_DATA_TBL VALUES (8,     'TEMP',          '81',      1);
INSERT INTO PAL_DATA_TBL VALUES (8,     'HUMIDITY',      '75',      1);
INSERT INTO PAL_DATA_TBL VALUES (8,     'WINDY',         'No',      1);
INSERT INTO PAL_DATA_TBL VALUES (8,     'CLASS',         '1',       1);
INSERT INTO PAL_DATA_TBL VALUES (9,     'OUTLOOK',      'Rain',     2);
INSERT INTO PAL_DATA_TBL VALUES (9,     'TEMP',         '71',       2);
INSERT INTO PAL_DATA_TBL VALUES (9,     'HUMIDITY',     '80',       2);
INSERT INTO PAL_DATA_TBL VALUES (9,     'WINDY',        'Yes',      2);
INSERT INTO PAL_DATA_TBL VALUES (9,     'CLASS',        '0',        2);
INSERT INTO PAL_DATA_TBL VALUES (10,    'OUTLOOK',      'Rain',     1);
INSERT INTO PAL_DATA_TBL VALUES (10,    'TEMP',         '65',       1);
INSERT INTO PAL_DATA_TBL VALUES (10,    'HUMIDITY',     '70',       1);
INSERT INTO PAL_DATA_TBL VALUES (10,    'WINDY',        'Yes',      1);
INSERT INTO PAL_DATA_TBL VALUES (10,    'CLASS',        '0'    ,       1);
INSERT INTO PAL_DATA_TBL VALUES (11,    'OUTLOOK',      'Rain',     1);
INSERT INTO PAL_DATA_TBL VALUES (11,    'TEMP',         '75',       1);
INSERT INTO PAL_DATA_TBL VALUES (11,    'HUMIDITY',     '80',       1);
INSERT INTO PAL_DATA_TBL VALUES (11,    'WINDY',        'No',       1);
INSERT INTO PAL_DATA_TBL VALUES (11,    'CLASS',        '1',        1);
INSERT INTO PAL_DATA_TBL VALUES (12,    'OUTLOOK',      'Rain',     2);
INSERT INTO PAL_DATA_TBL VALUES (12,    'TEMP',         '68',       2);
INSERT INTO PAL_DATA_TBL VALUES (12,    'HUMIDITY',     '80',       2);
INSERT INTO PAL_DATA_TBL VALUES (12,    'WINDY',        'No',       2);
INSERT INTO PAL_DATA_TBL VALUES (12,    'CLASS',        '1',        2);
INSERT INTO PAL_DATA_TBL VALUES (13,    'OUTLOOK',      'Rain',     2);
INSERT INTO PAL_DATA_TBL VALUES (13,    'TEMP',         '70',       2);
INSERT INTO PAL_DATA_TBL VALUES (13,    'HUMIDITY',     '96',       2);
INSERT INTO PAL_DATA_TBL VALUES (13,    'WINDY',        'No',       2);
INSERT INTO PAL_DATA_TBL VALUES (13,    'CLASS',        '0',        2);
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO PAL_PARAMETER_TBL VALUES ('FUNCTION',null,null,'DT');
DROP TABLE PAL_DATA_MODEL_TBL;
CREATE COLUMN TABLE PAL_DATA_MODEL_TBL(
     "ROW_INDEX" INTEGER,
     "PART_INDEX" INTEGER,
     "MODEL_CONTENT" VARCHAR (5000)
);
DO BEGIN
  meta_data_tab = SELECT * FROM META_DATA_TAB;
  data_tab = SELECT * FROM PAL_DATA_TBL;
  control_tab = SELECT * FROM PAL_PARAMETER_TBL;
```

```
  CALL
_SYS_AFL.PAL_UNIFIED_REGRESSION_PIVOT(:meta_data_tab, :data_tab, :control_tab,
model_tab, fit_tab, stats_tab, optimal_parameter_tab, place_holder_1_tab,
place_holder_2_tab);
  INSERT into PAL_DATA_MODEL_TBL SELECT * FROM :model_tab;
END
```

### Expected Result

MODEL:

| | ROW_INDEX | PART_INDEX | MODEL_CONTENT |
|---|---|---|---|
| 1 | 0 | ? | {"CurrentVersion":"VERSION1.3","DataDictionary":[{"DataType":"String","ID":0,"Name":"OUTLOOK","OpType":"Catego... |
| 2 | -1 | ? | DT |
| 3 | -2 | ? | {"feature":["OUTLOOK","TEMP","HUMIDITY","WINDY","CLASS"],"featureNum":5,"index":[0,1,2,3,4],"type":["CATEGOR... |

### Example 2: Based linear SHAP(MLR)

```
SET SCHEMA DM_PAL;
DROP TABLE META_DATA_TAB;
CREATE COLUMN TABLE META_DATA_TAB(
    "VARIABLE_NAME" NVARCHAR(50),
    "VARIABLE_TYPE" NVARCHAR(50)
);
INSERT INTO META_DATA_TAB VALUES ('X1',    'CONTINUOUS');
INSERT INTO META_DATA_TAB VALUES ('Y' ,    'TARGET');
INSERT INTO META_DATA_TAB VALUES ('X2',    'CATEGORICAL');
INSERT INTO META_DATA_TAB VALUES ('X3',    'CONTINUOUS');
DROP TABLE PAL_DATA_TBL;
CREATE COLUMN TABLE PAL_DATA_TBL(
    "ID" INT,
    "VARIABLE_NAME" NVARCHAR(50),
    "VALUE" NVARCHAR(50),
    "PURPOSE" INT
);
INSERT INTO PAL_DATA_TBL VALUES (0,    'Y',    '-6.879',   1);
INSERT INTO PAL_DATA_TBL VALUES (0,    'X1',   '0.00',     1);
INSERT INTO PAL_DATA_TBL VALUES (0,    'X2',   'A',        1);
INSERT INTO PAL_DATA_TBL VALUES (0,    'X3',   '1',        1);
INSERT INTO PAL_DATA_TBL VALUES (1,    'Y',    '-3.449',   1);
INSERT INTO PAL_DATA_TBL VALUES (1,    'X1',   '0.50',     1);
INSERT INTO PAL_DATA_TBL VALUES (1,    'X2',   'A',        1);
INSERT INTO PAL_DATA_TBL VALUES (1,    'X3',   '1',        1);
INSERT INTO PAL_DATA_TBL VALUES (2,    'Y',    '6.635',    1);
INSERT INTO PAL_DATA_TBL VALUES (2,    'X1',   '0.54',     1);
INSERT INTO PAL_DATA_TBL VALUES (2,    'X2',   'B',        1);
INSERT INTO PAL_DATA_TBL VALUES (2,    'X3',   '1',        1);
INSERT INTO PAL_DATA_TBL VALUES (3,    'Y',    '11.844',   2);
INSERT INTO PAL_DATA_TBL VALUES (3,    'X1',   '1.04',     2);
INSERT INTO PAL_DATA_TBL VALUES (3,    'X2',   'B',        2);
INSERT INTO PAL_DATA_TBL VALUES (3,    'X3',   '1',        2);
INSERT INTO PAL_DATA_TBL VALUES (4,    'Y',    '2.786',    1);
INSERT INTO PAL_DATA_TBL VALUES (4,    'X1',   '1.50',     1);
INSERT INTO PAL_DATA_TBL VALUES (4,    'X2',   'A',        1);
INSERT INTO PAL_DATA_TBL VALUES (4,    'X3',   '1',        1);
INSERT INTO PAL_DATA_TBL VALUES (5,    'Y',    '2.389',    1);
INSERT INTO PAL_DATA_TBL VALUES (5,    'X1',   '0.04',     1);
INSERT INTO PAL_DATA_TBL VALUES (5,    'X2',   'B',        1);
INSERT INTO PAL_DATA_TBL VALUES (5,    'X3',   '2',        1);
INSERT INTO PAL_DATA_TBL VALUES (6,    'Y',    '-0.011',   2);
INSERT INTO PAL_DATA_TBL VALUES (6,    'X1',   '2.00',     2);
INSERT INTO PAL_DATA_TBL VALUES (6,    'X2',   'A',        2);
INSERT INTO PAL_DATA_TBL VALUES (6,    'X3',   '2',        2);
INSERT INTO PAL_DATA_TBL VALUES (7,    'Y',    '8.839',    1);
INSERT INTO PAL_DATA_TBL VALUES (7,    'X1',   '2.04',     1);
INSERT INTO PAL_DATA_TBL VALUES (7,    'X2',   'B',        1);
```

```
INSERT INTO PAL_DATA_TBL VALUES (7,     'X3',    '2',         1);
INSERT INTO PAL_DATA_TBL VALUES (8,     'Y',     '4.689',     1);
INSERT INTO PAL_DATA_TBL VALUES (8,     'X1',    '1.54',      1);
INSERT INTO PAL_DATA_TBL VALUES (8,     'X2',    'B',         1);
INSERT INTO PAL_DATA_TBL VALUES (8,     'X3',    '1',         1);
INSERT INTO PAL_DATA_TBL VALUES (9,     'Y',     '-5.507',    1);
INSERT INTO PAL_DATA_TBL VALUES (9,     'X1',    '1.00',      1);
INSERT INTO PAL_DATA_TBL VALUES (9,     'X2',    'A',         1);
INSERT INTO PAL_DATA_TBL VALUES (9,     'X3',    '2',         1);
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO PAL_PARAMETER_TBL VALUES ('FUNCTION',null,null,'MLR');
INSERT INTO PAL_PARAMETER_TBL VALUES ('BACKGROUND_SIZE', 4,null,null);
DROP TABLE PAL_DATA_MODEL_TBL;
CREATE COLUMN TABLE PAL_DATA_MODEL_TBL(
     "ROW_INDEX" INTEGER,
     "PART_INDEX" INTEGER,
     "MODEL_CONTENT" VARCHAR (5000)
);
DO BEGIN
  meta_data_tab = SELECT * FROM META_DATA_TAB;
  data_tab = SELECT * FROM PAL_DATA_TBL;
  control_tab = SELECT * FROM PAL_PARAMETER_TBL;
  CALL
_SYS_AFL.PAL_UNIFIED_REGRESSION_PIVOT(:meta_data_tab, :data_tab, :control_tab,
model_tab, stats_tab, optimal_parameter_tab, partition_result,
place_holder_1_tab, place_holder_2_tab);
  INSERT into PAL_DATA_MODEL_TBL SELECT * FROM :model_tab;
END
```

### Expected Result

Model:

| | ROW_INDEX | PART_INDEX | MODEL_CONTENT |
|---|---|---|---|
| 1 | 1 | ? | <PMML version="4.0" xmlns="http://www.dmg.org/PMML-4_0" xmlns:xsi="http://www.... |
| 2 | -10 | ? | {"cate_vals":[{"cateVal":["B","B","A"]}],"categorical_size":1,"cont_vals":[{"contVal":[0.540000... |
| 3 | -1 | ? | MLR |
| 4 | -2 | ? | {"feature":["X1","X2","X3","Y"],"featureNum":4,"index":[0,1,2,3],"type":["CONTINUOUS","C... |

### Example 3: Based kernel SHAP(SVM)

```
SET SCHEMA DM_PAL;
DROP TABLE META_DATA_TAB;
CREATE COLUMN TABLE META_DATA_TAB(
    "VARIABLE_NAME" NVARCHAR(50),
    "VARIABLE_TYPE" NVARCHAR(50)
);
INSERT INTO META_DATA_TAB VALUES ('ATT1',     'CONTINUOUS');
INSERT INTO META_DATA_TAB VALUES ('LABEL',    'TARGET');
INSERT INTO META_DATA_TAB VALUES ('ATT2',     'CONTINUOUS');
INSERT INTO META_DATA_TAB VALUES ('ATT3',     'CONTINUOUS');
INSERT INTO META_DATA_TAB VALUES ('ATT4',     'CATEGORICAL');
DROP TABLE PAL_DATA_TBL;
CREATE COLUMN TABLE PAL_DATA_TBL(
    "ID" INT,
    "VARIABLE_NAME" NVARCHAR(50),
    "VALUE" NVARCHAR(50),
    "PURPOSE" INT
);
```

```
INSERT INTO PAL_DATA_TBL VALUES (0,      'ATT1',      '1',         1);
INSERT INTO PAL_DATA_TBL VALUES (0,      'ATT2',      '10',        1);
INSERT INTO PAL_DATA_TBL VALUES (0,      'ATT3',      '100',       1);
INSERT INTO PAL_DATA_TBL VALUES (0,      'ATT4',      'A',         1);
INSERT INTO PAL_DATA_TBL VALUES (0,      'LABEL',     '1',         1);
INSERT INTO PAL_DATA_TBL VALUES (1,      'ATT1',      '1.1',       1);
INSERT INTO PAL_DATA_TBL VALUES (1,      'ATT2',      '10.1',      1);
INSERT INTO PAL_DATA_TBL VALUES (1,      'ATT3',      '100',       1);
INSERT INTO PAL_DATA_TBL VALUES (1,      'ATT4',      'A',         1);
INSERT INTO PAL_DATA_TBL VALUES (1,      'LABEL',     '1',         1);
INSERT INTO PAL_DATA_TBL VALUES (2,      'ATT1',      '1.2',         1);
INSERT INTO PAL_DATA_TBL VALUES (2,      'ATT2',      '10.2',         1);
INSERT INTO PAL_DATA_TBL VALUES (2,      'ATT3',      '100',         1);
INSERT INTO PAL_DATA_TBL VALUES (2,      'ATT4',      'A',         1);
INSERT INTO PAL_DATA_TBL VALUES (2,      'LABEL',     '1',         1);
INSERT INTO PAL_DATA_TBL VALUES (3,      'ATT1',      '1.3',       1);
INSERT INTO PAL_DATA_TBL VALUES (3,      'ATT2',      '10.4',      1);
INSERT INTO PAL_DATA_TBL VALUES (3,      'ATT3',      '100',       1);
INSERT INTO PAL_DATA_TBL VALUES (3,      'ATT4',      'A',         1);
INSERT INTO PAL_DATA_TBL VALUES (3,      'LABEL',     '1',         1);
INSERT INTO PAL_DATA_TBL VALUES (4,      'ATT1',      '1.2',       2);
INSERT INTO PAL_DATA_TBL VALUES (4,      'ATT2',      '10.3',      2);
INSERT INTO PAL_DATA_TBL VALUES (4,      'ATT3',      '100',       2);
INSERT INTO PAL_DATA_TBL VALUES (4,      'ATT4',      'AB',        2);
INSERT INTO PAL_DATA_TBL VALUES (4,      'LABEL',     '1',         2);
INSERT INTO PAL_DATA_TBL VALUES (5,      'ATT1',      '4',         1);
INSERT INTO PAL_DATA_TBL VALUES (5,      'ATT2',      '40',        1);
INSERT INTO PAL_DATA_TBL VALUES (5,      'ATT3',      '400',       1);
INSERT INTO PAL_DATA_TBL VALUES (5,      'ATT4',      'AB',        1);
INSERT INTO PAL_DATA_TBL VALUES (5,      'LABEL',     '1',         1);
INSERT INTO PAL_DATA_TBL VALUES (6,      'ATT1',      '4.1',       1);
INSERT INTO PAL_DATA_TBL VALUES (6,      'ATT2',      '40.1',      1);
INSERT INTO PAL_DATA_TBL VALUES (6,      'ATT3',      '400',       1);
INSERT INTO PAL_DATA_TBL VALUES (6,      'ATT4',      'AB',        1);
INSERT INTO PAL_DATA_TBL VALUES (6,      'LABEL',     '1',         1);
INSERT INTO PAL_DATA_TBL VALUES (7,      'ATT1',      '4.2',       2);
INSERT INTO PAL_DATA_TBL VALUES (7,      'ATT2',      '40.2',      2);
INSERT INTO PAL_DATA_TBL VALUES (7,      'ATT3',      '400',       2);
INSERT INTO PAL_DATA_TBL VALUES (7,      'ATT4',      'AB',        2);
INSERT INTO PAL_DATA_TBL VALUES (7,      'LABEL',     '1',         2);
INSERT INTO PAL_DATA_TBL VALUES (8,      'ATT1',      '4.3',       2);
INSERT INTO PAL_DATA_TBL VALUES (8,      'ATT2',      '40.4',      2);
INSERT INTO PAL_DATA_TBL VALUES (8,      'ATT3',      '400',       2);
INSERT INTO PAL_DATA_TBL VALUES (8,      'ATT4',      'AB',        2);
INSERT INTO PAL_DATA_TBL VALUES (8,      'LABEL',     '2',         2);
INSERT INTO PAL_DATA_TBL VALUES (9,      'ATT1',      '4.2',       2);
INSERT INTO PAL_DATA_TBL VALUES (9,      'ATT2',      '40.3',      2);
INSERT INTO PAL_DATA_TBL VALUES (9,      'ATT3',      '400',       2);
INSERT INTO PAL_DATA_TBL VALUES (9,      'ATT4',      'AB',        2);
INSERT INTO PAL_DATA_TBL VALUES (9,      'LABEL',     '2',         2);
INSERT INTO PAL_DATA_TBL VALUES (10,     'ATT1',      '9',         1);
INSERT INTO PAL_DATA_TBL VALUES (10,     'ATT2',      '90',        1);
INSERT INTO PAL_DATA_TBL VALUES (10,     'ATT3',      '900',       1);
INSERT INTO PAL_DATA_TBL VALUES (10,     'ATT4',      'B',         1);
INSERT INTO PAL_DATA_TBL VALUES (10,     'LABEL',     '2',         1);
INSERT INTO PAL_DATA_TBL VALUES (11,     'ATT1',      '9.1',       1);
INSERT INTO PAL_DATA_TBL VALUES (11,     'ATT2',      '90.1',      1);
INSERT INTO PAL_DATA_TBL VALUES (11,     'ATT3',      '900',       1);
INSERT INTO PAL_DATA_TBL VALUES (11,     'ATT4',      'A',         1);
INSERT INTO PAL_DATA_TBL VALUES (11,     'LABEL',     '2',         1);
INSERT INTO PAL_DATA_TBL VALUES (12,     'ATT1',      '9.2',       1);
INSERT INTO PAL_DATA_TBL VALUES (12,     'ATT2',      '90.2',      1);
INSERT INTO PAL_DATA_TBL VALUES (12,     'ATT3',      '900',       1);
INSERT INTO PAL_DATA_TBL VALUES (12,     'ATT4',      'B',         1);
INSERT INTO PAL_DATA_TBL VALUES (12,     'LABEL',     '2',         1);
INSERT INTO PAL_DATA_TBL VALUES (13,     'ATT1',      '9.3',       1);
INSERT INTO PAL_DATA_TBL VALUES (13,     'ATT2',      '90.4',      1);
INSERT INTO PAL_DATA_TBL VALUES (13,     'ATT3',      '900',       1);
```

```
INSERT INTO PAL_DATA_TBL VALUES (13,    'ATT4',     'A',        1);
INSERT INTO PAL_DATA_TBL VALUES (13,    'LABEL',    '2',        1);
INSERT INTO PAL_DATA_TBL VALUES (14,    'ATT1',     '9.2',      1);
INSERT INTO PAL_DATA_TBL VALUES (14,    'ATT2',     '90.3',     1);
INSERT INTO PAL_DATA_TBL VALUES (14,    'ATT3',     '900',      1);
INSERT INTO PAL_DATA_TBL VALUES (14,    'ATT4',     'A',        1);
INSERT INTO PAL_DATA_TBL VALUES (14,    'LABEL',    '2',        1);
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO PAL_PARAMETER_TBL VALUES ('FUNCTION',null,null,'SVM');
INSERT INTO PAL_PARAMETER_TBL VALUES ('BACKGROUND_SIZE', 4,null,null);
INSERT INTO PAL_PARAMETER_TBL VALUES ('SCALE_INFO', 2,null,null);
DROP TABLE PAL_DATA_MODEL_TBL;
CREATE COLUMN TABLE PAL_DATA_MODEL_TBL(
     "ROW_INDEX" INTEGER,
     "PART_INDEX" INTEGER,
     "MODEL_CONTENT" VARCHAR (5000)
);
DO BEGIN
  meta_data_tab = SELECT * FROM META_DATA_TAB;
  data_tab = SELECT * FROM PAL_DATA_TBL;
  control_tab = SELECT * FROM PAL_PARAMETER_TBL;
  CALL
_SYS_AFL.PAL_UNIFIED_REGRESSION_PIVOT(:meta_data_tab, :data_tab, :control_tab,
model_tab, stats_tab, optimal_parameter_tab, partition_result,
place_holder_1_tab, place_holder_2_tab);
  INSERT into PAL_DATA_MODEL_TBL SELECT * FROM :model_tab;
END
```

### Expected Result

Model:

| | ROW_INDEX | PART_INDEX | MODEL_CONTENT |
|---|---|---|---|
| 1 | 0 | ? | {"C":100.0,"CONTENT":"{\"impute_model\":{\"column_statistics\":[{\"COLUMN_NAME\":\... |
| 2 | -10 | ? | {"cate_vals":[{"cateVal":["A","A"]}],"categorical_size":1,"cont_vals":[{"contVal":[1.2,1.100000... |
| 3 | -1 | ? | SVM |
| 4 | -2 | ? | {"feature":["ATT1","ATT2","ATT3","ATT4","LABEL"],"featureNum":5,"index":[0,1,2,3,4],"type":... |

## _SYS_AFL.PAL_UNIFIED_REGRESSION_PREDICT

### Parameter Table

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| TOP_K_ATTRIBUTIONS | INTEGER | 10 | Outputs the attributions of top k features which contribute the most. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| FEATURE_ATTRIBUTION_METHOD | INTEGER | 2 | Controls which method to use in tree model reasoning:<br>• 0: no reasoning;<br>• 1: Saabas;<br>• 2: tree SHAP. | Valid for the functions:<br>• DT<br>• RDT<br>• HGBT |
| SAMPLESIZE | INTEGER | 0 | Number of sampled combinations of features. If it equals to 0, then it will be determined by algorithm heuristically. | It is better to use a number which is larger than the number of the columns of the data table.<br>Available for the following functions:<br>• EXP<br>• GLM<br>• MLR<br>• MLP<br>• SVM |
| SEED | INTEGER | 0 | Specifies the seed for random number generator:<br>• 0: Uses the current time(in second) as seed;<br>• Others: Uses the specified value as seed. | Valid for the functions:<br>• EXP<br>• GLM<br>• MLR<br>• MLP<br>• SVM |
| IGNORE_CORRELATION | INTEGER | 0 | Specified if ignore the correlation between the features or not:<br>• 0: Ignore the correlation;<br>• 1: Not Ignore the correlation. | Valid for the functions:<br>• EXP<br>• GLM<br>• MLR |

### Examples

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

### Example 1: tree SHAP(DT)

```
DROP TABLE PAL_DATA_TBL_PRE;
CREATE COLUMN TABLE PAL_DATA_TBL_PRE(
    "ID" INT,
```

```
    "VARIABLE_NAME" NVARCHAR(50),
    "VALUE" NVARCHAR(50)
);
INSERT INTO PAL_DATA_TBL_PRE VALUES (0,     'OUTLOOK',      'Overcast');
INSERT INTO PAL_DATA_TBL_PRE VALUES (0,     'TEMP',         '75');
INSERT INTO PAL_DATA_TBL_PRE VALUES (0,     'HUMIDITY',     '70');
INSERT INTO PAL_DATA_TBL_PRE VALUES (0,     'WINDY',        'Yes');
INSERT INTO PAL_DATA_TBL_PRE VALUES (1,     'OUTLOOK',      'Rain');
INSERT INTO PAL_DATA_TBL_PRE VALUES (1,     'TEMP',         '78');
INSERT INTO PAL_DATA_TBL_PRE VALUES (1,     'HUMIDITY',     '70');
INSERT INTO PAL_DATA_TBL_PRE VALUES (1,     'WINDY',        'Yes');
INSERT INTO PAL_DATA_TBL_PRE VALUES (2,     'OUTLOOK',      'Sunny');
INSERT INTO PAL_DATA_TBL_PRE VALUES (2,     'TEMP',         '66');
INSERT INTO PAL_DATA_TBL_PRE VALUES (2,     'HUMIDITY',     '70');
INSERT INTO PAL_DATA_TBL_PRE VALUES (2,     'WINDY',        'Yes');
INSERT INTO PAL_DATA_TBL_PRE VALUES (3,     'OUTLOOK',      'Sunny');
INSERT INTO PAL_DATA_TBL_PRE VALUES (3,     'TEMP',         '69');
INSERT INTO PAL_DATA_TBL_PRE VALUES (3,     'HUMIDITY',     '70');
INSERT INTO PAL_DATA_TBL_PRE VALUES (3,     'WINDY',        'Yes');
INSERT INTO PAL_DATA_TBL_PRE VALUES (4,     'OUTLOOK',      'Rain');
INSERT INTO PAL_DATA_TBL_PRE VALUES (4,     'TEMP',         'null');
INSERT INTO PAL_DATA_TBL_PRE VALUES (4,     'HUMIDITY',     '70');
INSERT INTO PAL_DATA_TBL_PRE VALUES (4,     'WINDY',        'Yes');
INSERT INTO PAL_DATA_TBL_PRE VALUES (5,     'OUTLOOK',      'null');
INSERT INTO PAL_DATA_TBL_PRE VALUES (5,     'TEMP',         '70');
INSERT INTO PAL_DATA_TBL_PRE VALUES (5,     'HUMIDITY',     '70');
INSERT INTO PAL_DATA_TBL_PRE VALUES (5,     'WINDY',        'Yes');
INSERT INTO PAL_DATA_TBL_PRE VALUES (6,     'OUTLOOK',      '***');
INSERT INTO PAL_DATA_TBL_PRE VALUES (6,     'TEMP',         '70');
INSERT INTO PAL_DATA_TBL_PRE VALUES (6,     'HUMIDITY',     '70');
INSERT INTO PAL_DATA_TBL_PRE VALUES (6,     'WINDY',        'Yes');
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
CALL _SYS_AFL.PAL_UNIFIED_REGRESSION_PREDICT_PIVOT(PAL_DATA_TBL_PRE,
PAL_DATA_MODEL_TBL, #PAL_PARAMETER_TBL, ?,?);
```

### Expected Result

Result:

| | ID | SCORE | UPPER_BOUND | LOWER_BOUND | REASON |
|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | ? | [{"attr":"OUTLOOK","pct":41,"val":0.2071428571428571 6},{"attr":"HUMIDITY","pct":34,"val":0.1678571428571429 3},{"attr":"TEMP","pct":25,"val":0.1250000000000003}] |
| 2 | 1 | 0 | 0 | ? | [{"attr":"OUTLOOK","pct":63,"val":-0.4714285714285714 2},{"attr":"HUMIDITY","pct":20,"val":-0.1535714285714286 1},{"attr":"TEMP","pct":17,"val":0.1250000000000003}] |
| 3 | 2 | 1 | 0 | ? | [{"attr":"HUMIDITY","pct":85,"val":0.4357142857142857 8},{"attr":"TEMP","pct":14,"val":0.0714285714285714 25},{"attr":"OUTLOOK","pct":1,"val":-0.0071428571428571452}] |
| 4 | 3 | 1 | 0 | ? | [{"attr":"HUMIDITY","pct":79,"val":0.5071428571428572 3},{"attr":"TEMP","pct":11,"val":-0.0714285714285714 25},{"attr":"OUTLOOK","pct":10,"val":0.0642857142857142 79}] |
| 5 | 4 | 0 | 0 | ? | [{"attr":"OUTLOOK","pct":66,"val":-0.5071428571428571 2},{"attr":"HUMIDITY","pct":17,"val":0.1321428571428571 5},{"attr":"TEMP","pct":17,"val":-0.125}] |
| 6 | 5 | 0.8 | 0 | ? | [{"attr":"HUMIDITY","pct":100,"val":0.3000002142857142 2}] |
| 7 | 6 | 0.8 | 0 | ? | [{"attr":"HUMIDITY","pct":100,"val":0.3000002142857142 2}] |

### Example 2: linear SHAP

```
DROP TABLE PAL_DATA_TBL_PRE;
CREATE COLUMN TABLE PAL_DATA_TBL_PRE(
    "ID" INT,
    "VARIABLE_NAME" NVARCHAR(50),
    "VALUE" NVARCHAR(50)
);
INSERT INTO PAL_DATA_TBL_PRE VALUES (0,    'X1',    '1.69');
INSERT INTO PAL_DATA_TBL_PRE VALUES (0,    'X2',    'B');
INSERT INTO PAL_DATA_TBL_PRE VALUES (0,    'X3',    '1');
INSERT INTO PAL_DATA_TBL_PRE VALUES (1,    'X1',    '0.054');
INSERT INTO PAL_DATA_TBL_PRE VALUES (1,    'X2',    'B');
INSERT INTO PAL_DATA_TBL_PRE VALUES (1,    'X3',    '2');
INSERT INTO PAL_DATA_TBL_PRE VALUES (2,    'X1',    '0.123');
```

```
INSERT INTO PAL_DATA_TBL_PRE VALUES (2,     'X2',     'A');
INSERT INTO PAL_DATA_TBL_PRE VALUES (2,     'X3',     '2');
INSERT INTO PAL_DATA_TBL_PRE VALUES (3,     'X1',     '1.980');
INSERT INTO PAL_DATA_TBL_PRE VALUES (3,     'X2',     'A');
INSERT INTO PAL_DATA_TBL_PRE VALUES (3,     'X3',     '1');
INSERT INTO PAL_DATA_TBL_PRE VALUES (4,     'X1',     '0.563');
INSERT INTO PAL_DATA_TBL_PRE VALUES (4,     'X2',     'A');
INSERT INTO PAL_DATA_TBL_PRE VALUES (4,     'X3',     '1');
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
CALL _SYS_AFL.PAL_UNIFIED_REGRESSION_PREDICT_PIVOT(PAL_DATA_TBL_PRE,
PAL_DATA_MODEL_TBL, #PAL_PARAMETER_TBL, ?,?);
```

### Expected Result

Result:

| | ID | SCORE | UPPER_BOUND | LOWER_BOUND | REASON |
|---|---|---|---|---|---|
| 1 | 0 | 10.314766000000002 | ? | ? | [{"attr":"X2","pct":47.700000000000003,"val":-54.456376465755298},{"attr":"X3","pct":30.800... |
| 2 | 1 | 1.6859316000000018 | ? | ? | [{"attr":"X2","pct":49.100000000000001,"val":98.048973525654958},{"attr":"X3","pct":30.0,"va... |
| 3 | 2 | -7.4095508 | ? | ? | [{"attr":"X2","pct":46.600000000000001,"val":86.192016386746303},{"attr":"X3","pct":31.5,"va... |
| 4 | 3 | 2.0216020000000006 | ? | ? | [{"attr":"X2","pct":51.0,"val":-76.964351117636966},{"attr":"X3","pct":28.899999999999999,"v... |
| 5 | 4 | -3.1226748 | ? | ? | [{"attr":"X2","pct":95.5,"val":-8.672532946221736},{"attr":"X3","pct":2.8999999999999999,"val... |

### Example 3: kernel SHAP

```
DROP TABLE PAL_DATA_TBL_PRE;
CREATE COLUMN TABLE PAL_DATA_TBL_PRE(
    "ID" INT,
    "VARIABLE_NAME" NVARCHAR(50),
    "VALUE" NVARCHAR(50)
);
INSERT INTO PAL_DATA_TBL_PRE VALUES (0,      'ATT1',      '1');
INSERT INTO PAL_DATA_TBL_PRE VALUES (0,      'ATT2',      '10');
INSERT INTO PAL_DATA_TBL_PRE VALUES (0,      'ATT3',      '100');
INSERT INTO PAL_DATA_TBL_PRE VALUES (0,      'ATT4',      'A');
INSERT INTO PAL_DATA_TBL_PRE VALUES (1,      'ATT1',      '1.2');
INSERT INTO PAL_DATA_TBL_PRE VALUES (1,      'ATT2',      '10.2');
INSERT INTO PAL_DATA_TBL_PRE VALUES (1,      'ATT3',      '100');
INSERT INTO PAL_DATA_TBL_PRE VALUES (1,      'ATT4',      'A');
INSERT INTO PAL_DATA_TBL_PRE VALUES (2,      'ATT1',      '4.1');
INSERT INTO PAL_DATA_TBL_PRE VALUES (2,      'ATT2',      '40.1');
INSERT INTO PAL_DATA_TBL_PRE VALUES (2,      'ATT3',      '400');
INSERT INTO PAL_DATA_TBL_PRE VALUES (2,      'ATT4',      'AB');
INSERT INTO PAL_DATA_TBL_PRE VALUES (3,      'ATT1',      '4.2');
INSERT INTO PAL_DATA_TBL_PRE VALUES (3,      'ATT2',      '40.3');
INSERT INTO PAL_DATA_TBL_PRE VALUES (3,      'ATT3',      '400');
INSERT INTO PAL_DATA_TBL_PRE VALUES (3,      'ATT4',      'AB');
INSERT INTO PAL_DATA_TBL_PRE VALUES (4,      'ATT1',      '9.1');
INSERT INTO PAL_DATA_TBL_PRE VALUES (4,      'ATT2',      '90.1');
INSERT INTO PAL_DATA_TBL_PRE VALUES (4,      'ATT3',      '900');
INSERT INTO PAL_DATA_TBL_PRE VALUES (4,      'ATT4',      'A');
INSERT INTO PAL_DATA_TBL_PRE VALUES (5,      'ATT1',      '9.2');
INSERT INTO PAL_DATA_TBL_PRE VALUES (5,      'ATT2',      '90.2');
INSERT INTO PAL_DATA_TBL_PRE VALUES (5,      'ATT3',      '900');
INSERT INTO PAL_DATA_TBL_PRE VALUES (5,      'ATT4',      'A');
INSERT INTO PAL_DATA_TBL_PRE VALUES (6,      'ATT1',      '4');
INSERT INTO PAL_DATA_TBL_PRE VALUES (6,      'ATT2',      '40');
INSERT INTO PAL_DATA_TBL_PRE VALUES (6,      'ATT3',      '400');
INSERT INTO PAL_DATA_TBL_PRE VALUES (6,      'ATT4',      'A');
DROP TABLE #PAL_PARAMETER_TBL;
```

```
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
CALL _SYS_AFL.PAL_UNIFIED_REGRESSION_PREDICT_PIVOT(PAL_DATA_TBL_PRE,
PAL_DATA_MODEL_TBL, #PAL_PARAMETER_TBL, ?,?);
```

### Expected Result

Result:

| | ID | SCORE | UPPER_BOUND | LOWER_BOUND | REASON |
|---|---|---|---|---|---|
| 1 | 0 | 0.980143 | ? | ? | [{"attr":"ATT1","pct":94.799999999999997,"val":-0.023070936477644466},{"attr":"ATT2","pct":... |
| 2 | 1 | 1.00507 | ? | ? | [{"attr":"ATT1","pct":94.799999999999997,"val":0.0033421746516761885},{"attr":"ATT2","pct":... |
| 3 | 2 | 1.1328 | ? | ? | [{"attr":"ATT1","pct":51.700000000000003,"val":0.62405282899661552},{"attr":"ATT3","pct":27... |
| 4 | 3 | 1.1645 | ? | ? | [{"attr":"ATT1","pct":52.700000000000003,"val":0.64860160926991406},{"attr":"ATT3","pct":26... |
| 5 | 4 | 2.03555 | ? | ? | [{"attr":"ATT1","pct":79.199999999999989,"val":1.2925079895727964},{"attr":"ATT3","pct":19.... |
| 6 | 5 | 2.04525 | ? | ? | [{"attr":"ATT1","pct":79.399999999999991,"val":1.3024597341997233},{"attr":"ATT3","pct":19.... |
| 7 | 6 | 1.17406 | ? | ? | [{"attr":"ATT1","pct":53.5,"val":0.52977141543495121},{"attr":"ATT3","pct":32.7000000000000... |

## _SYS_AFL.PAL_UNIFIED_REGRESSION_SCORE

### Parameter Table

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| TOP_K_ATTRIBUTIONS | INTEGER | 10 | Outputs the attributions of top k features which contribute the most. | |
| FEATURE_ATTRIBUTION_METHOD | INTEGER | 2 | Controls which method to use in tree model reasoning:<br>• 0: no reasoning;<br>• 1: Saabas;<br>• 2: tree SHAP. | Valid for the functions:<br>• DT<br>• RDT<br>• HGBT |
| SAMPLESIZE | INTEGER | 0 | Number of sampled combinations of features. If it equals to 0, then it will be determined by algorithm heuristically. | It is better to use a number which is larger than the number of the columns of the data table.<br>Available for the following functions:<br>• EXP<br>• GLM<br>• MLR<br>• MLP<br>• SVM |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| SEED | INTEGER | 0 | Specifies the seed for random number generator:<br>• 0: Uses the current time(in second) as seed;<br>• Others: Uses the specified value as seed. | Valid for the functions:<br>• EXP<br>• GLM<br>• MLR<br>• MLP<br>• SVM |
| IGNORE_CORRELATION | INTEGER | 0 | Specified if ignore the correlation between the features or not:<br>• 0: Ignore the correlation;<br>• 1: Not ignore the correlation. | Valid for the functions:<br>• EXP<br>• GLM<br>• MLR |

Examples

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

**Example 1: tree SHAP(DT)**

```
DROP TABLE PAL_DATA_TBL_SCORE;
CREATE COLUMN TABLE PAL_DATA_TBL_SCORE(
    "ID" INT,
    "VARIABLE_NAME" NVARCHAR(50),
    "VALUE" NVARCHAR(50)
);
INSERT INTO PAL_DATA_TBL_SCORE VALUES (0,     'OUTLOOK',      'Overcast');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (0,     'TEMP',         '75');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (0,     'HUMIDITY',     '70');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (0,     'WINDY',        'Yes');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (0,     'CLASS',        '1');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (1,     'OUTLOOK',      'Rain');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (1,     'TEMP',         '78');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (1,     'HUMIDITY',     '70');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (1,     'WINDY',        'Yes');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (1,     'CLASS',        '1');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (2,     'OUTLOOK',      'Sunny');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (2,     'TEMP',         '66');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (2,     'HUMIDITY',     '70');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (2,     'WINDY',        'Yes');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (2,     'CLASS',        '1');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (3,     'OUTLOOK',      'Sunny');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (3,     'TEMP',         '69');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (3,     'HUMIDITY',     '70');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (3,     'WINDY',        'Yes');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (3,     'CLASS',        '1');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (4,     'OUTLOOK',      'Rain');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (4,     'TEMP',         'null');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (4,     'HUMIDITY',     '70');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (4,     'WINDY',        'Yes');
```

```
INSERT INTO PAL_DATA_TBL_SCORE VALUES (4,     'CLASS',        '0');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (5,     'OUTLOOK',      'null');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (5,     'TEMP',         '70');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (5,     'HUMIDITY',     '70');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (5,     'WINDY',        'Yes');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (5,     'CLASS',        '0.3');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (6,     'OUTLOOK',      '***');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (6,     'TEMP',         '70');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (6,     'HUMIDITY',     '70');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (6,     'WINDY',        'Yes');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (6,     'CLASS',        '0.5');
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FEATURE_ATTRIBUTION_METHOD',1,null,null);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('TOP_K_ATTRIBUTIONS',2,null,null);
CALL _SYS_AFL.PAL_UNIFIED_REGRESSION_SCORE_PIVOT(PAL_DATA_TBL_SCORE,
PAL_DATA_MODEL_TBL, #PAL_PARAMETER_TBL, ?,?);
```

### Expected Result

Result:

| | ID | SCORE | UPPER_BOUND | LOWER_BOUND | REASON |
|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | ? | [{"attr":"HUMIDITY","pct":60,"val":0.30000000000000004},{"attr":"OUTLOOK","pct":40,"val":0.19999999999999996}] |
| 2 | 1 | 0 | 0 | ? | [{"attr":"OUTLOOK","pct":73,"val":-0.80000000000000004},{"attr":"HUMIDITY","pct":27,"val":0.30000000000000004}] |
| 3 | 2 | 1 | 0 | ? | [{"attr":"HUMIDITY","pct":60,"val":0.30000000000000004},{"attr":"OUTLOOK","pct":40,"val":0.19999999999999996}] |
| 4 | 3 | 1 | 0 | ? | [{"attr":"HUMIDITY","pct":60,"val":0.30000000000000004},{"attr":"OUTLOOK","pct":40,"val":0.19999999999999996}] |
| 5 | 4 | 0 | 0 | ? | [{"attr":"OUTLOOK","pct":73,"val":-0.80000000000000004},{"attr":"HUMIDITY","pct":27,"val":0.30000000000000004}] |
| 6 | 5 | 0.8 | 0 | ? | [{"attr":"HUMIDITY","pct":100,"val":0.30000000000000004}] |
| 7 | 6 | 0.8 | 0 | ? | [{"attr":"HUMIDITY","pct":100,"val":0.30000000000000004}] |

Statistic:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | EVAR | -0.2724795640326978 |
| 2 | MAE | 0.2571428571428572 |
| 3 | MAPE | 0.46666666666666673 |
| 4 | MAX_ERROR | 1 |
| 5 | MSE | 0.19142857142857145 |
| 6 | R2 | -0.2779291553133516 |
| 7 | RMSE | 0.43752550946038726 |
| 8 | WMAPE | 0.375 |

### Example 2: linear SHAP

```
DROP TABLE PAL_DATA_TBL_SCORE;
CREATE COLUMN TABLE PAL_DATA_TBL_SCORE(
    "ID" INT,
    "VARIABLE_NAME" NVARCHAR(50),
    "VALUE" NVARCHAR(50)
);
INSERT INTO PAL_DATA_TBL_SCORE VALUES (0,    'Y',    '4');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (0,    'X1',    '1.69');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (0,    'X2',    'B');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (0,    'X3',    '1');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (1,    'Y',    '3');
```

```
INSERT INTO PAL_DATA_TBL_SCORE VALUES (1,    'X1',    '0.054');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (1,    'X2',    'B');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (1,    'X3',    '2');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (2,    'Y',    '7');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (2,    'X1',    '0.123');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (2,    'X2',    'A');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (2,    'X3',    '2');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (3,    'Y',    '9');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (3,    'X1',    '1.980');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (3,    'X2',    'A');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (3,    'X3',    '1');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (4,    'Y',    '0.5');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (4,    'X1',    '0.563');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (4,    'X2',    'A');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (4,    'X3',    '1');
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('TOP_K_ATTRIBUTIONS',2,null,null);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SAMPLESIZE',4,null,null);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SEED',1111,null,null);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('IGNORE_CORRELATION',1,null,null);
CALL _SYS_AFL.PAL_UNIFIED_REGRESSION_SCORE_PIVOT(PAL_DATA_TBL_SCORE,
PAL_DATA_MODEL_TBL, #PAL_PARAMETER_TBL, ?,?);
```

### Expected Result

Result:

| | ID | SCORE | UPPER_BOUND | LOWER_BOUND | REASON |
|---|---|---|---|---|---|
| 1 | 0 | 10.314766000000002 | ? | ? | [{"attr":"X2","pct":45.200000000000003,"val":2.336495000000002},{"attr":"X1","pct":28.800000000000001,"val":1.4884639999999996}] |
| 2 | 1 | 1.6859316000000018 | ? | ? | [{"attr":"X1","pct":54.700000000000003,"val":-4.4508703999999994},{"attr":"X2","pct":28.699999999999999,"val":2.3364950000000002}] |
| 3 | 2 | -7.4095508 | ? | ? | [{"attr":"X2","pct":55.799999999999997,"val":-7.0094850000000006},{"attr":"X1","pct":33.5,"val":-4.2003728000000002}] |
| 4 | 3 | 2.0216020000000006 | ? | ? | [{"attr":"X2","pct":64.299999999999997,"val":-7.0094850000000006},{"attr":"X1","pct":23.300000000000001,"val":2.5412799999999995}] |
| 5 | 4 | -3.1226748 | ? | ? | [{"attr":"X2","pct":64.0,"val":-7.0094850000000006},{"attr":"X1","pct":23.699999999999999,"val":-2.6029968000000001}] |

Statistic:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | EVAR | -4.155816591524087 |
| 2 | MAE | 6.527891599999999 |
| 3 | MAPE | 2.4191897425396824 |
| 4 | MAX_ERROR | 14.4095508 |
| 5 | MSE | 62.212002201074846 |
| 6 | R2 | -5.943303817084247 |
| 7 | RMSE | 7.887458538786422 |
| 8 | WMAPE | 1.3889131063829787 |

### Example 3: kernel SHAP

```
DROP TABLE PAL_DATA_TBL_SCORE;
CREATE COLUMN TABLE PAL_DATA_TBL_SCORE(
    "ID" INT,
    "VARIABLE_NAME" NVARCHAR(50),
    "VALUE" NVARCHAR(50)
);
INSERT INTO PAL_DATA_TBL_SCORE VALUES (0,     'ATT1',     '1');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (0,     'ATT2',     '10');
```

```
INSERT INTO PAL_DATA_TBL_SCORE VALUES (0,      'ATT3',      '100');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (0,      'ATT4',      'A');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (0,      'LABEL',     '2');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (1,      'ATT1',      '1.2');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (1,      'ATT2',      '10.2');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (1,      'ATT3',      '100');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (1,      'ATT4',      'A');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (1,      'LABEL',     '3');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (2,      'ATT1',      '4.1');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (2,      'ATT2',      '40.1');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (2,      'ATT3',      '400');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (2,      'ATT4',      'AB');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (2,      'LABEL',     '1');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (3,      'ATT1',      '4.2');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (3,      'ATT2',      '40.3');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (3,      'ATT3',      '400');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (3,      'ATT4',      'AB');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (3,      'LABEL',     '7');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (4,      'ATT1',      '9.1');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (4,      'ATT2',      '90.1');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (4,      'ATT3',      '900');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (4,      'ATT4',      'A');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (4,      'LABEL',     '4');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (5,      'ATT1',      '9.2');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (5,      'ATT2',      '90.2');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (5,      'ATT3',      '900');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (5,      'ATT4',      'A');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (5,      'LABEL',     '3');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (6,      'ATT1',      '4');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (6,      'ATT2',      '40');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (6,      'ATT3',      '400');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (6,      'ATT4',      'A');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (6,      'LABEL',     '2');
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('TOP_K_ATTRIBUTIONS',3,null,null);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SAMPLESIZE',4,null,null);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SEED',1111,null,null);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('IGNORE_CORRELATION',1,null,null);
CALL _SYS_AFL.PAL_UNIFIED_REGRESSION_SCORE_PIVOT(PAL_DATA_TBL_SCORE,
PAL_DATA_MODEL_TBL, #PAL_PARAMETER_TBL, ?,?);
```

### Expected Result

Result:

| | ID | SCORE | UPPER_BOUND | LOWER_BOUND | REASON |
|---|---|---|---|---|---|
| 1 | 0 | 0.980143 | ? | ? | [{"attr":"ATT1","pct":94.799999999999997,"val":-0.023070936477644466},{"attr":"ATT2","pct":5.1999999999999993,"val":0.0012640779399308855}] |
| 2 | 1 | 1.00507 | ? | ? | [{"attr":"ATT1","pct":94.799999999999997,"val":0.0033421746516761885},{"attr":"ATT2","pct":5.1999999999999993,"val":-0.00018286458819538609}] |
| 3 | 2 | 1.1328 | ? | ? | |
| 4 | 3 | 1.1645 | ? | ? | |
| 5 | 4 | 2.03555 | ? | ? | |
| 6 | 5 | 2.04525 | ? | ? | |
| 7 | 6 | 1.17406 | ? | ? | [{"attr":"ATT1","pct":null,"val":-1e+9999},{"attr":"ATT2","pct":null,"val":1e+9999},{"attr":"ATT3","pct":null,"val":-1e+9999}] |

Statistic:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | EVAR | 0.021960645834287584 |
| 2 | MAE | 1.8183181428571427 |
| 3 | MAPE | 0.48052578911564625 |
| 4 | MAX_ERROR | 5.8355 |
| 5 | MSE | 6.363334049135569 |
| 6 | R2 | -0.9487710525477682 |
| 7 | RMSE | 2.5225649742148506 |
| 8 | WMAPE | 0.5785557727272727 |

## 5.13.10.7 Missing Value Handling

Missing values in each column of the table can be handled by one of the following imputation types:

- MODE: Fills all missing values by the mode of the column, that is, the value that appears most often in the column. This type only suits column with categorical variables.
- MEAN: Fills all missing values by the mean of the column, that is, the average of all non-missing values in the column. This type only suits column with numerical variables.
- MEDIAN: Fills all missing values by the median of the column, that is, the value that separates the higher half of the non-missing values from the lower half. If there is an even number of observations, then the median is given by the mean of the two middle values. This type only suits column with numerical variables.
- ALS: Fills each missing value by the value imputed by a matrix completion model trained using alternating least squares method. This type only suits column with numerical variables.
- SPECIFIED_CATEGORICAL_VALUE: Fills all missing values by a specified categorical value. This type only suits column with categorical variables.
- SPECIFIED_NUMERICAL_VALUE: Fills all missing values by a specified numerical value. This type only suits column with numerical variables.
- DELETE: Deletes all rows with missing values. The entire row in table will be deleted.
- NONE: Does nothing. Leave the column untouched.

More detailed information is provided with the guidance of Missing Value Handling algorithm (preprocessing algorithms section).

**Parameter Table**

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| HANDLE_MISSING_VALUE | INTEGER | 0 | Whether to handle missing values:<br>• 0: No<br>• 1: Yes | |
| IMPUTATION_TYPE | INTEGER | 1 | Chooses the overall default imputation type for all columns:<br>• 0: NONE<br>• 1: MODE_MEAN<br>• 2: MODE_MEDIAN<br>• 3: MODE_ALLZERO<br>• 4: MODE_ALS<br>• 5: DELETE<br>NONE and DELETE: set all columns to imputation type NONE/DELETE.<br>MODE_MEAN, MODE_MEDIAN, MODE_ALLZERO, and MODE_ALS: set categorical columns to the first imputation type (the type before “”); set numerical columns to the second imputation type (the type after “”). ALLZERO means to fill all missing values in numerical columns with 0. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| `<column name>`_IMPUTATION_TYPE | INTEGER | No default value | Sets imputation type of a specified column. This value overwrites the default imputation type deduced by the IMPUTATION_TYPE parameter:<br>• 0: NONE<br>• 1: DELETE<br>• 100: MODE<br>• 101: SPECIFIED_CATEGORICAL_VALUE<br>• 200: MEAN<br>• 201: MEDIAN<br>• 203:SPECIFIED_NUMERICAL_VALUE<br>• 204: ALS<br>For type 101 (SPECIFIED_CATEGORICAL_VALUE), use the following syntax to specify a categorical value to be filled in a column V0:<br>INSERT INTO #PAL_PARAMETER_TBL VALUES('V0_IMPUTATION_TYPE', 101, NULL, '`<CATEGORICAL_VALUE>`');<br>For type 203 (SPECIFIED_NUMERICAL_VALUE), use the following syntax to specify a numerical value to be filled in a column V1:<br>INSERT INTO #PAL_PARAMETER_TBL VALUES('V1_IMPUTA- | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| | | | TION_TYPE', 203, `<NUMERICAL_VALUE>`, NULL);<br><br>**Note**<br>A column of integer type can be set to type 101 (SPECIFIED_CATEGORICAL_VALUE) or type 203 (SPECIFIED_NUMERICAL_VALUE):<br>• when setting to type 101 (SPECIFIED_CATEGORICAL_VALUE), the string must contain a valid integer value, otherwise an error will be thrown;<br>• when setting to type 203 (SPECIFIED_NUMERICAL_VALUE), the double value will be converted to integer. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| ALS_FACTOR_NUMBER | INTEGER | 3 | Length of factor vectors, that is, the f in ALS model.<br>**Note**<br>ALS_FACTOR_NUMBER should be less than the number of numerical columns to get a meaningful result. | |
| ALS_REGULARIZATION | DOUBLE | 1e-2 | L2 regularization of the factors, that is, the λ in ALS model. | |
| ALS_MAX_ITERATION | INTEGER | 20 | Maximum number of iterations when training ALS model. | |
| ALS_SEED | INTEGER | 0 | Specifies the seed of the random number generator used in the training of ALS model:<br>• 0: Uses the current time as the seed<br>• Others: Uses the specified value as the seed. | |
| ALS_EXIT_THRESHOLD | DOUBLE | 0.0 | The training of ALS model exits if the value of cost function is not decreased more than the value of ALS_EXIT_THRESHOLD since the last check.<br>If ALS_EXIT_THRESHOLD is set to 0, there is no check and the training only exits on reaching the maximum number of iterations. Evaluations of cost function require additional calculations. You can set this parameter to 0 to avoid it. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| ALS_EXIT_INTERVAL | INTEGER | 5 | During the training of ALS model, it will calculate cost function and check the exit criteria every `<ALS_EXIT_INTERVAL>` iterations. Evaluations of cost function require additional calculations. | |
| ALS_LINEAR_SYSTEM_SOLVER | INTEGER | 0 | • 0: cholesky solver<br>• 1: cg solver (recommended when ALS_FACTOR_NUMBER is large) | |
| ALS_CG_MAX_ITERATION | INTEGER | 3 | Maximum number of iteration of cg solver. | |
| ALS_CENTERING | INTEGER | 1 | Whether to center the data by column before training ALS model.<br>• 0: no centering<br>• 1: centering. | |
| ALS_SCALING | INTEGER | 1 | Whether to scale the data by column before training ALS model.<br>• 0: no scaling<br>• 1: scaling | |

### Examples

Assume that:

- DM_PAL is a schema belonging to USER1;
- USER1 has been assigned the AFL__SYS_AFL_AFLPAL_EXECUTE or AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION role.

#### Example 1: Exponential Regression (Training)

```
SET SCHEMA DM_PAL;
DROP TABLE META_DATA_TAB;
CREATE COLUMN TABLE META_DATA_TAB(
    "VARIABLE_NAME" NVARCHAR(50),
    "VARIABLE_TYPE" NVARCHAR(50)
);
INSERT INTO META_DATA_TAB VALUES ('X1',    'CONTINUOUS');
INSERT INTO META_DATA_TAB VALUES ('Y' ,    'TARGET');
INSERT INTO META_DATA_TAB VALUES ('X2',    'CONTINUOUS');
DROP TABLE PAL_DATA_TBL;
CREATE COLUMN TABLE PAL_DATA_TBL(
    "ID" INT,
    "VARIABLE_NAME" NVARCHAR(50),
```

```
    "VALUE" NVARCHAR(50),
    "PURPOSE" INT
);
INSERT INTO PAL_DATA_TBL VALUES (0,     'Y',    '0.5',      1);
INSERT INTO PAL_DATA_TBL VALUES (0,     'X1',   '0.13',     1);
INSERT INTO PAL_DATA_TBL VALUES (0,     'X2',   '0.33',     1);
INSERT INTO PAL_DATA_TBL VALUES (1,     'Y',    '0.15',     1);
INSERT INTO PAL_DATA_TBL VALUES (1,     'X1',   '0.14',     1);
INSERT INTO PAL_DATA_TBL VALUES (1,     'X2',   '0.34',     1);
INSERT INTO PAL_DATA_TBL VALUES (2,     'Y',    '0.25',     2);
INSERT INTO PAL_DATA_TBL VALUES (2,     'X1',   '0.15',     2);
--INSERT INTO PAL_DATA_TBL VALUES (2,     'X2',   '0.36',     2);
INSERT INTO PAL_DATA_TBL VALUES (3,     'Y',    '0.35',     1);
INSERT INTO PAL_DATA_TBL VALUES (3,     'X1',   '0.16',     1);
INSERT INTO PAL_DATA_TBL VALUES (3,     'X2',   '0.35',     1);
INSERT INTO PAL_DATA_TBL VALUES (4,     'Y',    '0.45',     1);
INSERT INTO PAL_DATA_TBL VALUES (4,     'X1',   '0.17',     1);
INSERT INTO PAL_DATA_TBL VALUES (4,     'X2',   '0.37',     1);
INSERT INTO PAL_DATA_TBL VALUES (5,     'Y',    '0.55',     1);
--INSERT INTO PAL_DATA_TBL VALUES (5,     'X1',   '0.18',     1);
INSERT INTO PAL_DATA_TBL VALUES (5,     'X2',   '0.38',     1);
INSERT INTO PAL_DATA_TBL VALUES (6,     'Y',    '0.65',     2);
INSERT INTO PAL_DATA_TBL VALUES (6,     'X1',   '0.19',     2);
INSERT INTO PAL_DATA_TBL VALUES (6,     'X2',   '0.39',     2);
INSERT INTO PAL_DATA_TBL VALUES (7,     'Y',    '0.75',     2);
INSERT INTO PAL_DATA_TBL VALUES (7,     'X1',   '0.19',     2);
INSERT INTO PAL_DATA_TBL VALUES (7,     'X2',   '0.31',     2);
INSERT INTO PAL_DATA_TBL VALUES (8,     'Y',    '0.85',     1);
INSERT INTO PAL_DATA_TBL VALUES (8,     'X1',   '0.11',     1);
INSERT INTO PAL_DATA_TBL VALUES (8,     'X2',   '0.32',     1);
INSERT INTO PAL_DATA_TBL VALUES (9,     'Y',    '0.95',     1);
INSERT INTO PAL_DATA_TBL VALUES (9,     'X1',   '0.12',     1);
INSERT INTO PAL_DATA_TBL VALUES (9,     'X2',   '0.33',     1);
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO PAL_PARAMETER_TBL VALUES ('FUNCTION',null,null,'EXP');
INSERT INTO PAL_PARAMETER_TBL VALUES ('HANDLE_MISSING_VALUE',1,null,null);
INSERT INTO PAL_PARAMETER_TBL VALUES ('PMML_EXPORT',2,null,null);
INSERT INTO PAL_PARAMETER_TBL VALUES ('PARTITION_METHOD',1,null,null);
INSERT INTO PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', null,0.5,null);
DROP TABLE PAL_DATA_MODEL_TBL;
CREATE COLUMN TABLE PAL_DATA_MODEL_TBL(
     "ROW_INDEX" INTEGER,
     "PART_INDEX" INTEGER,
     "MODEL_CONTENT" VARCHAR (5000)
);
DO BEGIN
  meta_data_tab = SELECT * FROM META_DATA_TAB;
  data_tab = SELECT * FROM PAL_DATA_TBL;
  control_tab = SELECT * FROM PAL_PARAMETER_TBL;
  CALL
_SYS_AFL.PAL_UNIFIED_REGRESSION_PIVOT(:meta_data_tab, :data_tab, :control_tab,
model_tab, fit_tab, stats_tab, optimal_parameter_tab, place_holder_1_tab,
place_holder_2_tab);
  INSERT into PAL_DATA_MODEL_TBL SELECT * FROM :model_tab;
END;
```

### Example 2: Exponential Regression (Prediction)

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_DATA_TBL_PRE;
CREATE COLUMN TABLE PAL_DATA_TBL_PRE(
```

```
    "ID" INT,
    "VARIABLE_NAME" NVARCHAR(50),
    "VALUE" NVARCHAR(50)
);
INSERT INTO PAL_DATA_TBL_PRE VALUES (0,  'X1',  '0.5');
--INSERT INTO PAL_DATA_TBL_PRE VALUES (0,  'X2',  '0.3');
INSERT INTO PAL_DATA_TBL_PRE VALUES (1,  'X1',  '4');
INSERT INTO PAL_DATA_TBL_PRE VALUES (1,  'X2',  '0.4');
INSERT INTO PAL_DATA_TBL_PRE VALUES (2,  'X1',  '0');
--INSERT INTO PAL_DATA_TBL_PRE VALUES (2,  'X2',  '1.6');
INSERT INTO PAL_DATA_TBL_PRE VALUES (3,  'X1',  '0.3');
INSERT INTO PAL_DATA_TBL_PRE VALUES (3,  'X2',  '0.45');
INSERT INTO PAL_DATA_TBL_PRE VALUES (4,  'X1',  '0.4');
INSERT INTO PAL_DATA_TBL_PRE VALUES (4,  'X2',  '1.7');
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('HANDLE_MISSING_VALUE',1,null,null);
CALL _SYS_AFL.PAL_UNIFIED_REGRESSION_PREDICT_PIVOT(PAL_DATA_TBL_PRE,
PAL_DATA_MODEL_TBL, #PAL_PARAMETER_TBL, ?,?);
```

### Example 3: Exponential Regression (Scoring)

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_DATA_TBL_SCORE;
CREATE COLUMN TABLE PAL_DATA_TBL_SCORE(
    "ID" INT,
    "VARIABLE_NAME" NVARCHAR(50),
    "VALUE" NVARCHAR(50)
);
INSERT INTO PAL_DATA_TBL_SCORE VALUES (0,   'Y',    '3');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (0,   'X1',   '0.5');
--INSERT INTO PAL_DATA_TBL_SCORE VALUES (0,   'X2',   '0.3');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (1,   'Y',    '4');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (1,   'X1',   '4');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (1,   'X2',   '0.4');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (2,   'Y',    '7');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (2,   'X1',   '0');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (2,   'X2',   '0.16');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (3,   'Y',    '9');
--INSERT INTO PAL_DATA_TBL_SCORE VALUES (3,   'X1',   '0.3');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (3,   'X2',   '0.45');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (4,   'Y',    '0.5');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (4,   'X1',   '0.4');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (4,   'X2',   '1.7');
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('HANDLE_MISSING_VALUE',1,null,null);
CALL _SYS_AFL.PAL_UNIFIED_REGRESSION_SCORE_PIVOT(PAL_DATA_TBL_SCORE,
PAL_DATA_MODEL_TBL, #PAL_PARAMETER_TBL, ?,?);
```

## 5.13.10.8 Prediction Intervals and Confidence Intervals

PAL provides interval estimates in addition to the point estimate of response value in the unified prediction or scoring regression procedure.

PAL supports two types of interval estimates: confidence intervals and prediction intervals. You can specify the `INTERVAL` parameter to denote the desired type of interval for the response value. Except the interval type, you also need to specify the `SIGNIFICANCE_LEVEL` parameter to represent the long-run proportion of corresponding intervals that fall outside of the true estimated target. The difference between the two types of intervals is tiny and confusing. The prediction interval predicts distribution of individual future observation of target, while the confidence interval predicts the mean statistic of future target population.

Prediction intervals and confidence intervals are provided for Generalised Linear Models (GLM), Multiple Linear Regression (MLR), and Random Decision Trees (RDT) in the prediction and scoring procedure. The following tables list the parameters for using prediction intervals or confidence intervals in GLM, MLR, and RDT respectively.

### Unified Regression Prediction or Scoring with Pivot

**Parameter Table**

Mandatory Parameters

None.

Optional Parameters

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| SIGNIFICANCE_LEVEL | DOUBLE | 0.05 | The significance level for the confidence interval and the prediction interval. Valid range is (0, 1). | |
| INTERVAL | INTEGER | 0 | Specifies the type of interval to output.<br>• 0: No interval is calculated and output;<br>• 1: Confidence Interval;<br>• 2: Prediction Interval. | |

### GLM

Currently confidence intervals and prediction intervals are only output for models obtained via the IRLS method. Since the `INTERVAL` parameter is 0, it doesn't output either interval by default. In order to output an interval, set the value to 1 or 2.

The interval target can be either response or link, which can be specified with the `TYPE` parameter. The intervals are calculated at link scale with approximate Student-t Distribution (for continuous distribution

family) or Normal Distribution (for discrete distribution family), based on the standard error of each provided data instance. If the interval target is response, convert the link interval to the response interval with the inverse of link function.

### Prediction Procedures

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| TYPE | VARCHAR or NVARCHAR | response | The prediction type:<br>• response: Predicts response μ;<br>• link: Predicts linear predictor η. | |
| SIGNIFICANCE_LEVEL | DOUBLE | 0.05 | The significance level for the confidence interval and prediction interval. Valid range is (0, 1). | Valid only for IRLS. |

## MLR

Currently confidence intervals and prediction intervals are only output for models trained with the parameter `OUTPUT_COEFCOV` set to 1 and the model format set to JSON. Therefore, it also requires `JSON_EXPORT` to be set to 1, which is the default. Note that `OUTPUT_COEFCOV` is 0 by default, so the trained model can't be used to calculate intervals during prediction or scoring. In order to output either interval during prediction or scoring, it requires that the model contains coefficient covariances information. Otherwise, the prediction or scoring procedure outputs error messages when outputting either interval.

The interval target is response. Intervals are calculated at response level with Student-t Distribution, based on the standard error for each provided data instance. When computing the standard error, it also requires unbiased MSE statistic in model. However, if the residual degree freedom is 0, there's no definition for it. So, in this case neither interval can be obtained and is output as a NULL value.

### Training Procedures

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| OUTPUT_COEFCOV | INTEGER | 0 | Specifies whether to output coefficient covariance information.<br>• 0: Doesn't output related information;<br>• 1: Outputs related information. | Valid only when JSON_EXPORT=1. |

## RDT

Currently only prediction intervals are available in function RDT. It is output for model trained with parameter `OUTPUT_LEAF_VALUES` set to 1 and model format is JSON (MODEL_FORMAT is 1 or COMPRESSION is 1).

### Training Procedures

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| OUTPUT_LEAF_VALUES | INTEGER | 0 | Controls whether to save target values in each leaf node.<br>• 0: Doesn't output target values in each leaf node;<br>• 1: Outputs target values in each leaf node. | Valid only when JSON_EXPORT = 1 or COMPRESSION = 1. |

## Examples

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA DM_PAL;
DROP TABLE META_DATA_TAB;
CREATE COLUMN TABLE META_DATA_TAB(
    "VARIABLE_NAME" NVARCHAR(50),
    "VARIABLE_TYPE" NVARCHAR(50)
);
INSERT INTO META_DATA_TAB VALUES ('X1',    'CONTINUOUS');
INSERT INTO META_DATA_TAB VALUES ('Y' ,    'TARGET');
INSERT INTO META_DATA_TAB VALUES ('X2',    'CATEGORICAL');
INSERT INTO META_DATA_TAB VALUES ('X3',    'CONTINUOUS');
DROP TABLE PAL_DATA_TBL;
CREATE COLUMN TABLE PAL_DATA_TBL(
    "ID" INT,
    "VARIABLE_NAME" NVARCHAR(50),
    "VALUE" NVARCHAR(50),
    "PURPOSE" INT
);
INSERT INTO PAL_DATA_TBL VALUES (0,    'Y',    '-6.879',   1);
INSERT INTO PAL_DATA_TBL VALUES (0,    'X1',   '0.00',     1);
INSERT INTO PAL_DATA_TBL VALUES (0,    'X2',   'A',        1);
INSERT INTO PAL_DATA_TBL VALUES (0,    'X3',   '1',        1);
INSERT INTO PAL_DATA_TBL VALUES (1,    'Y',    '-3.449',   1);
INSERT INTO PAL_DATA_TBL VALUES (1,    'X1',   '0.50',     1);
INSERT INTO PAL_DATA_TBL VALUES (1,    'X2',   'A',        1);
INSERT INTO PAL_DATA_TBL VALUES (1,    'X3',   '1',        1);
INSERT INTO PAL_DATA_TBL VALUES (2,    'Y',    '6.635',    1);
INSERT INTO PAL_DATA_TBL VALUES (2,    'X1',   '0.54',     1);
INSERT INTO PAL_DATA_TBL VALUES (2,    'X2',   'B',        1);
INSERT INTO PAL_DATA_TBL VALUES (2,    'X3',   '1',        1);
INSERT INTO PAL_DATA_TBL VALUES (3,    'Y',    '11.844',   2);
INSERT INTO PAL_DATA_TBL VALUES (3,    'X1',   '1.04',     2);
INSERT INTO PAL_DATA_TBL VALUES (3,    'X2',   'B',        2);
INSERT INTO PAL_DATA_TBL VALUES (3,    'X3',   '1',        2);
INSERT INTO PAL_DATA_TBL VALUES (4,    'Y',    '2.786',    1);
INSERT INTO PAL_DATA_TBL VALUES (4,    'X1',   '1.50',     1);
INSERT INTO PAL_DATA_TBL VALUES (4,    'X2',   'A',        1);
INSERT INTO PAL_DATA_TBL VALUES (4,    'X3',   '1',        1);
INSERT INTO PAL_DATA_TBL VALUES (5,    'Y',    '2.389',    1);
INSERT INTO PAL_DATA_TBL VALUES (5,    'X1',   '0.04',     1);
INSERT INTO PAL_DATA_TBL VALUES (5,    'X2',   'B',        1);
INSERT INTO PAL_DATA_TBL VALUES (5,    'X3',   '2',        1);
INSERT INTO PAL_DATA_TBL VALUES (6,    'Y',    '-0.011',   2);
INSERT INTO PAL_DATA_TBL VALUES (6,    'X1',   '2.00',     2);
```

```
INSERT INTO PAL_DATA_TBL VALUES (6,    'X2',   'A',        2);
INSERT INTO PAL_DATA_TBL VALUES (6,    'X3',   '2',        2);
INSERT INTO PAL_DATA_TBL VALUES (7,    'Y',    '8.839',    1);
INSERT INTO PAL_DATA_TBL VALUES (7,    'X1',   '2.04',     1);
INSERT INTO PAL_DATA_TBL VALUES (7,    'X2',   'B',        1);
INSERT INTO PAL_DATA_TBL VALUES (7,    'X3',   '2',        1);
INSERT INTO PAL_DATA_TBL VALUES (8,    'Y',    '4.689',    1);
INSERT INTO PAL_DATA_TBL VALUES (8,    'X1',   '1.54',     1);
INSERT INTO PAL_DATA_TBL VALUES (8,    'X2',   'B',        1);
INSERT INTO PAL_DATA_TBL VALUES (8,    'X3',   '1',        1);
INSERT INTO PAL_DATA_TBL VALUES (9,    'Y',    '-5.507',   1);
INSERT INTO PAL_DATA_TBL VALUES (9,    'X1',   '1.00',     1);
INSERT INTO PAL_DATA_TBL VALUES (9,    'X2',   'A',        1);
INSERT INTO PAL_DATA_TBL VALUES (9,    'X3',   '2',        1);
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO PAL_PARAMETER_TBL VALUES ('FUNCTION',null,null,'MLR');
INSERT INTO PAL_PARAMETER_TBL VALUES ('OUTPUT_COEFCOV',1,null,null);
DROP TABLE PAL_DATA_MODEL_TBL;
CREATE COLUMN TABLE PAL_DATA_MODEL_TBL(
     "ROW_INDEX" INTEGER,
     "PART_INDEX" INTEGER,
     "MODEL_CONTENT" VARCHAR (5000)
);
DO BEGIN
  meta_data_tab = SELECT * FROM META_DATA_TAB;
  data_tab = SELECT * FROM PAL_DATA_TBL;
  control_tab = SELECT * FROM PAL_PARAMETER_TBL;
  CALL
_SYS_AFL.PAL_UNIFIED_REGRESSION_PIVOT(:meta_data_tab, :data_tab, :control_tab,
model_tab, stats_tab, optimal_parameter_tab, partition_result,
place_holder_1_tab, place_holder_2_tab);
  INSERT into PAL_DATA_MODEL_TBL SELECT * FROM :model_tab;
END;
-------------------------------Predict------------------------------
DROP TABLE PAL_DATA_TBL_PRE;
CREATE COLUMN TABLE PAL_DATA_TBL_PRE(
    "ID" INT,
    "VARIABLE_NAME" NVARCHAR(50),
    "VALUE" NVARCHAR(50)
);
INSERT INTO PAL_DATA_TBL_PRE VALUES (0,    'X1',    '1.69');
INSERT INTO PAL_DATA_TBL_PRE VALUES (0,    'X2',    'B');
INSERT INTO PAL_DATA_TBL_PRE VALUES (0,    'X3',    '1');
INSERT INTO PAL_DATA_TBL_PRE VALUES (1,    'X1',    '0.054');
INSERT INTO PAL_DATA_TBL_PRE VALUES (1,    'X2',    'B');
INSERT INTO PAL_DATA_TBL_PRE VALUES (1,    'X3',    '2');
INSERT INTO PAL_DATA_TBL_PRE VALUES (2,    'X1',    '0.123');
INSERT INTO PAL_DATA_TBL_PRE VALUES (2,    'X2',    'A');
INSERT INTO PAL_DATA_TBL_PRE VALUES (2,    'X3',    '2');
INSERT INTO PAL_DATA_TBL_PRE VALUES (3,    'X1',    '1.980');
INSERT INTO PAL_DATA_TBL_PRE VALUES (3,    'X2',    'A');
INSERT INTO PAL_DATA_TBL_PRE VALUES (3,    'X3',    '1');
INSERT INTO PAL_DATA_TBL_PRE VALUES (4,    'X1',    '0.563');
INSERT INTO PAL_DATA_TBL_PRE VALUES (4,    'X2',    'A');
INSERT INTO PAL_DATA_TBL_PRE VALUES (4,    'X3',    '1');
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
```

```
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SIGNIFICANCE_LEVEL',null,0.15,null);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('INTERVAL',1,null,null);
CALL _SYS_AFL.PAL_UNIFIED_REGRESSION_PREDICT_PIVOT(PAL_DATA_TBL_PRE,
PAL_DATA_MODEL_TBL, #PAL_PARAMETER_TBL, ?,?);
```

**Expected Result**:

RESULT:

Result 1 × Result 2 × Messages × History

Rows (5) SQL

| | ID | SCORE | LOWER_BOUND | UPPER_BOUND | REASON |
|---|---|---|---|---|---|
| 1 | 0 | 10.314759999999998 | 5.647079323602136 | 14.98244067639786 | NULL |
| 2 | 1 | 1.6859255999999974 | -4.323975918487589 | 7.695827118487584 | NULL |
| 3 | 2 | -7.4095608 | -13.167031289599453 | -1.6520903104005464 | NULL |
| 4 | 3 | 2.0215920000000014 | -3.387594310274483 | 7.430778310274485 | NULL |
| 5 | 4 | -3.1226847999999996 | -6.987703277474724 | 0.7423336774747251 | NULL |

## 5.13.11 Unified Massive Regression

The unified massive regression execution function is a unified interface to call regression algorithms with unpivot data format.

The unified massive regression algorithms include:

- GEO (Bi-Variate Geometric Regression)
- LOG (Bi-Variate Natural Logarithmic Regression)
- EXP (Exponential Regression)
- GLM (Generalised Linear Models)
- MLR (Multiple Linear Regression)
- POL(Polynomial Regression)
- DT (Decision Trees)
- HGBT (Hybrid Gradient Boosting Tree)
- MLP (Multilayer Perceptron)
- RDT (Random Decision Trees)
- SVM (Support Vector Machine)

The keyword group name ('PAL_MASSIVE_PROCESSING_SPECIAL_GROUP_ID') is valid for all groups if the group name is not defined. For example, There are 2 groups ('A', 'B'),

```
INSERT INTO #PAL_PARAMETER_TBL VALUES
('PAL_MASSIVE_PROCESSING_SPECIAL_GROUP_ID', 'ALPHA', NULL, 1.0, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES
('PAL_MASSIVE_PROCESSING_SPECIAL_GROUP_ID', 'VERBOSE', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('B', 'VERBOSE', 0, NULL, NULL);
```

Therefore, A: ALPHA (1.0), VERBOSE (1); B: VERBOSE (0).

For more details on the feature of the unified massive regression, see the corresponding topics (Local Interpretability, Data Partition...) in the chapter on unified regression.

Moreover, it is clear that the models created from the unified massive regression and the unified regression are incompatible.

## Prerequisites

- All prerequisites for the unified regression function apply in the unified massive regression execution interface.

## _SYS_AFL.PAL_UNIFIED_MASSIVE_REGRESSION

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA Table>` | IN |
| 2 | `<PARAMETER Table>` | IN |
| 3 | `<MODEL Table>` | OUT |
| 4 | `<STATISTICS Table>` | OUT |
| 5 | `<OPTPARAMETER Table>` | OUT |
| 6 | `<PARTITION_TYPE Table>` | OUT |
| 7 | `<ERRORMSG Table>` | OUT |
| 8 | `<PLACE_HOLDER1 Table>` | OUT |
| 9 | `<PLACE_HOLDER2 Table>` | OUT |

### Input Table

| Table | Column | Data Type | Description |
|---|---|---|---|
| DATA | 1st column | INTEGER, VRACHAR or NVARCHAR | Group id. |
| | 2nd column | INTEGER, VRACHAR or NVARCHAR | Record id. If this column is absent, the KEY parameter must be set to 0. |
| | Feature columns | INTEGER, DOUBLE, DECIMAL(p,s), VARCHAR, or NVARCHAR | Variable values. |
| | LABEL column | INTEGER or DOUBLE | Dependent variable. |
| | Purpose column (Optional) | INTEGER | An optional column used for user-defined data partition. This column is needed if PARTITION_METHOD is set to 1. Otherwise, the column does not exist. The valid values are:<br>• 1: training<br>• 2: validation |

| Table | Column | Data Type | Description |
|---|---|---|---|
| PARAMETER | 1st column | INTEGER, VRACHAR or NVARCHAR | Group id. |
| | 2nd column | VARCHAR or NVARCHAR | Parameter name. |
| | 3rd column | INTEGER | Int value. |
| | 4th column | DOUBLE | Double value. |
| | 5th column | VARCHAR or NVARCHAR | String value. |

### Parameter Table

Mandatory Parameters

The following parameters are mandatory and must be given a value.

| Name | Data Type | Description |
|---|---|---|
| FUNCTION | VARCHAR or NVARCHAR | The algorithm name, including:<br>• GEO;<br>• LOG;<br>• EXP;<br>• GLM;<br>• MLR;<br>• POL;<br>• DT;<br>• HGBT;<br>• MLP;<br>• RDT;<br>• SVM. |

Optional Parameters

The parameter of the unified massive regression execution function with pivot is the same as the unified regression function.

See Parameter Tables [page 1910] for details about parameters for a specific algorithm.

### Output Tables

| Table | Column | Column Name | Column Data Type |
|---|---|---|---|
| MODEL | 1st column | GROUP_ID | NVARCHAR |
| | 2nd column | ROW_INDEX | INTEGER |
| | 3rd column | PART_INDEX | INTEGER |
| | 4th column | MODEL_CONTENT | VARCHAR, NVARCHAR, NCLOB |
| STATISTICS | 1st column | GROUP_ID | NVARCHAR |
| | 2nd column | STAT_NAME | VARCHAR or NVARCHAR |

| Table | Column | Column Name | Column Data Type |
|---|---|---|---|
| | 3rd column | STAT_VALUE | VARCHAR or NVARCHAR |
| OPTPARAMETER | 1st column | GROUP_ID | NVARCHAR |
| | 2nd column | PARAM_NAME | VARCHAR or NVARCHAR |
| | 3rd column | INT_VALUE | INTEGER |
| | 4th column | DOUBLE_VALUE | DOUBLE |
| | 5th column | STRING_VALUE | VARCHAR or NVARCHAR |
| PARTITION_TYPE | 1st column | GROUP_ID | NVARCHAR |
| | 2nd column | ID column name | ID column data type |
| | 3rd column | TYPE | INTEGER |
| ERRORMSG | 1st column | GROUP_ID | VARCHAR |
| | 2nd column | ERROR_TIMESTAMP | VARCHAR, NVARCHAR |
| | 3rd column | ERRORCODE | INTEGER |
| | 4th column | MASSAGE | VARCHAR, NVARCHAR |
| PLCAE_HOLDER_1 | 1st column | OBJECT | VARCHAR, NVARCHAR |
| | 2nd column | KEY | VARCHAR, NVARCHAR |
| | 3rd column | VALUE | VARCHAR, NVARCHAR |
| PLCAE_HOLDER_2 | 1st column | OBJECT | VARCHAR, NVARCHAR |
| | 2nd column | KEY | VARCHAR, NVARCHAR |
| | 3rd column | VALUE | VARCHAR, NVARCHAR |

### Example

Assume that:

- DM_PAL is a schema belonging to USER1;
- USER1 has been assigned the AFL__SYS_AFL_AFLPAL_EXECUTE or AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION role.

Example: Hybrid Gradient Boosting Tree

```
SET SCHEMA DM_PAL;
-----------------------------------Unified Massive Regression (Hybrid Gradient
Boosting Tree)-------------------------
DROP TABLE PAL_DATA_TBL;
CREATE COLUMN TABLE PAL_DATA_TBL(
     "GROUP_ID" VARCHAR(20),
     "ID" INTEGER,
     "X1" DOUBLE,
     "X2" VARCHAR (100),
     "X3" INTEGER,
     "Y" DOUBLE
);
INSERT INTO PAL_DATA_TBL VALUES ('1', 0, 0.0, 'A', 1, -6.879);
INSERT INTO PAL_DATA_TBL VALUES ('1', 1, 0.50, 'A', 1, -3.449);
INSERT INTO PAL_DATA_TBL VALUES ('1', 2, 0.54, 'B', 1, 6.635);
INSERT INTO PAL_DATA_TBL VALUES ('1', 3, 1.04, 'B', 1, 11.844);
INSERT INTO PAL_DATA_TBL VALUES ('1', 4, 1.50, 'A', 1, 2.786);
```

```
INSERT INTO PAL_DATA_TBL VALUES ('1', 5, 0.04, 'B', 2, 2.389);
INSERT INTO PAL_DATA_TBL VALUES ('1', 6, 2.00, 'A', 2, -0.011);
INSERT INTO PAL_DATA_TBL VALUES ('1', 7, 2.04, 'B', 2, 8.839);
INSERT INTO PAL_DATA_TBL VALUES ('1', 8, 1.54, 'B', 1, 4.689);
INSERT INTO PAL_DATA_TBL VALUES ('1', 9, 1.00, 'A', 2, -5.507);
INSERT INTO PAL_DATA_TBL VALUES ('2', 0, 10.0, 'B', 2, -3.879);
INSERT INTO PAL_DATA_TBL VALUES ('2', 1, 0.30, 'A', 1, -13.449);
INSERT INTO PAL_DATA_TBL VALUES ('2', 2, 1.54, 'B', 2, 6.035);
INSERT INTO PAL_DATA_TBL VALUES ('2', 3, 10.04, 'A', 1, 1.844);
INSERT INTO PAL_DATA_TBL VALUES ('2', 4, 1.55, 'A', 1, 22.786);
INSERT INTO PAL_DATA_TBL VALUES ('2', 5, 10.04, 'B', 2, 12.389);
INSERT INTO PAL_DATA_TBL VALUES ('2', 6, 2.50, 'A', 2, -1.011);
INSERT INTO PAL_DATA_TBL VALUES ('2', 7, 2.04, 'B', 2, 8.839);
INSERT INTO PAL_DATA_TBL VALUES ('2', 8, 1.54, 'B', 1, 4.689);
INSERT INTO PAL_DATA_TBL VALUES ('2', 9, 1.00, 'A', 2, -5.507);
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL (
    "GROUP_ID" VARCHAR(50),
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO PAL_PARAMETER_TBL VALUES ('PAL_MASSIVE_PROCESSING_SPECIAL_GROUP_ID',
'FUNCTION',null,null,'HGBT');
INSERT INTO PAL_PARAMETER_TBL VALUES ('PAL_MASSIVE_PROCESSING_SPECIAL_GROUP_ID',
'KEY',1,null,null);
INSERT INTO PAL_PARAMETER_TBL VALUES ('PAL_MASSIVE_PROCESSING_SPECIAL_GROUP_ID',
'PARTITION_METHOD',2,null,null);
INSERT INTO PAL_PARAMETER_TBL VALUES ('PAL_MASSIVE_PROCESSING_SPECIAL_GROUP_ID',
'PARTITION_TRAINING_PERCENT', null,0.7,null);
INSERT INTO PAL_PARAMETER_TBL VALUES ('PAL_MASSIVE_PROCESSING_SPECIAL_GROUP_ID',
'OUTPUT_PARTITION_RESULT', 1,null, null);
INSERT INTO PAL_PARAMETER_TBL VALUES ('PAL_MASSIVE_PROCESSING_SPECIAL_GROUP_ID',
'THREAD_RATIO', null,0.5,null);
INSERT INTO PAL_PARAMETER_TBL VALUES ('PAL_MASSIVE_PROCESSING_SPECIAL_GROUP_ID',
'LABEL', null,null,'X3');
DROP TABLE PAL_DATA_MODEL_TBL;
CREATE COLUMN TABLE PAL_DATA_MODEL_TBL(
     "GROUP_ID" VARCHAR(50),
     "ROW_INDEX" INTEGER,
     "PART_INDEX" INTEGER,
     "MODEL_CONTENT" VARCHAR (5000)
);
DROP TABLE PAL_DATA_STATISTICS_TBL;
CREATE COLUMN TABLE PAL_DATA_STATISTICS_TBL(
     "GROUP_ID" INTEGER,
     "STAT_NAME" NVARCHAR (100),
     "STAT_VALUE" NVARCHAR (200)
);
DROP TABLE PAL_DATA_OPTIMAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_DATA_OPTIMAL_PARAMETER_TBL(
     "GROUP_ID" INTEGER,
     "PARAM_NAME" NVARCHAR (100),
     "INT_VALUE" INTEGER,
     "DOUBLE_VALUE" DOUBLE,
     "STRING_VALUE" NVARCHAR (200)
);
DROP TABLE PAL_DATA_PARTITION_TYPE_TBL;
CREATE COLUMN TABLE PAL_DATA_PARTITION_TYPE_TBL(
     "GROUP_ID" INTEGER,
     "ID" INTEGER,
     "TYPE" INTEGER
);
DO BEGIN
  data_tab = SELECT * FROM PAL_DATA_TBL;
  control_tab = SELECT * FROM PAL_PARAMETER_TBL;
```

```
  CALL _SYS_AFL.PAL_UNIFIED_MASSIVE_REGRESSION(:data_tab, :control_tab,
model_tab, stats_tab, optimal_parameter_tab, partition_result, errormsg,
place_holder_1_tab, place_holder_2_tab);
  INSERT into PAL_DATA_MODEL_TBL SELECT * FROM :model_tab;
  INSERT into PAL_DATA_STATISTICS_TBL SELECT * FROM :stats_tab;
  INSERT into PAL_DATA_OPTIMAL_PARAMETER_TBL SELECT *
FROM :optimal_parameter_tab;
  INSERT into PAL_DATA_PARTITION_TYPE_TBL SELECT * FROM :partition_result;
  select * from :errormsg;
END;
select * from PAL_DATA_MODEL_TBL;
select * from PAL_DATA_STATISTICS_TBL;
select * from PAL_DATA_OPTIMAL_PARAMETER_TBL;
select * from PAL_DATA_PARTITION_TYPE_TBL;
```

**Expected Result**:

MODEL:

Rows (22)

| | GROUP_ID | ROW_INDEX | PART_INDEX | |
|---|---|---|---|---|
| 1 | 2 | 1 | -1 | {"ver":"rapidjson","model_version":"2.0","nrow":7,"nclass":1,"var":{"repl":[2.5,0,1.844],"map":[["B","A"]],"name":["X1","X2","Y","X3"],"typ":[100,83,100,105],"cate":[[]]},"param":{"obj" |
| 2 | 2 | 2 | 0 | {"nodes":[{"id":0,"n":7,"gn":21.125,"w":0.4875,"ch":[]}],"height":0,"nnode":1} |
| 3 | 2 | 3 | 1 | {"nodes":[{"id":0,"n":7,"gn":11.490019531249997,"w":0.3595312499999999,"ch":[]}],"height":0,"nnode":1} |
| 4 | 2 | 4 | 2 | {"nodes":[{"id":0,"n":7,"gn":6.249493435668947,"w":0.26515429687500005,"ch":[]}],"height":0,"nnode":1} |
| 5 | 2 | 5 | 3 | {"nodes":[{"id":0,"n":7,"gn":3.399138538994314,"w":0.19555129394531255,"sp":0,"md":82,"x":10.02,"ch":[1,2]},{"id":1,"n":5,"gn":3.2842270965735129,"w":0.22195361328125 |
| 6 | 2 | 6 | 4 | {"nodes":[{"id":0,"n":7,"gn":1.9886050465538517,"w":0.14957207885742189,"sp":0,"md":82,"x":10.02,"ch":[1,2]},{"id":1,"n":5,"gn":1.8473777418226008,"w":0.1664652099609 |
| 7 | 2 | 7 | 5 | {"nodes":[{"id":0,"n":7,"gn":1.1684389407374878,"w":0.11465137628173827,"ch":[]}],"height":0,"nnode":1} |
| 8 | 2 | 8 | 6 | {"nodes":[{"id":0,"n":7,"gn":0.635521242610499,"w":0.08455539000778197,"sp":0,"md":82,"x":10.02,"ch":[1,2]},{"id":1,"n":5,"gn":0.6167839194960318,"w":0.09618606340026 |
| 9 | 2 | 9 | 7 | {"nodes":[{"id":0,"n":7,"gn":0.3716154093355791,"w":0.06465812675159453,"sp":1,"md":82,"x":0,"ch":[1,2]},{"id":1,"n":4,"gn":0.3450301551705533,"w":0.0788069971072998," |
| 10 | 2 | 10 | 8 | {"nodes":[{"id":0,"n":7,"gn":0.2173111783781807,"w":0.04944442093831672,"sp":1,"md":82,"x":0,"ch":[1,2]},{"id":1,"n":4,"gn":0.1992894176265116 3,"w":0.059893317801547 |
| 11 | 2 | 11 | 9 | {"nodes":[{"id":0,"n":7,"gn":0.12696152940775744,"w":0.03779308410062972 7,"sp":1,"md":82,"x":0,"ch":[1,2]},{"id":1,"n":4,"gn":0.11510956762107302,"w":0.04551892152917 |
| 12 | 1 | 1 | -1 | {"ver":"rapidjson","model_version":"2.0","nrow":7,"nclass":1,"var":{"repl":[1.04,0,2.786],"map":[["A","B"]],"name":["X1","X2","Y","X3"],"typ":[100,83,100,105],"cate":[[]]},"param":{"ob |
| 13 | 1 | 2 | 0 | {"nodes":[{"id":0,"n":7,"gn":12.5,"w":0.375,"ch":[]}],"height":0,"nnode":1} |
| 14 | 1 | 3 | 1 | {"nodes":[{"id":0,"n":7,"gn":6.798828125,"w":0.2765625,"ch":[]}],"height":0,"nnode":1} |
| 15 | 1 | 4 | 2 | {"nodes":[{"id":0,"n":7,"gn":3.6979251098632819,"w":0.20396484375,"sp":0,"md":82,"x":0.77,"ch":[1,2]},{"id":1,"n":2,"gn":0.1618782552083333,"w":0.06968749999999999,"ch" |
| 16 | 1 | 5 | 3 | {"nodes":[{"id":0,"n":7,"gn":2.1975742312316894,"w":0.1572345703125,"sp":0,"md":82,"x":0.77,"ch":[1,2]},{"id":1,"n":2,"gn":0.10360208333333329,"w":0.05574999999999999, |
| 17 | 1 | 6 | 4 | {"nodes":[{"id":0,"n":7,"gn":1.308619082305813 5,"w":0.12133410351562499,"sp":0,"md":82,"x":1.75,"ch":[1,2]},{"id":1,"n":5,"gn":0.3585157939866577,"w":0.0733330546875,"s |
| 18 | 1 | 7 | 5 | {"nodes":[{"id":0,"n":7,"gn":0.7934192775989752,"w":0.09447733523437498,"sp":0,"md":82,"x":0.77,"ch":[1,2]},{"id":1,"n":2,"gn":0.02186847652629532,"w":0.02561355687499 |
| 19 | 1 | 8 | 6 | {"nodes":[{"id":0,"n":7,"gn":0.4700132776778921,"w":0.07271622496992185,"sp":0,"md":82,"x":1.75,"ch":[1,2]},{"id":1,"n":5,"gn":0.10055439695021841,"w":0.0388370435828 |
| 20 | 1 | 9 | 7 | {"nodes":[{"id":0,"n":7,"gn":0.31199908138616047,"w":0.05924516575716794,"sp":0,"md":82,"x":1.75,"ch":[1,2]},{"id":1,"n":5,"gn":0.07041326897517929,"w":0.0324992159078 |
| 21 | 1 | 10 | 8 | {"nodes":[{"id":0,"n":7,"gn":0.20743232358388093,"w":0.04830749051978026,"sp":2,"md":82,"x":1.3875,"ch":[1,2]},{"id":1,"n":3,"gn":0.2100585927937156,"w":0.06874822425 |
| 22 | 1 | 11 | 9 | {"nodes":[{"id":0,"n":7,"gn":0.13632617752953098,"w":0.03916209260505908,"sp":2,"md":82,"x":1.3875,"ch":[1,2]},{"id":1,"n":3,"gn":0.1356422851391674 9,"w":0.0552444695 |

STATISTIC:

Rows (18)

| | GROUP_ID | STAT_NAME | STAT_VALUE |
|---|---|---|---|
| 1 | 2 | TEST_EVAR | 0 |
| 2 | 2 | TEST_MAE | 0.8667258198238645 |
| 3 | 2 | TEST_MAPE | 0.8667258198238645 |
| 4 | 2 | TEST_MAX_ERROR | 0.8956610462372183 |
| 5 | 2 | TEST_MSE | 0.751632270413146 |
| 6 | 2 | TEST_R2 | 0 |
| 7 | 2 | TEST_RMSE | 0.8669672833579973 |
| 8 | 2 | TEST_WMAPE | 0.8667258198238645 |
| 9 | 2 | IMP | {"X1":"0.3643681986729232","X2":"0.44927196315801793","Y":"0.1863598381690589 3"} |
| 10 | 1 | TEST_EVAR | -0.4950704224936031 |
| 11 | 1 | TEST_MAE | 0.46318169873807985 |
| 12 | 1 | TEST_MAPE | 0.2904428019931853 |
| 13 | 1 | TEST_MAX_ERROR | 1.0364333804693673 |
| 14 | 1 | TEST_MSE | 0.38411881582989565 |
| 15 | 1 | TEST_R2 | -0.7285346712345306 |
| 16 | 1 | TEST_RMSE | 0.61977319708898 |
| 17 | 1 | TEST_WMAPE | 0.3473862740535599 |
| 18 | 1 | IMP | {"X1":"0.7210725097563332","X2":"0.0620099499659669","Y":"0.2169175402776999"} |

PARTITION_TYPE:

Rows (20)

| | GROUP_ID | ID | TYPE |
|---|---|---|---|
| 1 | 2 | 0 | 1 |
| 2 | 2 | 2 | 1 |
| 3 | 2 | 3 | 1 |
| 4 | 2 | 5 | 1 |
| 5 | 2 | 6 | 1 |
| 6 | 2 | 7 | 1 |
| 7 | 2 | 9 | 1 |
| 8 | 2 | 1 | 2 |
| 9 | 2 | 4 | 2 |
| 10 | 2 | 8 | 2 |
| 11 | 1 | 0 | 1 |
| 12 | 1 | 2 | 1 |
| 13 | 1 | 3 | 1 |
| 14 | 1 | 4 | 1 |
| 15 | 1 | 6 | 1 |
| 16 | 1 | 7 | 1 |
| 17 | 1 | 9 | 1 |
| 18 | 1 | 1 | 2 |
| 19 | 1 | 5 | 2 |
| 20 | 1 | 8 | 2 |

# _SYS_AFL.PAL_UNIFIED_MASSIVE_REGRESSION_PREDICT

## Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA Table>` | IN |
| 2 | `<MODEL Table>` | IN |
| 3 | `<PARAMETER Table>` | IN |
| 4 | `<RESULT Table>` | OUT |
| 5 | `<ERRORMSG Table>` | OUT |
| 6 | `<PLACE_HOLDER1 Table>` | OUT |

## Input Table

| Table | Column | Data Type | Description |
|---|---|---|---|
| DATA | 1st column | INTEGER, VRACHAR or NVARCHAR | Group id. |
| | 2nd column | INTEGER, VRACHAR or NVARCHAR | Record id. |
| | Feature columns | INTEGER, DOUBLE, DECIMAL(p,s), VARCHAR, or NVARCHAR | Independent variables. |
| MODEL | 1st column | INTEGER, VRACHAR or NVARCHAR | Group id. |
| | 2nd column | INTEGER | Row index. |
| | 3rd column | INTEGER | Part index. |

| Table | Column | Data Type | Description |
|---|---|---|---|
| | 4th column | VARCHAR, NVARCHAR, or NCLOB | Model content. |
| PARAMETER | 1st column | INTEGER, VRACHAR or NVARCHAR | Group id. |
| | 2nd column | VARCHAR or NVARCHAR | Parameter name. |
| | 3rd column | INTEGER | Int value. |
| | 4th column | DOUBLE | Double value. |
| | 5th column | VARCHAR or NVARCHAR | String value. |

### Parameter Table

The parameter of the unified massive regression prediction execution function is the same as the unified regression prediction function.

See Parameter Tables [page 1910] for details about parameters for a specific algorithm.

### Output Tables

| Table | Column | Column Name | Column Data Type |
|---|---|---|---|
| RESULT | 1st column | GROUP_ID | Group id column data type. |
| | 2nd column | ID column name | ID column data type |
| | 3rd column | SCORE | DOUBLE |
| | 4th column | UPPER_BOUND | DOUBLE |
| | 5th column | LOWER_BOUND | DOUBLE |
| | 6th column | REASON | VARCHAR or NVARCHAR |
| ERRORMSG | 1st column | GROUP_ID | Group id column data type. |
| | 2nd column | ERROR_TIMESTAMP | VARCHAR, NVARCHAR |
| | 3rd column | ERRORCODE | INTEGER |
| | 4th column | MASSAGE | VARCHAR, NVARCHAR |
| PLCAE_HOLDER1 | 1st column | GROUP_ID | Group id column data type. |
| | 2nd column | OBJECT | VARCHAR, NVARCHAR |
| | 3rd column | KEY | VARCHAR, NVARCHAR |
| | 4th column | VALUE | VARCHAR, NVARCHAR |

### Example

Assume that:

- DM_PAL is a schema belonging to USER1;
- USER1 has been assigned the AFL__SYS_AFL_AFLPAL_EXECUTE or AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION role.

Example: Hybrid Gradient Boosting Tree Prediction

```
-------------------------------------------Unified Massive Predict(Hybrid
Gradient Boosting Tree)--------------------------------
DROP TABLE PAL_DATA_TBL_PRE;
CREATE COLUMN TABLE PAL_DATA_TBL_PRE(
     "GROUP_ID" VARCHAR(50),
     "ID" INTEGER,
     "X1" DOUBLE,
     "X2" VARCHAR (100),
     "Y" DOUBLE
);
INSERT INTO PAL_DATA_TBL_PRE VALUES ('1', 0, 1.69, 'B', 1.2);
INSERT INTO PAL_DATA_TBL_PRE VALUES ('1', 1, 0.054, 'B', 2.1);
INSERT INTO PAL_DATA_TBL_PRE VALUES ('1', 2, 0.123, 'A', 2.4);
INSERT INTO PAL_DATA_TBL_PRE VALUES ('1', 3, 1.980, 'A', 1.8);
INSERT INTO PAL_DATA_TBL_PRE VALUES ('1', 4, 0.563, 'A', 1.0);
INSERT INTO PAL_DATA_TBL_PRE VALUES ('2', 0, 10.69, 'B', 3.2);
INSERT INTO PAL_DATA_TBL_PRE VALUES ('2', 1, 1.054, 'B', 2.1);
INSERT INTO PAL_DATA_TBL_PRE VALUES ('2', 2, 10.123, 'A', 6.4);
INSERT INTO PAL_DATA_TBL_PRE VALUES ('2', 3, 1.44, 'A', 1.8);
INSERT INTO PAL_DATA_TBL_PRE VALUES ('2', 4, 9.543, 'B', 1.0);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "GROUP_ID" VARCHAR(50),
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES
('PAL_MASSIVE_PROCESSING_SPECIAL_GROUP_ID', 'FUNCTION',null,null,'HGBT');
CALL _SYS_AFL.PAL_UNIFIED_MASSIVE_REGRESSION_PREDICT(PAL_DATA_TBL_PRE,
PAL_DATA_MODEL_TBL, #PAL_PARAMETER_TBL, ?,?,?);
```

**Expected Result**:

RESULT:

Rows (10)

| | GROUP_ID | ID | SCORE | UPPER_BOUND | LOWER_BOUND | REASON |
|---|---|---|---|---|---|---|
| 1 | 2 | 0 | 1.608943855252111 | NULL | NULL | [{"attr":"X1","val":-0.25745518012905946,"pct":56.131916351954227},{"attr":"X2","val":0.1602236258308358,"pct":34.93291204408809 |
| 2 | 2 | 1 | 1.8956610462372183 | NULL | NULL | [{"attr":"X1","val":0.10591073713702204,"pct":63.5269600079176},{"attr":"X2","val":0.05169989954986173,"pct":31.010429536226448 |
| 3 | 2 | 2 | 1.1021594596542013 | NULL | NULL | [{"attr":"X1","val":-0.41806748450263817,"pct":64.81685592558127},{"attr":"X2","val":-0.18594846539349534,"pct":28.8293046882890 |
| 4 | 2 | 3 | 1.8522582066171875 | NULL | NULL | [{"attr":"X1","val":0.1769892107523828,"pct":71.11491423119135},{"attr":"X2","val":-0.06278141368552993,"pct":25.225802355860816 |
| 5 | 2 | 4 | 1.8956610462372183 | NULL | NULL | [{"attr":"X2","val":0.09562987583083579,"pct":57.36033438808005},{"attr":"X1","val":0.06198076085604798,"pct":37.17705515606401 |
| 6 | 1 | 0 | 1.315464990019172 | NULL | NULL | [{"attr":"X1","val":-0.11896769655332518,"pct":48.390107522323678},{"attr":"Y","val":0.11864619223355268,"pct":48.25933564850068 |
| 7 | 1 | 1 | 0.9635666195306328 | NULL | NULL | [{"attr":"X1","val":-0.38017844939478337,"pct":91.3068718061549},{"attr":"Y","val":0.019422521368218445,"pct":4.66467699982302},{ |
| 8 | 1 | 2 | 0.8893988202770597 | NULL | NULL | [{"attr":"X1","val":-0.3695955770812023,"pct":87.64804041623246},{"attr":"X2","val":-0.050320319756129428,"pct":11.9332527044120 |
| 9 | 1 | 3 | 1.5830148183970596 | NULL | NULL | [{"attr":"X1","val":0.3257759371231938,"pct":76.88303731171129},{"attr":"X2","val":-0.07413178246168675,"pct":17.495081580657819 |
| 10 | 1 | 4 | 1.0876223055977656 | NULL | NULL | [{"attr":"X1","val":-0.31508142393497337,"pct":70.16192713780544},{"attr":"Y","val":0.11457534583882942,"pct":25.51349097683355} |

# _SYS_AFL.PAL_UNIFIED_MASSIVE_REGRESSION_SCORE

## Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA Table>` | IN |
| 2 | `<MODEL Table>` | IN |

| Position | Table Name | Parameter Type |
|---|---|---|
| 3 | `<PARAMETER Table>` | IN |
| 4 | `<RESULT Table>` | OUT |
| 5 | `<STATISTIC Table>` | OUT |
| 6 | `<ERRORMSG Table>` | OUT |

### Input Table

| Table | Column | Data Type | Description |
|---|---|---|---|
| DATA | 1st column | INTEGER, VRACHAR or NVARCHAR | Group id. |
| | 2nd column | INTEGER, VRACHAR or NVARCHAR | Record id. |
| | Feature columns | INTEGER, DOUBLE, DECIMAL(p,s), VARCHAR, or NVARCHAR | Independent variables. |
| | LABEL column | INTEGER or DOUBLE | Dependent variable. |
| MODEL | 1st column | INTEGER, VRACHAR or NVARCHAR | Group id. |
| | 2nd column | INTEGER | Row index. |
| | 3rd column | INTEGER | Part index. |
| | 4th column | VARCHAR, NVARCHAR, or NCLOB | Model content. |
| PARAMETER | 1st column | INTEGER, VRACHAR or NVARCHAR | Group id. |
| | 2nd column | VARCHAR or NVARCHAR | Parameter name. |
| | 3rd column | INTEGER | Int value. |
| | 4th column | DOUBLE | Double value. |
| | 5th column | VARCHAR or NVARCHAR | String value. |

### Parameter Table

The parameter of the unified massive regression scoring execution function is the same as the unified regression scoring function.

See Parameter Tables [page 1910] for details about parameters for a specific algorithm.

### Output Tables

| Table | Column | Column Name | Column Data Type |
|---|---|---|---|
| RESULT | 1st column | GROUP_ID | Group id column data type. |
| | 2nd column | ID column name | ID column data type |
| | 3rd column | SCORE | DOUBLE |

| Table | Column | Column Name | Column Data Type |
|---|---|---|---|
| | 4th column | UPPER_BOUND | DOUBLE |
| | 5th column | LOWER_BOUND | DOUBLE |
| | 6th column | REASON | VARCHAR or NVARCHAR |
| STATISTICS | 1st column | GROUP_ID | Group id column data type. |
| | 2nd column | STAT_NAME | NVARCHAR |
| | 3rd column | STAT_VALUE | NVARCHAR |
| ERRORMSG | 1st column | GROUP_ID | Group id column data type. |
| | 2nd column | ERROR_TIMESTAMP | VARCHAR, NVARCHAR |
| | 3rd column | ERRORCODE | INTEGER |
| | 4th column | MASSAGE | VARCHAR, NVARCHAR |

### Example

Assume that:

- DM_PAL is a schema belonging to USER1;
- USER1 has been assigned the AFL__SYS_AFL_AFLPAL_EXECUTE or AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION role.

Example: Hybrid Gradient Boosting Tree Scoring

```
-------------------------------------------Unified Massive Scoring(Hybrid
Gradient Boosting Tree)--------------------------------
DROP TABLE PAL_DATA_TBL_SCORE;
CREATE COLUMN TABLE PAL_DATA_TBL_SCORE(
     "GROUP_ID" VARCHAR(50),
     "ID" INTEGER,
     "X1" DOUBLE,
     "X2" VARCHAR (100),
     "X3" INTEGER,
     "Y" DOUBLE
);
INSERT INTO PAL_DATA_TBL_SCORE VALUES ('1', 0, 1.69, 'B', 1, 1.2);
INSERT INTO PAL_DATA_TBL_SCORE VALUES ('1', 1, 0.054, 'B', 2, 2.1);
INSERT INTO PAL_DATA_TBL_SCORE VALUES ('1', 2, 0.123, 'A', 2, 2.4);
INSERT INTO PAL_DATA_TBL_SCORE VALUES ('1', 3, 1.980, 'A', 1, 1.8);
INSERT INTO PAL_DATA_TBL_SCORE VALUES ('1', 4, 0.563, 'A', 1, 1.0);
INSERT INTO PAL_DATA_TBL_SCORE VALUES ('2', 0, 10.69, 'B', 1, 3.2);
INSERT INTO PAL_DATA_TBL_SCORE VALUES ('2', 1, 1.054, 'B', 2, 2.1);
INSERT INTO PAL_DATA_TBL_SCORE VALUES ('2', 2, 10.123, 'A', 2, 6.4);
INSERT INTO PAL_DATA_TBL_SCORE VALUES ('2', 3, 1.44, 'A', 1, 1.8);
INSERT INTO PAL_DATA_TBL_SCORE VALUES ('2', 4, 0.563, 'A', 1, 1.0);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "GROUP_ID" VARCHAR(50),
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES
('PAL_MASSIVE_PROCESSING_SPECIAL_GROUP_ID', 'FUNCTION',null,null,'HGBT');
INSERT INTO #PAL_PARAMETER_TBL VALUES
('PAL_MASSIVE_PROCESSING_SPECIAL_GROUP_ID', 'LABEL', null,null,'X3');
```

```
CALL _SYS_AFL.PAL_UNIFIED_MASSIVE_REGRESSION_SCORE(PAL_DATA_TBL_SCORE,
PAL_DATA_MODEL_TBL, #PAL_PARAMETER_TBL, ?,?,?);
```

**Expected Result**:

RESULT:

Rows (10)

| | GROUP_ID | ID | SCORE | UPPER_BOUND | LOWER_BOUND | REASON |
|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 1.315464990019172 | *NULL* | *NULL* | [{"attr":"X1","val":-0.11896769655332518,"pct":48.390107522323678},{"attr":"Y","val":0.11864619223355268,"pct":48.25933564850068 |
| 2 | 1 | 1 | 0.9635666195306328 | *NULL* | *NULL* | [{"attr":"X1","val":-0.38017844939478337,"pct":91.3068718061549},{"attr":"Y","val":0.019422521368218445,"pct":4.66467699982302},{ |
| 3 | 1 | 2 | 0.8893988202770597 | *NULL* | *NULL* | [{"attr":"X1","val":-0.3695955770812023,"pct":87.64804041623246},{"attr":"X2","val":-0.050320319756129428,"pct":11.9332527044120 |
| 4 | 1 | 3 | 1.5830148183970596 | *NULL* | *NULL* | [{"attr":"X1","val":0.3257759371231938,"pct":76.88303731171129},{"attr":"X2","val":-0.07413178246168675,"pct":17.495081580657819 |
| 5 | 1 | 4 | 1.0876223055977656 | *NULL* | *NULL* | [{"attr":"X1","val":-0.31508142393497337,"pct":70.16192713780544},{"attr":"Y","val":0.11457534583882942,"pct":25.51349097683355} |
| 6 | 2 | 0 | 1.608943855252111 | *NULL* | *NULL* | [{"attr":"X1","val":-0.25745518012905946,"pct":56.131916351954227},{"attr":"X2","val":0.1602236258308358,"pct":34.93291204408809 |
| 7 | 2 | 1 | 1.8956610462372183 | *NULL* | *NULL* | [{"attr":"X1","val":0.10591073713702204,"pct":63.5269600079176},{"attr":"X2","val":0.05169989954986173,"pct":31.010429536226448 |
| 8 | 2 | 2 | 1.1021594596542013 | *NULL* | *NULL* | [{"attr":"X1","val":-0.41806748450263817,"pct":64.81685592558127},{"attr":"X2","val":-0.18594846539349534,"pct":28.8293046882890 |
| 9 | 2 | 3 | 1.8522582066171875 | *NULL* | *NULL* | [{"attr":"X1","val":0.1769892107523828,"pct":71.11491423119135},{"attr":"X2","val":-0.06278141368552993,"pct":25.225802355860816 |
| 10 | 2 | 4 | 1.8522582066171875 | *NULL* | *NULL* | [{"attr":"X1","val":0.1769892107523828,"pct":71.11491423119135},{"attr":"X2","val":-0.06278141368552993,"pct":25.225802355860816 |

STATISTICS:

Rows (16)

| | GROUP_ID | STAT_NAME | STAT_VALUE |
|---|---|---|---|
| 1 | 1 | EVAR | -1.0709820087265474 |
| 2 | 1 | MAE | 0.6266273348412609 |
| 3 | 1 | MAPE | 0.4119238788220302 |
| 4 | 1 | MAX_ERROR | 1.1106011797229403 |
| 5 | 1 | MSE | 0.5509462478779537 |
| 6 | 1 | R2 | -1.29560936615814 |
| 7 | 1 | RMSE | 0.7422575347397652 |
| 8 | 1 | WMAPE | 0.4475909534580435 |
| 9 | 2 | EVAR | -0.9138445997100977 |
| 10 | 2 | MAE | 0.6631279525190132 |
| 11 | 2 | MAPE | 0.5629100031081553 |
| 12 | 2 | MAX_ERROR | 0.8978405403457987 |
| 13 | 2 | MSE | 0.5281009947005482 |
| 14 | 2 | R2 | -1.2004208112522843 |
| 15 | 2 | RMSE | 0.7267055763516255 |
| 16 | 2 | WMAPE | 0.4736628232278666 |

## 5.13.12 Unified Massive Regression with Pivoted Input Data

The unified massive regression with pivoted execution function is a unified interface to call regression algorithms with pivot data format.

The unified massive regression with pivoted algorithms include:

- GEO (Bi-Variate Geometric Regression)
- LOG (Bi-Variate Natural Logarithmic Regression)
- EXP (Exponential Regression)
- GLM (Generalised Linear Models)
- MLR (Multiple Linear Regression)
- POL(Polynomial Regression)
- DT (Decision Trees)
- HGBT (Hybrid Gradient Boosting Tree)
- MLP (Multilayer Perceptron)
- RDT (Random Decision Trees)
- SVM (Support Vector Machine)

The keyword group name ('PAL_MASSIVE_PROCESSING_SPECIAL_GROUP_ID') is valid for all groups if the group name is not defined. For example, There are 2 groups ('A', 'B'),

```
INSERT INTO #PAL_PARAMETER_TBL VALUES
('PAL_MASSIVE_PROCESSING_SPECIAL_GROUP_ID', 'ALPHA', NULL, 1.0, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES
('PAL_MASSIVE_PROCESSING_SPECIAL_GROUP_ID', 'VERBOSE', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('B', 'VERBOSE', 0, NULL, NULL);
```

Therefore, A: ALPHA (1.0), VERBOSE (1); B: VERBOSE (0).

For more details on the feature of the unified massive regression with pivot, see the corresponding topics (Local Interpretability, Data Partition...) in the chapter on unified regression with pivot.

Moreover, it is clear that the models created from the unified massive regression with pivot and the unified regression with pivot are incompatible.

## Prerequisites

- All prerequisites for the unified regression with pivot function apply in the unified massive regression with pivot execution interface.

## _SYS_AFL.PAL_UNIFIED_MASSIVE_REGRESSION_PIVOT

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<METADATA Table>` | IN |
| 2 | `<DATA Table>` | IN |
| 3 | `<PARAMETER Table>` | IN |
| 4 | `<MODEL Table>` | OUT |
| 5 | `<STATISTICS Table>` | OUT |
| 6 | `<OPTPARAMETER Table>` | OUT |
| 7 | `<PARTITION_TYPE Table>` | OUT |
| 8 | `<ERRORMSG Table>` | OUT |
| 9 | `<PLACE_HOLDER1 Table>` | OUT |
| 10 | `<PLACE_HOLDER2 Table>` | OUT |

### Input Table

| Table | Column | Data Type | Description |
|---|---|---|---|
| METADATA | 1st column | VARCHAR or NVARCHAR | The name of the variable. |

| Table | Column | Data Type | Description |
|---|---|---|---|
| | 2nd column | VARCHAR or NVARCHAR | The type of the variable:<br>• CONTINUOUS;<br>• CATEGORICAL;<br>• TARGET. |
| DATA | 1st column | INTEGER, VRACHAR or NVARCHAR | Group id. |
| | 2nd column | INTEGER, VRACHAR or NVARCHAR | Record id. |
| | 3rd column | VARCHAR or NVARCHAR | Variable name. |
| | 4th column | VARCHAR or NVARCHAR | Variable value. |
| | 5th column | INTEGER | Self-defined data partition:<br>• 1:training<br>• 2:validation |
| PARAMETER | 1st column | INTEGER, VRACHAR or NVARCHAR | Group id. |
| | 2nd column | VARCHAR or NVARCHAR | Parameter name. |
| | 3rd column | INTEGER | Int value. |
| | 4th column | DOUBLE | Double value. |
| | 5th column | VARCHAR or NVARCHAR | String value. |

### Parameter Table

Mandatory Parameters

The following parameters are mandatory and must be given a value.

| Name | Data Type | Description |
|---|---|---|
| FUNCTION | VARCHAR or NVARCHAR | The algorithm name, including:<br>• GEO;<br>• LOG;<br>• EXP;<br>• GLM;<br>• MLR;<br>• POL;<br>• DT;<br>• HGBT;<br>• MLP;<br>• RDT;<br>• SVM. |

Optional Parameters

The parameter of the unified massive regression with pivot execution function is the same as the unified regression with pivot function.

See Parameter Tables [page 2030] for details about parameters for a specific algorithm.

### Output Tables

| Table | Column | Column Name | Column Data Type |
|---|---|---|---|
| MODEL | 1st column | GROUP_ID | Group id column data type. |
| | 2nd column | ROW_INDEX | INTEGER |
| | 3rd column | PART_INDEX | INTEGER |
| | 4th column | MODEL_CONTENT | VARCHAR, NVARCHAR, NCLOB |
| STATISTICS | 1st column | GROUP_ID | Group id column data type. |
| | 2nd column | STAT_NAME | VARCHAR or NVARCHAR |
| | 3rd column | STAT_VALUE | VARCHAR or NVARCHAR |
| OPTPARAMETER | 1st column | GROUP_ID | Group id column data type. |
| | 2nd column | PARAM_NAME | VARCHAR or NVARCHAR |
| | 3rd column | INT_VALUE | INTEGER |
| | 4th column | DOUBLE_VALUE | DOUBLE |
| | 5th column | STRING_VALUE | VARCHAR or NVARCHAR |
| PARTITION_TYPE | 1st column | GROUP_ID | Group id column data type. |
| | 2nd column | ID column name | ID column data type |
| | 3rd column | TYPE | INTEGER |
| ERRORMSG | 1st column | GROUP_ID | Group id column data type. |
| | 2nd column | ERROR_TIMESTAMP | VARCHAR, NVARCHAR |
| | 3rd column | ERRORCODE | INTEGER |
| | 4th column | MASSAGE | VARCHAR, NVARCHAR |
| PLCAE_HOLDER1 | 1st column | GROUP_ID | Group id column data type. |
| | 2nd column | OBJECT | VARCHAR, NVARCHAR |
| | 3rd column | KEY | VARCHAR, NVARCHAR |
| | 4th column | VALUE | VARCHAR, NVARCHAR |
| PLCAE_HOLDER2 | 1st column | GROUP_ID | Group id column data type. |
| | 2nd column | OBJECT | VARCHAR, NVARCHAR |
| | 3rd column | KEY | VARCHAR, NVARCHAR |
| | 4th column | VALUE | VARCHAR, NVARCHAR |

### Example

Assume that:

- DM_PAL is a schema belonging to USER1;

- USER1 has been assigned the AFL__SYS_AFL_AFLPAL_EXECUTE or AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION role.

Example: Multiple Linear Regression

```
SET SCHEMA DM_PAL;
-----------------------------------Unified Massive Regression Pivot (Multiple
Linear Regression)--------------------------
DROP TABLE META_DATA_TAB;
CREATE COLUMN TABLE META_DATA_TAB(
    "VARIABLE_NAME" NVARCHAR(50),
    "VARIABLE_TYPE" NVARCHAR(50)
);
INSERT INTO META_DATA_TAB VALUES ('X1',    'CONTINUOUS');
INSERT INTO META_DATA_TAB VALUES ('Y' ,    'TARGET');
INSERT INTO META_DATA_TAB VALUES ('X2',    'CATEGORICAL');
INSERT INTO META_DATA_TAB VALUES ('X3',    'CONTINUOUS');
DROP TABLE PAL_DATA_TBL;
CREATE COLUMN TABLE PAL_DATA_TBL(
    "GROUP_ID" INTEGER,
    "ID" INTEGER,
    "VARIABLE_NAME" NVARCHAR(50),
    "VALUE" NVARCHAR(50),
    "PURPOSE" INTEGER
);
INSERT INTO PAL_DATA_TBL VALUES (1, 0,    'Y',    '-6.879',   1);
INSERT INTO PAL_DATA_TBL VALUES (1, 0,    'X1',   '0.00',     1);
INSERT INTO PAL_DATA_TBL VALUES (1, 0,    'X2',   'A',        1);
INSERT INTO PAL_DATA_TBL VALUES (1, 0,    'X3',   '1',        1);
INSERT INTO PAL_DATA_TBL VALUES (1, 1,    'Y',    '-3.449',   1);
INSERT INTO PAL_DATA_TBL VALUES (1, 1,    'X1',   '0.50',     1);
INSERT INTO PAL_DATA_TBL VALUES (1, 1,    'X2',   'A',        1);
INSERT INTO PAL_DATA_TBL VALUES (1, 1,    'X3',   '1',        1);
INSERT INTO PAL_DATA_TBL VALUES (1, 2,    'Y',    '6.635',    1);
INSERT INTO PAL_DATA_TBL VALUES (1, 2,    'X1',   '0.54',     1);
INSERT INTO PAL_DATA_TBL VALUES (1, 2,    'X2',   'B',        1);
INSERT INTO PAL_DATA_TBL VALUES (1, 2,    'X3',   '1',        1);
INSERT INTO PAL_DATA_TBL VALUES (1, 3,    'Y',    '11.844',   2);
INSERT INTO PAL_DATA_TBL VALUES (1, 3,    'X1',   '1.04',     2);
INSERT INTO PAL_DATA_TBL VALUES (1, 3,    'X2',   'B',        2);
INSERT INTO PAL_DATA_TBL VALUES (1, 3,    'X3',   '1',        2);
INSERT INTO PAL_DATA_TBL VALUES (1, 4,    'Y',    '2.786',    1);
INSERT INTO PAL_DATA_TBL VALUES (1, 4,    'X1',   '1.50',     1);
INSERT INTO PAL_DATA_TBL VALUES (1, 4,    'X2',   'A',        1);
INSERT INTO PAL_DATA_TBL VALUES (1, 4,    'X3',   '1',        1);
INSERT INTO PAL_DATA_TBL VALUES (1, 5,    'Y',    '2.389',    1);
INSERT INTO PAL_DATA_TBL VALUES (1, 5,    'X1',   '0.04',     1);
INSERT INTO PAL_DATA_TBL VALUES (1, 5,    'X2',   'B',        1);
INSERT INTO PAL_DATA_TBL VALUES (1, 5,    'X3',   '2',        1);
INSERT INTO PAL_DATA_TBL VALUES (1, 6,    'Y',    '-0.011',   2);
INSERT INTO PAL_DATA_TBL VALUES (1, 6,    'X1',   '2.00',     2);
INSERT INTO PAL_DATA_TBL VALUES (1, 6,    'X2',   'A',        2);
INSERT INTO PAL_DATA_TBL VALUES (1, 6,    'X3',   '2',        2);
INSERT INTO PAL_DATA_TBL VALUES (1, 7,    'Y',    '8.839',    1);
INSERT INTO PAL_DATA_TBL VALUES (1, 7,    'X1',   '2.04',     1);
INSERT INTO PAL_DATA_TBL VALUES (1, 7,    'X2',   'B',        1);
INSERT INTO PAL_DATA_TBL VALUES (1, 7,    'X3',   '2',        1);
INSERT INTO PAL_DATA_TBL VALUES (1, 8,    'Y',    '4.689',    1);
INSERT INTO PAL_DATA_TBL VALUES (1, 8,    'X1',   '1.54',     1);
INSERT INTO PAL_DATA_TBL VALUES (1, 8,    'X2',   'B',        1);
INSERT INTO PAL_DATA_TBL VALUES (1, 8,    'X3',   '1',        1);
INSERT INTO PAL_DATA_TBL VALUES (1, 9,    'Y',    '-5.507',   1);
INSERT INTO PAL_DATA_TBL VALUES (1, 9,    'X1',   '1.00',     1);
INSERT INTO PAL_DATA_TBL VALUES (1, 9,    'X2',   'A',        1);
INSERT INTO PAL_DATA_TBL VALUES (1, 9,    'X3',   '2',        1);
INSERT INTO PAL_DATA_TBL VALUES (2, 0,    'Y',    '6.879',   1);
INSERT INTO PAL_DATA_TBL VALUES (2, 0,    'X1',   '1.00',     1);
```

```
INSERT INTO PAL_DATA_TBL VALUES (2, 0,    'X2',   'B',        1);
INSERT INTO PAL_DATA_TBL VALUES (2, 0,    'X3',   '1',        1);
INSERT INTO PAL_DATA_TBL VALUES (2, 1,    'Y',    '3.449',   1);
INSERT INTO PAL_DATA_TBL VALUES (2, 1,    'X1',   '1.50',     1);
INSERT INTO PAL_DATA_TBL VALUES (2, 1,    'X2',   'B',        1);
INSERT INTO PAL_DATA_TBL VALUES (2, 1,    'X3',   '1',        1);
INSERT INTO PAL_DATA_TBL VALUES (2, 2,    'Y',    '0.635',    1);
INSERT INTO PAL_DATA_TBL VALUES (2, 2,    'X1',   '1.54',     1);
INSERT INTO PAL_DATA_TBL VALUES (2, 2,    'X2',   'A',        1);
INSERT INTO PAL_DATA_TBL VALUES (2, 2,    'X3',   '1',        1);
INSERT INTO PAL_DATA_TBL VALUES (2, 3,    'Y',    '1.844',   2);
INSERT INTO PAL_DATA_TBL VALUES (2, 3,    'X1',   '1.04',     2);
INSERT INTO PAL_DATA_TBL VALUES (2, 3,    'X2',   'A',        2);
INSERT INTO PAL_DATA_TBL VALUES (2, 3,    'X3',   '1',        2);
INSERT INTO PAL_DATA_TBL VALUES (2, 4,    'Y',    '10.786',    1);
INSERT INTO PAL_DATA_TBL VALUES (2, 4,    'X1',   '5.50',     1);
INSERT INTO PAL_DATA_TBL VALUES (2, 4,    'X2',   'A',        1);
INSERT INTO PAL_DATA_TBL VALUES (2, 4,    'X3',   '1',        1);
INSERT INTO PAL_DATA_TBL VALUES (2, 5,    'Y',    '22.389',    1);
INSERT INTO PAL_DATA_TBL VALUES (2, 5,    'X1',   '0.04',     1);
INSERT INTO PAL_DATA_TBL VALUES (2, 5,    'X2',   'A',        1);
INSERT INTO PAL_DATA_TBL VALUES (2, 5,    'X3',   '2',        1);
INSERT INTO PAL_DATA_TBL VALUES (2, 6,    'Y',    '-0.011',   2);
INSERT INTO PAL_DATA_TBL VALUES (2, 6,    'X1',   '10.00',     2);
INSERT INTO PAL_DATA_TBL VALUES (2, 6,    'X2',   'B',        2);
INSERT INTO PAL_DATA_TBL VALUES (2, 6,    'X3',   '2',        2);
INSERT INTO PAL_DATA_TBL VALUES (2, 7,    'Y',    '8.839',    1);
INSERT INTO PAL_DATA_TBL VALUES (2, 7,    'X1',   '2.04',     1);
INSERT INTO PAL_DATA_TBL VALUES (2, 7,    'X2',   'B',        1);
INSERT INTO PAL_DATA_TBL VALUES (2, 7,    'X3',   '2',        1);
INSERT INTO PAL_DATA_TBL VALUES (2, 8,    'Y',    '4.689',    1);
INSERT INTO PAL_DATA_TBL VALUES (2, 8,    'X1',   '1.54',     1);
INSERT INTO PAL_DATA_TBL VALUES (2, 8,    'X2',   'B',        1);
INSERT INTO PAL_DATA_TBL VALUES (2, 8,    'X3',   '1',        1);
INSERT INTO PAL_DATA_TBL VALUES (2, 9,    'Y',    '-5.507',   1);
INSERT INTO PAL_DATA_TBL VALUES (2, 9,    'X1',   '1.00',     1);
INSERT INTO PAL_DATA_TBL VALUES (2, 9,    'X2',   'A',        1);
INSERT INTO PAL_DATA_TBL VALUES (2, 9,    'X3',   '2',        1);
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL (
    "GROUP_ID" INTEGER,
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO PAL_PARAMETER_TBL VALUES (1, 'FUNCTION',null,null,'MLR');
INSERT INTO PAL_PARAMETER_TBL VALUES (1, 'PARTITION_METHOD',1,null,null);
INSERT INTO PAL_PARAMETER_TBL VALUES (1, 'OUTPUT_PARTITION_RESULT',1,null,null);
INSERT INTO PAL_PARAMETER_TBL VALUES (1, 'THREAD_RATIO', null,0.5,null);
INSERT INTO PAL_PARAMETER_TBL VALUES (1, 'BACKGROUND_SIZE', 4,null,null);
INSERT INTO PAL_PARAMETER_TBL VALUES (2, 'FUNCTION',null,null,'MLR');
INSERT INTO PAL_PARAMETER_TBL VALUES (2, 'PARTITION_METHOD',1,null,null);
INSERT INTO PAL_PARAMETER_TBL VALUES (2, 'OUTPUT_PARTITION_RESULT',1,null,null);
INSERT INTO PAL_PARAMETER_TBL VALUES (2, 'THREAD_RATIO', null,0.5,null);
INSERT INTO PAL_PARAMETER_TBL VALUES (2, 'BACKGROUND_SIZE', 4,null,null);
DROP TABLE PAL_DATA_MODEL_TBL;
CREATE COLUMN TABLE PAL_DATA_MODEL_TBL(
     "GROUP_ID" INTEGER,
     "ROW_INDEX" INTEGER,
     "PART_INDEX" INTEGER,
     "MODEL_CONTENT" VARCHAR (5000)
);
DROP TABLE PAL_DATA_STATISTICS_TBL;
CREATE COLUMN TABLE PAL_DATA_STATISTICS_TBL(
     "GROUP_ID" INTEGER,
     "STAT_NAME" NVARCHAR (100),
     "STAT_VALUE" NVARCHAR (200)
```

```
);
DROP TABLE PAL_DATA_OPTIMAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_DATA_OPTIMAL_PARAMETER_TBL(
     "GROUP_ID" INTEGER,
     "PARAM_NAME" NVARCHAR (100),
     "INT_VALUE" INTEGER,
     "DOUBLE_VALUE" DOUBLE,
     "STRING_VALUE" NVARCHAR (200)
);
DROP TABLE PAL_DATA_PARTITION_TYPE_TBL;
CREATE COLUMN TABLE PAL_DATA_PARTITION_TYPE_TBL(
     "GROUP_ID" INTEGER,
     "ID" INTEGER,
     "TYPE" INTEGER
);
DO BEGIN
  meta_data_tab = SELECT * FROM META_DATA_TAB;
  data_tab = SELECT * FROM PAL_DATA_TBL;
  control_tab = SELECT * FROM PAL_PARAMETER_TBL;
  CALL
_SYS_AFL.PAL_UNIFIED_MASSIVE_REGRESSION_PIVOT(:meta_data_tab, :data_tab, :control
_tab, model_tab, stats_tab, optimal_parameter_tab, partition_result, errormsg,
place_holder_1_tab, place_holder_2_tab);
  INSERT into PAL_DATA_MODEL_TBL SELECT * FROM :model_tab;
  INSERT into PAL_DATA_STATISTICS_TBL SELECT * FROM :stats_tab;
  INSERT into PAL_DATA_OPTIMAL_PARAMETER_TBL SELECT *
FROM :optimal_parameter_tab;
  INSERT into PAL_DATA_PARTITION_TYPE_TBL SELECT * FROM :partition_result;
  select * from :errormsg;
END;
select * from PAL_DATA_MODEL_TBL;
select * from PAL_DATA_STATISTICS_TBL;
select * from PAL_DATA_OPTIMAL_PARAMETER_TBL;
select * from PAL_DATA_PARTITION_TYPE_TBL;
```

**Expected Result**:

MODEL:

Rows (8)

| | GROUP_ID | ROW_INDEX | PART_INDEX | |
|---|---|---|---|---|
| 1 | 1 | 1 | NULL | <PMML version="4.0" xmlns="http://www.dmg.org/PMML-4_0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" > <Header copyright="SAP" > <Application name="F |
| 2 | 1 | -10 | NULL | {"cate_vals":[{"cateVal":["B","A","A","B"]}],"categorical_size":1,"cont_vals":[{"contVal":[0.54000000000000004,0.0,1.0,1.54]},{"contVal":[1.0,1.0,2.0,1.0]},{"contVal":[6.6349999999 |
| 3 | 1 | -1 | NULL | MLR |
| 4 | 1 | -2 | NULL | {"feature":["X1","X2","X3","Y"],"featureNum":4,"index":[0,1,2,3],"type":["CONTINUOUS","CATEGORICAL","CONTINUOUS","CONTINUOUS"]} |
| 5 | 2 | 1 | NULL | <PMML version="4.0" xmlns="http://www.dmg.org/PMML-4_0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" > <Header copyright="SAP" > <Application name="F |
| 6 | 2 | -10 | NULL | {"cate_vals":[{"cateVal":["A","B","A","B"]}],"categorical_size":1,"cont_vals":[{"contVal":[1.54,1.0,1.0,1.54]},{"contVal":[1.0,1.0,2.0,1.0]},{"contVal":[0.63500000000000001,6.87899 |
| 7 | 2 | -1 | NULL | MLR |
| 8 | 2 | -2 | NULL | {"feature":["X1","X2","X3","Y"],"featureNum":4,"index":[0,1,2,3],"type":["CONTINUOUS","CATEGORICAL","CONTINUOUS","CONTINUOUS"]} |

STATISTIC:

Rows (16)

| | GROUP_ID | STAT_NAME | STAT_VALUE |
|---|---|---|---|
| 1 | 1 | TEST_EVAR | 0.8267792311153285 |
| 2 | 1 | TEST_MAE | 2.8427835999999997 |
| 3 | 1 | TEST_MAPE | 17.30461011789628 |
| 4 | 1 | TEST_MAX_ERROR | 5.309797199999999 |
| 5 | 1 | TEST_MSE | 14.167574699013917 |
| 6 | 1 | TEST_R2 | 0.5967704177761926 |
| 7 | 1 | TEST_RMSE | 3.7639838866570505 |
| 8 | 1 | TEST_WMAPE | 0.48048400236626376 |
| 9 | 2 | TEST_EVAR | -31.838525093092272 |
| 10 | 2 | TEST_MAE | 8.06074986 |
| 11 | 2 | TEST_MAPE | 608.7345055270164 |
| 12 | 2 | TEST_MAX_ERROR | 13.3757801 |
| 13 | 2 | TEST_MSE | 93.22523475760448 |
| 14 | 2 | TEST_R2 | -107.3691455396046 |
| 15 | 2 | TEST_RMSE | 9.655321577120281 |
| 16 | 2 | TEST_WMAPE | 8.795144418985268 |

PARTITION_TYPE:

Rows (20)

| | GROUP_ID | ID | TYPE |
|---|---|---|---|
| 1 | 1 | 0 | 1 |
| 2 | 1 | 1 | 1 |
| 3 | 1 | 2 | 1 |
| 4 | 1 | 4 | 1 |
| 5 | 1 | 5 | 1 |
| 6 | 1 | 7 | 1 |
| 7 | 1 | 8 | 1 |
| 8 | 1 | 9 | 1 |
| 9 | 1 | 3 | 2 |
| 10 | 1 | 6 | 2 |
| 11 | 2 | 0 | 1 |
| 12 | 2 | 1 | 1 |
| 13 | 2 | 2 | 1 |
| 14 | 2 | 4 | 1 |
| 15 | 2 | 5 | 1 |
| 16 | 2 | 7 | 1 |
| 17 | 2 | 8 | 1 |
| 18 | 2 | 9 | 1 |
| 19 | 2 | 3 | 2 |
| 20 | 2 | 6 | 2 |

# _SYS_AFL.PAL_UNIFIED_MASSIVE_REGRESSION_PREDICT_PIVOT

## Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA Table>` | IN |
| 2 | `<MODEL Table>` | IN |
| 3 | `<PARAMETER Table>` | IN |
| 4 | `<RESULT Table>` | OUT |
| 5 | `<ERRORMSG Table>` | OUT |
| 6 | `<PLACE_HOLDER1 Table>` | OUT |

## Input Table

| Table | Column | Data Type | Description |
|---|---|---|---|
| DATA | 1st column | INTEGER, VRACHAR or NVARCHAR | Group id. |
| | 2nd column | INTEGER, VRACHAR or NVARCHAR | Record id. |
| | 3rd column | VARCHAR or NVARCHAR | Variable name. |
| | 4th column | VARCHAR or NVARCHAR | Variable value. |
| MODEL | 1st column | INTEGER, VRACHAR or NVARCHAR | Group id. |
| | 2nd column | INTEGER | Row index. |
| | 3rd column | INTEGER | Part index. |

| Table | Column | Data Type | Description |
|---|---|---|---|
| | 4th column | VARCHAR, NVARCHAR, or NCLOB | Model content. |
| PARAMETER | 1st column | INTEGER, VRACHAR or NVARCHAR | Group id. |
| | 2nd column | VARCHAR or NVARCHAR | Parameter name. |
| | 3rd column | INTEGER | Int value. |
| | 4th column | DOUBLE | Double value. |
| | 5th column | VARCHAR or NVARCHAR | String value. |

### Parameter Table

The parameter of the unified massive regression prediction with pivot execution function is the same as the unified regression prediction with pivot function.

See Parameter Tables [page 2030] for details about parameters for a specific algorithm.

### Output Tables

| Table | Column | Column Name | Column Data Type |
|---|---|---|---|
| RESULT | 1st column | GROUP_ID | Group id column data type. |
| | 2nd column | ID column name | ID column data type |
| | 3rd column | SCORE | DOUBLE |
| | 4th column | UPPER_BOUND | DOUBLE |
| | 5th column | LOWER_BOUND | DOUBLE |
| | 6th column | REASON | VARCHAR or NVARCHAR |
| ERRORMSG | 1st column | GROUP_ID | Group id column data type. |
| | 2nd column | ERROR_TIMESTAMP | VARCHAR, NVARCHAR |
| | 3rd column | ERRORCODE | INTEGER |
| | 4th column | MASSAGE | VARCHAR, NVARCHAR |
| PLCAE_HOLDER1 | 1st column | GROUP_ID | Group id column data type. |
| | 2nd column | OBJECT | VARCHAR, NVARCHAR |
| | 3rd column | KEY | VARCHAR, NVARCHAR |
| | 4th column | VALUE | VARCHAR, NVARCHAR |

### Example

Assume that:

- DM_PAL is a schema belonging to USER1;
- USER1 has been assigned the AFL__SYS_AFL_AFLPAL_EXECUTE or AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION role.

Example: Multiple Linear Regression Prediction

```
-----------------------------------------Unified Massive Predict
Pivot(Multiple Linear Regression)-------------------------------
DROP TABLE PAL_DATA_TBL_PRE;
CREATE COLUMN TABLE PAL_DATA_TBL_PRE(
    "GROUP_ID" INTEGER,
    "ID" INTEGER,
    "VARIABLE_NAME" NVARCHAR(50),
    "VALUE" NVARCHAR(50)
);
INSERT INTO PAL_DATA_TBL_PRE VALUES (1, 0,    'X1',    '1.69');
INSERT INTO PAL_DATA_TBL_PRE VALUES (1, 0,    'X2',    'B');
INSERT INTO PAL_DATA_TBL_PRE VALUES (1, 0,    'X3',    '1');
INSERT INTO PAL_DATA_TBL_PRE VALUES (1, 1,    'X1',    '0.054');
INSERT INTO PAL_DATA_TBL_PRE VALUES (1, 1,    'X2',    'B');
INSERT INTO PAL_DATA_TBL_PRE VALUES (1, 1,    'X3',    '2');
INSERT INTO PAL_DATA_TBL_PRE VALUES (1, 2,    'X1',    '0.123');
INSERT INTO PAL_DATA_TBL_PRE VALUES (1, 2,    'X2',    'A');
INSERT INTO PAL_DATA_TBL_PRE VALUES (1, 2,    'X3',    '2');
INSERT INTO PAL_DATA_TBL_PRE VALUES (1, 3,    'X1',    '1.980');
INSERT INTO PAL_DATA_TBL_PRE VALUES (1, 3,    'X2',    'A');
INSERT INTO PAL_DATA_TBL_PRE VALUES (1, 3,    'X3',    '1');
INSERT INTO PAL_DATA_TBL_PRE VALUES (1, 4,    'X1',    '0.563');
INSERT INTO PAL_DATA_TBL_PRE VALUES (1, 4,    'X2',    'A');
INSERT INTO PAL_DATA_TBL_PRE VALUES (1, 4,    'X3',    '1');
INSERT INTO PAL_DATA_TBL_PRE VALUES (2, 0,    'X1',    '10.69');
INSERT INTO PAL_DATA_TBL_PRE VALUES (2, 0,    'X2',    'A');
INSERT INTO PAL_DATA_TBL_PRE VALUES (2, 0,    'X3',    '2');
INSERT INTO PAL_DATA_TBL_PRE VALUES (2, 1,    'X1',    '1.054');
INSERT INTO PAL_DATA_TBL_PRE VALUES (2, 1,    'X2',    'B');
INSERT INTO PAL_DATA_TBL_PRE VALUES (2, 1,    'X3',    '20');
INSERT INTO PAL_DATA_TBL_PRE VALUES (2, 2,    'X1',    '10.123');
INSERT INTO PAL_DATA_TBL_PRE VALUES (2, 2,    'X2',    'A');
INSERT INTO PAL_DATA_TBL_PRE VALUES (2, 2,    'X3',    '2');
INSERT INTO PAL_DATA_TBL_PRE VALUES (2, 3,    'X1',    '1.980');
INSERT INTO PAL_DATA_TBL_PRE VALUES (2, 3,    'X2',    'A');
INSERT INTO PAL_DATA_TBL_PRE VALUES (2, 3,    'X3',    '1');
INSERT INTO PAL_DATA_TBL_PRE VALUES (2, 4,    'X1',    '0.563');
INSERT INTO PAL_DATA_TBL_PRE VALUES (2, 4,    'X2',    'A');
INSERT INTO PAL_DATA_TBL_PRE VALUES (2, 4,    'X3',    '1');
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "GROUP_ID" INTEGER,
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
CALL _SYS_AFL.PAL_UNIFIED_MASSIVE_REGRESSION_PREDICT_PIVOT(PAL_DATA_TBL_PRE,
PAL_DATA_MODEL_TBL, #PAL_PARAMETER_TBL, ?,?, ?);
```

**Expected Result**:

RESULT:

Rows (10)

| | GROUP_ID | ID | SCORE | UPPER_BOUND | LOWER_BOUND | REASON |
|---|---|---|---|---|---|---|
| 1 | 2 | 0 | 13.58969382 | NULL | NULL | [{"attr":"X3","val":20.85413623666895,"pct":58.9},{"attr":"X1","val":-13.7944677953491,"pct":38.9},{"attr":"X2","val":0.765229518680150 |
| 2 | 2 | 1 | 79.696532652 | NULL | NULL | [{"attr":"X3","val":71.31235192944217,"pct":96.5},{"attr":"X1","val":2.4443550790645478,"pct":3.3},{"attr":"X2","val":0.175029783493271 |
| 3 | 2 | 2 | 13.292706024 | NULL | NULL | [{"attr":"X3","val":19.74111535035817,"pct":59.0},{"attr":"X1","val":-12.960519083161385,"pct":38.800000000000007},{"attr":"X2","val":0 |
| 4 | 2 | 3 | 5.08208034 | NULL | NULL | [{"attr":"X1","val":-1.0958860574863239,"pct":68.69999999999999},{"attr":"X2","val":0.45671322163638469,"pct":28.6},{"attr":"X3","val" |
| 5 | 2 | 4 | 4.339872744 | NULL | NULL | [{"attr":"X3","val":-2.8251134000272719,"pct":66.89999999999999},{"attr":"X1","val":0.988250318474857,"pct":23.400000000000003},{ |
| 6 | 1 | 0 | 8.7284858 | NULL | NULL | [{"attr":"X1","val":3.7219618319512257,"pct":48.1},{"attr":"X2","val":2.723057057034578,"pct":35.2},{"attr":"X3","val":1.29351551101419 |
| 7 | 1 | 1 | 1.4132442800000002 | NULL | NULL | [{"attr":"X2","val":9.239541580054555,"pct":51.2},{"attr":"X1","val":-5.082508810370803,"pct":28.1},{"attr":"X3","val":-3.7337398896837 |
| 8 | 1 | 2 | -6.72318414 | NULL | NULL | [{"attr":"X1","val":-3.2109472378142005,"pct":41.6},{"attr":"X3","val":-3.185327466412240 6,"pct":41.300000000000007},{"attr":"X2","val |
| 9 | 1 | 3 | 1.3381136000000005 | NULL | NULL | [{"attr":"X2","val":-8.208507755580908,"pct":49.0},{"attr":"X1","val":6.588783011548036,"pct":39.3},{"attr":"X3","val":1.96788694403287 |
| 10 | 1 | 4 | -3.44542334 | NULL | NULL | [{"attr":"X2","val":-5.803054740885406,"pct":80.9},{"attr":"X3","val":1.1602674109481153,"pct":16.200000000000004},{"attr":"X1","val":0 |

## _SYS_AFL.PAL_UNIFIED_MASSIVE_REGRESSION_SCORE_PIVOT

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA Table>` | IN |
| 2 | `<MODEL Table>` | IN |
| 3 | `<PARAMETER Table>` | IN |
| 4 | `<RESULT Table>` | OUT |
| 5 | `<STATISTIC Table>` | OUT |
| 6 | `<ERRORMSG Table>` | OUT |

### Input Table

| Table | Column | Data Type | Description |
|---|---|---|---|
| DATA | 1st column | INTEGER, VRACHAR or NVARCHAR | Group id. |
| | 2nd column | INTEGER, VRACHAR or NVARCHAR | Record id. |
| | 3rd column | VARCHAR or NVARCHAR | Variable name. |
| | 4th column | VARCHAR or NVARCHAR | Variable value. |
| MODEL | 1st column | INTEGER, VRACHAR or NVARCHAR | Group id. |
| | 2nd column | INTEGER | Row index. |
| | 3rd column | INTEGER | Part index. |
| | 4th column | VARCHAR, NVARCHAR, or NCLOB | Model content. |
| PARAMETER | 1st column | INTEGER, VRACHAR or NVARCHAR | Group id. |
| | 2nd column | VARCHAR or NVARCHAR | Parameter name. |
| | 3rd column | INTEGER | Int value. |
| | 4th column | DOUBLE | Double value. |
| | 5th column | VARCHAR or NVARCHAR | String value. |

### Parameter Table

The parameter of the unified massive regression scoring with pivot execution function is the same as the unified regression scoring with pivot function.

See Parameter Tables [page 2030] for details about parameters for a specific algorithm.

### Output Tables

| Table | Column | Column Name | Column Data Type |
|---|---|---|---|
| RESULT | 1st column | GROUP_ID | Group id column data type. |
| | 2nd column | ID column name | ID column data type |
| | 3rd column | SCORE | DOUBLE |
| | 4th column | UPPER_BOUND | DOUBLE |
| | 5th column | LOWER_BOUND | DOUBLE |
| | 6th column | REASON | VARCHAR or NVARCHAR |
| STATISTICS | 1st column | GROUP_ID | Group id column data type. |
| | 2nd column | STAT_NAME | NVARCHAR |
| | 3rd column | STAT_VALUE | NVARCHAR |
| ERRORMSG | 1st column | GROUP_ID | Group id column data type. |
| | 2nd column | ERROR_TIMESTAMP | VARCHAR, NVARCHAR |
| | 3rd column | ERRORCODE | INTEGER |
| | 4th column | MASSAGE | VARCHAR, NVARCHAR |

### Example

Assume that:

- DM_PAL is a schema belonging to USER1;
- USER1 has been assigned the AFL__SYS_AFL_AFLPAL_EXECUTE or AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION role.

Example: Multiple Linear Regression Scoring

```
-------------------------------------------Unified Massive Scoring
Pivot(Multiple Linear Regression)-------------------------------
DROP TABLE PAL_DATA_TBL_SCORE;
CREATE COLUMN TABLE PAL_DATA_TBL_SCORE(
    "GROUP_ID" INTEGER,
    "ID" INTEGER,
    "VARIABLE_NAME" NVARCHAR(50),
    "VALUE" NVARCHAR(50)
);
INSERT INTO PAL_DATA_TBL_SCORE VALUES (1, 0,    'Y',    '4');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (1, 0,    'X1',    '1.69');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (1, 0,    'X2',    'B');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (1, 0,    'X3',    '1');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (1, 1,    'Y',    '3');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (1, 1,    'X1',    '0.054');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (1, 1,    'X2',    'B');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (1, 1,    'X3',    '2');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (1, 2,    'Y',    '7');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (1, 2,    'X1',    '0.123');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (1, 2,    'X2',    'A');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (1, 2,    'X3',    '2');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (1, 3,    'Y',    '9');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (1, 3,    'X1',    '1.980');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (1, 3,    'X2',    'A');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (1, 3,    'X3',    '1');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (1, 4,    'Y',    '0.5');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (1, 4,    'X1',    '0.563');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (1, 4,    'X2',    'A');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (1, 4,    'X3',    '1');
```

```
INSERT INTO PAL_DATA_TBL_SCORE VALUES (2, 0,    'Y',    '8');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (2, 0,    'X1',    '10.69');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (2, 0,    'X2',    'A');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (2, 0,    'X3',    '1');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (2, 1,    'Y',    '30');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (2, 1,    'X1',    '10.054');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (2, 1,    'X2',    'A');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (2, 1,    'X3',    '5');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (2, 2,    'Y',    '10');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (2, 2,    'X1',    '10.123');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (2, 2,    'X2',    'B');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (2, 2,    'X3',    '2');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (2, 3,    'Y',    '10');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (2, 3,    'X1',    '10.980');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (2, 3,    'X2',    'A');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (2, 3,    'X3',    '1');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (2, 4,    'Y',    '0.5');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (2, 4,    'X1',    '0.563');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (2, 4,    'X2',    'A');
INSERT INTO PAL_DATA_TBL_SCORE VALUES (2, 4,    'X3',    '1');
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "GROUP_ID" INTEGER,
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
CALL _SYS_AFL.PAL_UNIFIED_MASSIVE_REGRESSION_SCORE_PIVOT(PAL_DATA_TBL_SCORE,
PAL_DATA_MODEL_TBL, #PAL_PARAMETER_TBL, ?,?, ?);
```

**Expected Result**:

RESULT:

Rows (10)

| | GROUP_ID | ID | SCORE | UPPER_BOUND | LOWER_BOUND | REASON |
|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 8.7284858 | NULL | NULL | [{"attr":"X1","val":3.6599990836695035,"pct":47.300000000000007},{"attr":"X2","val":3.498968151681714,"pct":45.2},{"attr":"X3","val":0. |
| 2 | 1 | 1 | 1.4132442800000002 | NULL | NULL | [{"attr":"X2","val":7.338268478632669,"pct":51.5},{"attr":"X1","val":-5.135187240510666,"pct":36.0},{"attr":"X3","val":-1.7797883581220 |
| 3 | 1 | 2 | -6.72318414 | NULL | NULL | [{"attr":"X2","val":-3.308432167741501,"pct":42.9},{"attr":"X1","val":-3.1750753130900435,"pct":41.2},{"attr":"X3","val":-1.229628059168 |
| 4 | 1 | 3 | 1.3381136000000005 | NULL | NULL | [{"attr":"X2","val":-7.4979930205194889,"pct":48.9},{"attr":"X1","val":6.609145146855452,"pct":43.1},{"attr":"X3","val":1.2370100736640 |
| 5 | 1 | 4 | -3.44542334 | NULL | NULL | [{"attr":"X2","val":-5.252204943157921,"pct":86.5},{"attr":"X3","val":0.549139220915436,"pct":9.1},{"attr":"X1","val":0.267690982242485 |
| 6 | 2 | 0 | 9.64427382 | NULL | NULL | [{"attr":"X3","val":16.88486030192893,"pct":53.4},{"attr":"X1","val":-13.861427325236804,"pct":43.9},{"attr":"X2","val":0.8560449833078 |
| 7 | 2 | 1 | 25.092824652 | NULL | NULL | [{"attr":"X3","val":30.89166231691008,"pct":69.8},{"attr":"X1","val":-12.457385953061907,"pct":28.200000000000004},{"attr":"X2","val":0 |
| 8 | 2 | 2 | 13.429206023999999 | NULL | NULL | [{"attr":"X3","val":21.181887712559104,"pct":61.0},{"attr":"X1","val":-12.859441984569289,"pct":37.1},{"attr":"X2","val":-0.658035563989 |
| 9 | 2 | 3 | 9.79617234 | NULL | NULL | [{"attr":"X3","val":17.45871183992967,"pct":53.6},{"attr":"X1","val":-14.286434741097592,"pct":43.8},{"attr":"X2","val":0.8590993811679 |
| 10 | 2 | 4 | 4.339872744 | NULL | NULL | [{"attr":"X3","val":-3.1544311647380676,"pct":64.6},{"attr":"X1","val":0.9801247451845659,"pct":20.1},{"attr":"X2","val":0.749383303553 |

STATISTICS:

Rows (16)

| | GROUP_ID | STAT_NAME | STAT_VALUE |
|---|---|---|---|
| 1 | 1 | EVAR | -3.2188565928568202 |
| 2 | 1 | MAE | 6.32914708 |
| 3 | 1 | MAPE | 2.48273245831746 |
| 4 | 1 | MAX_ERROR | 13.72318414 |
| 5 | 1 | MSE | 57.49460463088472 |
| 6 | 1 | R2 | -5.416808552554099 |
| 7 | 1 | RMSE | 7.582519675601556 |
| 8 | 1 | WMAPE | 1.3466270382978724 |
| 9 | 2 | EVAR | 0.8948769149946142 |
| 10 | 2 | MAE | 2.8048711192 |
| 11 | 2 | MAPE | 1.6824311190999999 |
| 12 | 2 | MAX_ERROR | 4.907175347999999 |
| 13 | 2 | MSE | 10.665925730257872 |
| 14 | 2 | R2 | 0.8888502945992302 |
| 15 | 2 | RMSE | 3.2658728894826683 |
| 16 | 2 | WMAPE | 0.23973257429059827 |

## 5.13.13 Unified Exponential Smoothing

Unified exponential smoothing is a unified interface to call some exponential smoothing algorithms with unpivot data format.

The exponential smoothing algorithms include:

- SESM (Single Exponential Smoothing)
- DESM (Double Exponential Smoothing)
- TESM (Triple Exponential Smoothing)
- BESM (Brown Exponential Smoothing)
- AESM (Auto Exponential Smoothing)
- MSESM (Massive Single Exponential Smoothing)
- MDESM (Massive Double Exponential Smoothing)
- MTESM (Massive Triple Exponential Smoothing)
- MBESM (Massive Brwon Exponential Smoothing)
- MAESM (Massive Auto Exponential Smoothing)

The procedure provides two different mode: single mode and massive mode.

- Single mode: Processing a single dataset and parameter;
- Massive mode: Processing multiple datasets and parameters by using an additional column(Group_id).

See the corresponding topic for details about a specific exponential smoothing method.

### Prerequisites

- All prerequisites for the original corresponding exponential smoothing function apply in the unified exponential smoothing interface.

### _SYS_AFL.PAL_UNIFIED_EXPONENTIALSMOOTHING

#### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | <DATA Table> | IN |
| 2 | <PARAMETER Table> | IN |
| 3 | <FORECAST Table> | OUT |
| 4 | <STATISTICS Table> | OUT |
| 5 | <ERRORMsgTBL Table> | OUT |
| 6 | <PLACE_HOLDER_1 Table> | OUT |

| Position | Table Name | Parameter Type |
|---|---|---|
| 7 | <PLACE_HOLDER_2 Table> | OUT |

Input Table

| Table | Column | Data Type | Description |
|---|---|---|---|
| DATA | 1st column(Optional) | INTEGER, VRACHAR or NVARCHAR | Group ID. Only valid for massive mode. |
| | 2nd column | INTEGER, VRACHAR or NVARCHAR | Record ID. |
| | 3rd column | DOUBLE | Raw data. |
| PARAMETER | 1st column(Optional) | INTEGER, VRACHAR or NVARCHAR | Group ID. Only valid for massive mode. |
| | 2nd column | VARCHAR or NVARCHAR | parameter name. |
| | 3rd column | INTEGER | int value. |
| | 4th column | DOUBLE | double value. |
| | 5th column | VARCHAR or NVARCHAR | string value. |

### Parameter Table

Mandatory Parameters

The following parameters are mandatory and must be given a value.

| Name | Data Type | Description |
|---|---|---|
| FUNCTION | VARCHAR or NVARCHAR | The algorithm name, including:<br>• SESM;<br>• DESM;<br>• TESM;<br>• BESM;<br>• AESM;<br>• MSESM;<br>• MDESM;<br>• MTESM;<br>• MBESM;<br>• MAESM; |

Optional Parameters

See Parameter Tables for details about parameters for a specific algorithm.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| FORECAST_NUM | INTEGER | 0 | Number of values to be forecast. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| PREDICTION_CONFIDENCE_1 | DOUBLE | 0.8 | Prediction confidence for interval 1 | Only valid when the upper and lower columns are provided in the result table. |
| PREDICTION_CONFIDENCE_2 | DOUBLE | 0.95 | Prediction confidence for interval 2 | Only valid when the upper and lower columns are provided in the result table. |

Output Tables

| Table | Column | Column Name | Column Data Type |
|---|---|---|---|
| FORECAST | 1st column | GROUP_ID | VARCHAR |
| | 2nd column | TIMESTAMP | INTEGER |
| | 3rd column | VALUE | DOUBLE |
| | 4th column | PI1_LOWER | DOUBLE |
| | 5th column | PI1_UPPER | DOUBLE |
| | 6th column | PI2_LOWER | DOUBLE |
| | 7th column | PI2_UPPER | DOUBLE |
| | 8th column | REASON<br>**Note**<br>Reason code is not supported now. | VARCHAR |
| STATISTICS | 1st column | GROUP_ID | VARCHAR |
| | 2nd column | STAT_NAME | VARCHAR, NVARCHAR |
| | 3rd column | STAT_VALUE | VARCHAR, NVARCHAR |
| ERRORMsgTBL | 1st column | GROUP_ID | VARCHAR |
| | 2nd column | ERROR_TIMESTAMP | VARCHAR, NVARCHAR |
| | 3rd column | ERRORCODE | INTEGER |
| | 4th column | MASSAGE | VARCHAR, NVARCHAR |
| PLCAE_HOLDER_1 | 1st column | OBJECT | VARCHAR, NVARCHAR |
| | 2nd column | KEY | VARCHAR, NVARCHAR |
| | 3rd column | VALUE | VARCHAR, NVARCHAR |
| PLACE_HOLDER_2 | 1st column | OJBECT | VARCHAR, NVARCHAR |
| | 2nd column | KEY | VARCHAR, NVARCHAR |
| | 3rd column | VALUE | VARCHAR, NVARCHAR |

Example

Assume that:

- DM_PAL is a schema belonging to USER1;
- USER1 has been assigned the AFL__SYS_AFL_AFLPAL_EXECUTE or AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION role.

Example1: Single Exponential Smoothing

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_DATA_TBL;
CREATE COLUMN TABLE PAL_DATA_TBL(
    "ID" INTEGER,
    "VALUE" DOUBLE
);
INSERT INTO PAL_DATA_TBL VALUES (0,    200.0);
INSERT INTO PAL_DATA_TBL VALUES (1,    135.0);
INSERT INTO PAL_DATA_TBL VALUES (2,    195.0);
INSERT INTO PAL_DATA_TBL VALUES (3,    197.5);
INSERT INTO PAL_DATA_TBL VALUES (4,    310.0);
INSERT INTO PAL_DATA_TBL VALUES (5,    175.0);
INSERT INTO PAL_DATA_TBL VALUES (6,    155.0);
INSERT INTO PAL_DATA_TBL VALUES (7,    130.0);
INSERT INTO PAL_DATA_TBL VALUES (8,    220.0);
INSERT INTO PAL_DATA_TBL VALUES (9,    277.5);
INSERT INTO PAL_DATA_TBL VALUES (10,   235.0);
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO PAL_PARAMETER_TBL VALUES ('FUNCTION',null,null,'SESM');
INSERT INTO PAL_PARAMETER_TBL VALUES ('ADAPTIVE_METHOD',0,null,null);
INSERT INTO PAL_PARAMETER_TBL VALUES ('MEASURE_NAME',null,null,'MSE');
INSERT INTO PAL_PARAMETER_TBL VALUES ('ALPHA', null,0.1,null);
INSERT INTO PAL_PARAMETER_TBL VALUES ('FORECAST_NUM', 12,null,null);
INSERT INTO PAL_PARAMETER_TBL VALUES ('EXPOST_FLAG', 1,null,null);
CALL _SYS_AFL.PAL_UNIFIED_EXPONENTIALSMOOTHING(PAL_DATA_TBL,
PAL_PARAMETER_TBL, ?, ?, ?, ?, ?);
```

**Expected Result**:

FORECAST:

Rows (22)

| | GROUP_ID | TIMESTAMP | VALUE | PI1_LOWER | PI1_UPPER | PI2_LOWER | PI2_UPPER | REASON |
|---|---|---|---|---|---|---|---|---|
| 1 | NULL | 2 | 200 | NULL | NULL | NULL | NULL | NULL |
| 2 | NULL | 3 | 193.5 | NULL | NULL | NULL | NULL | NULL |
| 3 | NULL | 4 | 193.65 | NULL | NULL | NULL | NULL | NULL |
| 4 | NULL | 5 | 194.035 | NULL | NULL | NULL | NULL | NULL |
| 5 | NULL | 6 | 205.6315 | NULL | NULL | NULL | NULL | NULL |
| 6 | NULL | 7 | 202.56834999999998 | NULL | NULL | NULL | NULL | NULL |
| 7 | NULL | 8 | 197.81151499999999 | NULL | NULL | NULL | NULL | NULL |
| 8 | NULL | 9 | 191.0303635 | NULL | NULL | NULL | NULL | NULL |
| 9 | NULL | 10 | 193.92732715 | NULL | NULL | NULL | NULL | NULL |
| 10 | NULL | 11 | 202.284594435 | NULL | NULL | NULL | NULL | NULL |
| 11 | NULL | 12 | 205.55613499150002 | 126.700940203298 | 284.41132977970204 | 84.9575216121871 | 326.15474837081297 | NULL |
| 12 | NULL | 13 | 205.55613499150002 | 126.30764502143097 | 284.80462496156906 | 84.35602853732595 | 326.75624144567405 | NULL |
| 13 | NULL | 14 | 205.55613499150002 | 125.91629207360504 | 285.19597790939497 | 83.75750585310104 | 327.35476412989897 | NULL |
| 14 | NULL | 15 | 205.55613499150002 | 125.52685286653129 | 285.58541711646876 | 83.16190998278736 | 327.95036000021264 | NULL |
| 15 | NULL | 16 | 205.55613499150002 | 125.1392995968596 | 285.97297038614045 | 82.56919840483033 | 328.5430715781697 | NULL |
| 16 | NULL | 17 | 205.55613499150002 | 124.75360512801403 | 286.358664854986 | 81.97932961741841 | 329.1329403655816 | NULL |
| 17 | NULL | 18 | 205.55613499150002 | 124.36974296801866 | 286.74252701498136 | 81.39226310457079 | 329.7200068784292 | NULL |
| 18 | NULL | 19 | 205.55613499150002 | 123.98768724826262 | 287.12458273473743 | 80.80795930366119 | 330.30431067933887 | NULL |
| 19 | NULL | 20 | 205.55613499150002 | 123.60741270315617 | 287.50485727984386 | 80.22637957430457 | 330.88589040869545 | NULL |
| 20 | NULL | 21 | 205.55613499150002 | 123.22889465063255 | 287.88337533236745 | 79.64748616853727 | 331.46478381446275 | NULL |
| 21 | NULL | 22 | 205.55613499150002 | 122.85210897345293 | 288.2601610095471 | 79.07124220222539 | 332.04102778077464 | NULL |
| 22 | NULL | 23 | 205.55613499150002 | 122.4770321012746 | 288.6352378817254 | 78.49761162764055 | 332.6146583553595 | NULL |

STATISTIC:

Rows (1)

| | GROUP_ID | STAT_NAME | STAT_VALUE |
|---|---|---|---|
| 1 | *NULL* | MSE | 3438.3321253085405 |

Example2: Massive Single Exponential Smoothing

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_DATA_TBL;
CREATE COLUMN TABLE PAL_DATA_TBL(
    "GROUP_ID" NVARCHAR(50),
    "ID" INTEGER,
    "VALUE" DOUBLE
);
INSERT INTO PAL_DATA_TBL VALUES ('Hello1',  0,  200.0);
INSERT INTO PAL_DATA_TBL VALUES ('Hello1',  1,  135.0);
INSERT INTO PAL_DATA_TBL VALUES ('Hello1',  2,  195.0);
INSERT INTO PAL_DATA_TBL VALUES ('Hello1',  3,  197.5);
INSERT INTO PAL_DATA_TBL VALUES ('Hello1',  4,  310.0);
INSERT INTO PAL_DATA_TBL VALUES ('Hello1',  5,  175.0);
INSERT INTO PAL_DATA_TBL VALUES ('Hello1',  6,  155.0);
INSERT INTO PAL_DATA_TBL VALUES ('Hello1',  7,  130.0);
INSERT INTO PAL_DATA_TBL VALUES ('Hello1',  8,  220.0);
INSERT INTO PAL_DATA_TBL VALUES ('Hello1',  9,  277.5);
INSERT INTO PAL_DATA_TBL VALUES ('Hello1',  10, 235.0);
--full zero
INSERT INTO PAL_DATA_TBL VALUES ('Hello2',  1,    0.0);
INSERT INTO PAL_DATA_TBL VALUES ('Hello2',  2,    0.0);
INSERT INTO PAL_DATA_TBL VALUES ('Hello2',  3,    0.0);
INSERT INTO PAL_DATA_TBL VALUES ('Hello2',  4,    0.0);
INSERT INTO PAL_DATA_TBL VALUES ('Hello2',  5,    0.0);
INSERT INTO PAL_DATA_TBL VALUES ('Hello2',  6,    0.0);
INSERT INTO PAL_DATA_TBL VALUES ('Hello2',  7,    0.0);
INSERT INTO PAL_DATA_TBL VALUES ('Hello2',  8,    0.0);
INSERT INTO PAL_DATA_TBL VALUES ('Hello2',  9,    0.0);
INSERT INTO PAL_DATA_TBL VALUES ('Hello2',  10,    0.0);
INSERT INTO PAL_DATA_TBL VALUES ('Hello2',  11,    0.0);
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL (
    "GROUP_ID" VARCHAR (50),
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO PAL_PARAMETER_TBL VALUES (null, 'FUNCTION',null,null,'MSESM');
INSERT INTO PAL_PARAMETER_TBL VALUES ('Hello1', 'ADAPTIVE_METHOD',0,null,null);
INSERT INTO PAL_PARAMETER_TBL VALUES ('Hello1', 'MEASURE_NAME',null,null,'MSE');
INSERT INTO PAL_PARAMETER_TBL VALUES ('Hello1', 'ALPHA', null,0.1,null);
INSERT INTO PAL_PARAMETER_TBL VALUES ('Hello1', 'FORECAST_NUM', 12,null,null);
INSERT INTO PAL_PARAMETER_TBL VALUES ('Hello1', 'EXPOST_FLAG', 1,null,null);
INSERT INTO PAL_PARAMETER_TBL VALUES ('Hello2', 'ADAPTIVE_METHOD',1,null,null);
INSERT INTO PAL_PARAMETER_TBL VALUES ('Hello2',
'MEASURE_NAME',null,null,'MAPE');
INSERT INTO PAL_PARAMETER_TBL VALUES ('Hello2', 'ALPHA', null,0.5,null);
INSERT INTO PAL_PARAMETER_TBL VALUES ('Hello2', 'FORECAST_NUM', 12,null,null);
INSERT INTO PAL_PARAMETER_TBL VALUES ('Hello2', 'EXPOST_FLAG', 0,null,null);
CALL _SYS_AFL.PAL_UNIFIED_EXPONENTIALSMOOTHING(PAL_DATA_TBL,
PAL_PARAMETER_TBL, ?, ?, ?, ?, ?);
```

**Expected Result**:

FORECAST:

Rows (34)

| | GROUP_ID | TIMESTAMP | VALUE | PI1_LOWER | PI1_UPPER | PI2_LOWER | PI2_UPPER | REASON |
|---|---|---|---|---|---|---|---|---|
| 5 | Hello2 | 16 | 0 | 0 | 0 | 0 | 0 | NULL |
| 6 | Hello2 | 17 | 0 | 0 | 0 | 0 | 0 | NULL |
| 7 | Hello2 | 18 | 0 | 0 | 0 | 0 | 0 | NULL |
| 8 | Hello2 | 19 | 0 | 0 | 0 | 0 | 0 | NULL |
| 9 | Hello2 | 20 | 0 | 0 | 0 | 0 | 0 | NULL |
| 10 | Hello2 | 21 | 0 | 0 | 0 | 0 | 0 | NULL |
| 11 | Hello2 | 22 | 0 | 0 | 0 | 0 | 0 | NULL |
| 12 | Hello2 | 23 | 0 | 0 | 0 | 0 | 0 | NULL |
| 13 | Hello1 | 2 | 200 | NULL | NULL | NULL | NULL | NULL |
| 14 | Hello1 | 3 | 193.5 | NULL | NULL | NULL | NULL | NULL |
| 15 | Hello1 | 4 | 193.65 | NULL | NULL | NULL | NULL | NULL |
| 16 | Hello1 | 5 | 194.035 | NULL | NULL | NULL | NULL | NULL |
| 17 | Hello1 | 6 | 205.6315 | NULL | NULL | NULL | NULL | NULL |
| 18 | Hello1 | 7 | 202.56834999999998 | NULL | NULL | NULL | NULL | NULL |
| 19 | Hello1 | 8 | 197.81151499999999 | NULL | NULL | NULL | NULL | NULL |
| 20 | Hello1 | 9 | 191.0303635 | NULL | NULL | NULL | NULL | NULL |
| 21 | Hello1 | 10 | 193.92732715 | NULL | NULL | NULL | NULL | NULL |
| 22 | Hello1 | 11 | 202.284594435 | NULL | NULL | NULL | NULL | NULL |
| 23 | Hello1 | 12 | 205.55613499150002 | 126.700940203298 | 284.41132977970204 | 84.9575216121871 | 326.15474837081297 | NULL |
| 24 | Hello1 | 13 | 205.55613499150002 | 126.30764502143097 | 284.80462496156906 | 84.35602853732595 | 326.75624144567405 | NULL |
| 25 | Hello1 | 14 | 205.55613499150002 | 125.91629207360504 | 285.19597790939497 | 83.75750585310104 | 327.35476412989897 | NULL |
| 26 | Hello1 | 15 | 205.55613499150002 | 125.52685286653129 | 285.58541711646876 | 83.16190998278736 | 327.95036000021264 | NULL |
| 27 | Hello1 | 16 | 205.55613499150002 | 125.1392995968596 | 285.97297038614045 | 82.56919840483033 | 328.5430715781697 | NULL |
| 28 | Hello1 | 17 | 205.55613499150002 | 124.75360512801403 | 286.358664854986 | 81.97932961741841 | 329.1329403655816 | NULL |

STATISTIC:

Rows (2)

| | GROUP_ID | STAT_NAME | STAT_VALUE |
|---|---|---|---|
| 1 | Hello2 | MAPE | 0 |
| 2 | Hello1 | MSE | 3438.3321253085405 |

# 5.13.13.1 Parameter Tables

This topic lists all parameters for unified exponential smoothing.

## _SYS_AFL.PAL_UNIFIED_EXPONENTIALSMOOTHING

### UNIFIED EXPONENTIAL SMOOTHING

Mandatory parameters

| Name | Data Type | Description |
| --- | --- | --- |
| FUNCTION | VARCHAR or NVARCHAR | The algorithm name, including:<br>• SESM;<br>• DESM;<br>• TESM;<br>• BESM;<br>• AESM;<br>• MSESM;<br>• MDESM;<br>• MTESM;<br>• MBESM;<br>• MAESM; |

Optional parameters

| Name | Data Type | Default Value | Description | Dependency |
| --- | --- | --- | --- | --- |
| FORECAST_NUM | INTEGER | 0 | Number of values to be forecast. | |
| PREDICTION_CONFIDENCE_1 | DOUBLE | 0.8 | Prediction confidence for interval 1 | Only valid when the upper and lower columns are provided in the result table. |
| PREDICTION_CONFIDENCE_2 | DOUBLE | 0.95 | Prediction confidence for interval 2 | Only valid when the upper and lower columns are provided in the result table. |
| DECOM_STATE | INTEGER | 0 | Specifies whether or not to decompose the forecast value.<br>• 0: non-decomposition<br>• 1: decomposition | |

### SESM & MSESM

Mandatory parameters

None

Optional parameters

| Name | Data Type | Default Value | Description | Denpendency |
|---|---|---|---|---|
| ALPHA | DOUBLE | 0.1 (Single exponential smoothing) 0.2 (Adaptive-response-rate single exponential smoothing) | The smoothing constant alpha for single exponential smoothing or the initialization value for adaptive-response-rate single exponential smoothing ($0 < \alpha < 1$). | |
| DELTA | DOUBLE | 0.2 | Value of weighted for At and Mt. | Only valid when ADAPTIVE_METHOD is 1. |
| ADAPTIVE_METHOD | INTEGER | 0 | • 0: Single exponential smoothing<br>• 1: Adaptive-response-rate single exponential smoothing | |
| MEASURE_NAME | VARCHAR or NVARCHAR | No default value | Measure name. Supported measures are MPE, MSE, RMSE, ET, MAD, MASE, WMAPE, SMAPE, and MAPE. For detailed information on measures, see Forecast Accuracy Measures. | |
| IGNORE_ZERO | INTEGER | 0 | • 0: Uses zero values in the input dataset when calculating MPE or MAPE.<br>• 1: Ignores zero values in the input dataset when calculating MPE or MAPE. | Only valid when MEASURE_NAME is MPE or MAPE. |
| EXPOST_FLAG | INTEGER | 1 | • 0: Does not output the expost forecast, and just outputs the forecast values.<br>• 1: Outputs the expost forecast and the forecast values. | |

DESM & MDESM

Mandatory parameters

None

Optional parameters

| Name | Data Type | Default Value | Description | Denpendency |
|---|---|---|---|---|
| ALPHA | DOUBLE | 0.1 | Value of the smoothing constant alpha ($0 < \alpha < 1$). | |
| BETA | DOUBLE | 0.1 | Value of the smoothing constant beta ($0 < \beta < 1$). | |
| PHI | DOUBLE | 0.1 | Value of the damped smoothing constant $\Phi$ ($0 < \Phi < 1$). | |
| DAMPED | INTEGER | 0 | • 0: Uses the Holt's linear method.<br>• 1: Uses the additive damped trend Holt's linear method. | |
| MEASURE_NAME | VARCHAR or NVARCHAR | No default value | Measure name. Supported measures are MPE, MSE, RMSE, ET, MAD, MASE, WMAPE, SMAPE, and MAPE. For detailed information on measures, see Forecast Accuracy Measures. | |
| IGNORE_ZERO | INTEGER | 0 | • 0: Uses zero values in the input dataset when calculating MPE or MAPE.<br>• 1: Ignores zero values in the input dataset when calculating MPE or MAPE. | Only valid when MEASURE_NAME is MPE or MAPE. |
| EXPOST_FLAG | INTEGER | 1 | • 0: Does not output the expost forecast, and just outputs the forecast values.<br>• 1: Outputs the expost forecast and the forecast values. | |

TESM & MTESM

Mandatory parameters

None

Optional parameters

| Name | Data Type | Default Value | Description | Denpendency |
|---|---|---|---|---|
| ALPHA | DOUBLE | 0.1 | Value of the smoothing constant alpha (0 < α < 1). | |
| BETA | DOUBLE | 0.1 | Value of the smoothing constant beta (0 < β < 1). | |
| GAMMA | DOUBLE | 0.1 | Value of the smoothing constant gamma ( 0 < γ < 1). | |
| CYCLE | INTEGER | 2 | A cycle of length L (L > 1). For example, quarterly data cycle is 4, and monthly data cycle is 12. | |
| SEASONAL | INTEGER | 2 | • 0: Multiplicative triple exponential smoothing<br>• 1: Additive triple exponential smoothing | When SEASONAL is set to 1, the default value of INITIAL_METHOD is 1; When SEASONAL is set to 0, the default value of INITIAL_METHOD is 0. |
| INITIAL_METHOD | INTEGER | 0 or 1 | Initialization method for trend and seasonal:<br>• 0: The first initialization method as the description part.<br>• 1: The second initialization method as the description part. | When SEASONAL is set to 1, the default value of INITIAL_METHOD is 1; When SEASONAL is set to 0, the default value of INITIAL_METHOD is 0. |
| PHI | DOUBLE | 0.1 | Value of the damped smoothing constant Φ (0 < Φ < 1). | |
| DAMPED | INTEGER | 0 | • 0: Uses the Holt's linear method.<br>• 1: Uses the additive damped trend Holt's linear method. | |

| Name | Data Type | Default Value | Description | Denpendency |
|---|---|---|---|---|
| MEASURE_NAME | VARCHAR or NVARCHAR | No default value | Measure name. Supported measures are MPE, MSE, RMSE, ET, MAD, MASE, WMAPE, SMAPE, and MAPE. For detailed information on measures, see Forecast Accuracy Measures. | |
| IGNORE_ZERO | INTEGER | 0 | • 0: Uses zero values in the input dataset when calculating MPE or MAPE.<br>• 1: Ignores zero values in the input dataset when calculating MPE or MAPE. | Only valid when MEASURE_NAME is MPE or MAPE. |
| EXPOST_FLAG | INTEGER | 1 | • 0: Does not output the expost forecast, and just outputs the forecast values.<br>• 1: Outputs the expost forecast and the forecast values. | |
| LEVEL_START | DOUBLE | No default value | The initial value for level component S. If this value is not provided, it will be calculated in the way as described earlier. Note: LEVEL_START cannot be zero. If it is set to zero, 0.0000000001 will be used instead. | |
| TREND_START | DOUBLE | No default value | The initial value for trend component B. If this value is not provided, it will be calculated in the way as described earlier. | |

| Name | Data Type | Default Value | Description | Denpendency |
|---|---|---|---|---|
| SEASON_START | INTEGER or DOUBLE | No default value | A list of initial values for seasonal component C. If this parameter is not used, the initial values of C will be calculated in the way as described earlier.Two values must be provided for each cycle: Cycle ID: An INTEGER which represents which cycle the initial value is used for. Initial value: A DOUBLE precision number which represents the initial value for the corresponding cycle. For example: To give the initial value 0.5 to the 3rd cycle, insert tuple ('SEASON_START', 3, 0.5, NULL) into the parameter table. Note: The initial values of all cycles must be provided if this parameter is used. | |

BESM & MBESM

Mandatory parameters

None

Optional parameters

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| ALPHA | DOUBLE | 0.1 (Brown exponential smoothing) 0.2 (Adaptive brown exponential smoothing) | The smoothing constant alpha for brown exponential smoothing or the initialization value for adaptive brown exponential smoothing ($0 < \alpha < 1$). | |
| DELTA | DOUBLE | 0.2 | Value of weighted for At and Mt. | Only valid when ADAPTIVE_METHOD is 1. |
| ADAPTIVE_METHOD | INTEGER | 0 | • 0: Brown exponential smoothing<br>• 1: Adaptive brown exponential smoothing | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| MEASURE_NAME | VARCHAR or NVARCHAR | No default value | Measure name. Supported measures are MPE, MSE, RMSE, ET, MAD, MASE, WMAPE, SMAPE, and MAPE. For detailed information on measures, see Forecast Accuracy Measures. | |
| IGNORE_ZERO | INTEGER | 0 | • 0: Uses zero values in the input dataset when calculating MPE or MAPE.<br>• 1: Ignores zero values in the input dataset when calculating MPE or MAPE. | Only valid when MEASURE_NAME is MPE or MAPE. |
| EXPOST_FLAG | INTEGER | 1 | • 0: Does not output the expost forecast, and just outputs the forecast values.<br>• 1: Outputs the expost forecast and the forecast values. | |

AESM & MAESM

Mandatory Parameters

None.

Optional Parameters

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| MODELSELECTION | INTEGER | 0 | When this is set to 1, the algorithm will select the best model among Single/Double/Triple/Damped Double/Damped Triple Exponential Smoothing models. If FORECAST_MODEL_NAME is set, the model defined by FORECAST_MODEL_NAME will be used. When this is set to 0, the algorithm will not perform the model selection. | |
| FORECAST_MODEL_NAME | VARCHAR or NVARCHAR | No default value | Name of the statistical model used for calculating the forecast.<br>• SESM: Single Exponential Smoothing<br>• DESM: Double Exponential Smoothing<br>• TESM: Triple Exponential Smoothing | This parameter must be set unless MODELSELECTION is set to 1. |
| OPTIMIZER_TIME_BUDGET | INTEGER | 1 | Time budget for Nelder-Mead optimization process. The time unit is second and the value should be larger than zero. | |
| MAX_ITERATION | INTEGER | 100 | Maximum number of iterations for simulated annealing. | |
| OPTIMIZER_RANDOM_SEED | INTEGER | System time | Random seed for simulated annealing. The value should be larger than zero. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| THREAD_RATIO | DOUBLE | -1.0 | Specifies the ratio of total number of threads that can be used by this function.<br>• 0: single thread<br>• 0–1: uses the specified percentage of available threads. PAL uses all available threads if the number is 1.0.<br>• -1: uses all of available threads. | |
| EXPOST_FLAG | INTEGER | 1 | • 0: Does not output the expost forecast, and just outputs the forecast values.<br>• 1: Outputs the expost forecast and the forecast values. | |
| ALPHA | DOUBLE | Computed automatically | Weight for smoothing. Value range: $0 < \alpha < 1$ | |
| BETA | DOUBLE | Computed automatically | Weight for the trend component. Value range: 0 Value range: $0 \leq \beta < 1$ Note: If it is not set, the optimized value will be computed automatically. | Only valid when the model is set by user or identified by the algorithm as a DESM/TESM. Value 0 is allowed under TESM model only. |
| GAMMA | DOUBLE | Computed automatically | Weight for the seasonal component. Value range: $0 < \gamma < 1$ | Only valid when the model is set by user or identified by the algorithm as TESM. |
| PHI | DOUBLE | Computed automatically | Value of the damped smoothing constant $\Phi$ ($0 < \Phi < 1$). | Only valid when the model is set by user or identified by the algorithm as a DAMPED model. |
| CYCLE | INTEGER | Computed automatically | Length of a cycle ($L > 1$). For example, the cycle of quarterly data is 4, and the cycle of monthly data is 12. | Only valid when the model is set by user or identified by the algorithm as a TESM. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| SEASONAL | INTEGER | If MODELSELECTION is set to 1, the default value will be computed automatically. Otherwise, the default value is 0. | • 0: Multiplicative triple exponential smoothing<br>• 1: Additive triple exponential smoothing | Only valid when the model is set by user or identified by the algorithm as a TESM. |
| INITIAL_METHOD | INTEGER | 0 or 1 | Initialization method for the trend and seasonal components. Refer to Triple Exponential Smoothing for detailed information on initialization method. | Only valid when the model is set by user or identified by the algorithm as a TESM. |
| TRAINING_RATIO | DOUBLE | 1.0 | The ratio of training data to the whole time series. Assuming the size of time series is N, and the training ratio is r, the first N*r time series is used to train, whereas only the latter N*(1-r) one is used to test. | If this parameter is set to 0.0 or 1.0, or the resulting training data (N*r) is less than 1 or equal to the size of time series, no train-and-test procedure is carried out. |
| DAMPED | INTEGER | If MODELSELECTION is set to 1, the default value will be computed automatically. Otherwise, the default value is 0. | For DESM:<br>• 0: Uses the Holt's linear method.<br>• 1: Uses the additive damped trend Holt's linear method.<br>For TESM:<br>• 0: Uses the Holt Winter method.<br>• 1: Uses the additive damped seasonal Holt Winter method. | |
| ACCURACY_MEASURE | VARCHAR Or NVARCHAR | MSE | The criterion used for the optimization. Available values are MSE and MAPE. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| SEASONALITY_CRITERION | DOUBLE | 0.5 | The criterion of the auto-correlation coefficient for accepting seasonality, in the range of (0, 1). The larger it is, the less probable a time series is regarded to be seasonal. | Only valid when FORECAST_MODEL_NAME is TESM or MODELSELECTION is set to 1, and CYCLE is not defined. |
| TREND_TEST_METHOD | INTEGER | 1 | The method to test trend:<br>• 1: MK test<br>• 2: Difference-sign test | Only valid when MODELSELECTION is set to 1. |
| TREND_TEST_ALPHA | DOUBLE | 0.05 | Tolerance probability for trend test. The value range is (0, 0.5). | Only valid when MODELSELECTION is set to 1. |
| ALPHA_MIN | DOUBLE | 0.0000000001 | Sets the minimum value of ALPHA. | Only valid when ALPHA is not defined. |
| BETA_MIN | DOUBLE | 0.0000000001 | Sets the minimum value of BETA. | Only valid when BETA is not defined. |
| GAMMA_MIN | DOUBLE | 0.0000000001 | Sets the minimum value of GAMMA. | Only valid when GAMMA is not defined. |
| PHI_MIN | DOUBLE | 0.0000000001 | Sets the minimum value of PHI. | Only valid when PHI is not defined. |
| ALPHA_MAX | DOUBLE | 1 | Sets the maximum value of ALPHA. | Only valid when ALPHA is not defined. |
| BETA_MAX | DOUBLE | 1 | Sets the maximum value of BETA. | Only valid when BETA is not defined. |
| GAMMA_MAX | DOUBLE | 1 | Sets the maximum value of GAMMA. | Only valid when GAMMA is not defined. |
| PHI_MAX | DOUBLE | 1 | Sets the maximum value of PHI. | Only valid when PHI is not defined. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| LEVEL_START | DOUBLE | No default value | The initial value for level component S. If this value is not provided, it will be calculated in the way as described in Triple Exponential Smoothing. Note: LEVEL_START cannot be zero. If it is set to zero, 0.0000000001 will be used instead. Refer to Triple Exponential Smoothing for detailed information | |
| TREND_START | DOUBLE | No default value | The initial value for trend component B. If this value is not provided, it will be calculated in the way as described in Triple Exponential Smoothing. Refer to Triple Exponential Smoothing for detailed information | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| SEASON_START | INTEGER or DOUBLE | No defalut value | A list of initial values for seasonal component C. If this parameter is not used, the initial values of C will be calculated in the way as described in Triple Exponential Smoothing.Two values must be provided for each cycle:<br>• Cycle ID: An INTEGER which represents which cycle the initial value is used for.<br>• Initial value: A DOUBLE precision number which represents the initial value for the corresponding cycle.<br>For example: To give the initial value 0.5 to the 3rd cycle, insert tuple ('SEASON_START', 3, 0.5, NULL) into the parameter table. Note: The initial values of all cycles must be provided if this parameter is used. Refer to Triple Exponential Smoothing for detailed information | |

## 5.13.14 Unified Exponential Smoothing with Pivoted Input Data

The unified exponential smoothing with pivoted is a unified interface to call some exponential smoothing algorithms with pivot data format.

The expoential smoothing algorithms include:

- SESM (Single Exponential Smoothing)
- DESM (Double Exponential Smoothing)
- TESM (Triple Exponential Smoothing)
- BESM (Brown Exponential Smoothing)

- AESM (Auto Exponential Smoothing)
- MSESM (Massive Single Exponential Smoothing)
- MDESM (Massive Double Exponential Smoothing)
- MTESM (Massive Triple Exponential Smoothing)
- MBESM (Massive Brwon Exponential Smoothing)
- MAESM (Massive Auto Exponential Smoothing)

The procedure provides two different mode: single mode and massive mode.

- Single mode: Processing a single dataset and parameter;
- Massive mode: Processing multiple datasets and parameters by using an additional column(Group_id).

See the corresponding topic for details about a specific exponential smoothing method.

## Prerequisites

- All prerequisites for the original corresponding exponential smoothing function apply in the unified exponential smoothing interface.

## _SYS_AFL.PAL_UNIFIED_EXPONENTIALSMOOTHING_PIVOT

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | <METADATA Table> | IN |
| 2 | <DATA Table> | IN |
| 3 | <PARAMETER Table> | IN |
| 4 | <FORECAST Table> | OUT |
| 5 | <STATISTICS Table> | OUT |
| 6 | <ERRORMsgTBL Table> | OUT |
| 7 | <PLACE_HOLDER_1 Table> | OUT |
| 8 | <PLACE_HOLDER_2 Table> | OUT |

Input Table

| Table | Column | Data Type | Description |
|---|---|---|---|
| METADATA | 1st column | VARCHAR or NVARCHAR | The name of the variable. |

| Table | Column | Data Type | Description |
|---|---|---|---|
| | 2nd column | VARCHAR or NVARCHAR | The type of the variable:<br>• CONTINUOUS;<br>• CATEGORICAL;<br>• TARGET.<br>Note: "GROUP_ID" must be defined first when using massive mode. |
| DATA | 1st column | INTEGER, VRACHAR or NVARCHAR | Record ID. |
| | 2nd column | VARCHAR or NVARCHAR | Variable name. |
| | 3rd column | VARCHAR or NVARCHAR | Variable value. |
| PARAMETER | 1st column(Optional) | INTEGER, VRACHAR or NVARCHAR | Group ID. Only valid for massive mode. |
| | 2nd column | VARCHAR or NVARCHAR | parameter name. |
| | 3rd column | INTEGER | int value. |
| | 4th column | DOUBLE | double value. |
| | 5th column | VARCHAR or NVARCHAR | string value. |

### Parameter Table

Mandatory Parameters

The following parameters are mandatory and must be given a value.

| Name | Data Type | Description |
|---|---|---|
| FUNCTION | VARCHAR or NVARCHAR | The algorithm name, including:<br>• SESM;<br>• DESM;<br>• TESM;<br>• BESM;<br>• AESM;<br>• MSESM;<br>• MDESM;<br>• MTESM;<br>• MBESM;<br>• MAESM; |

Optional Parameters

See Parameter Tables for details about parameters for a specific algorithm.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| FORECAST_NUM | INTEGER | 0 | Number of values to be forecast. | |
| PREDICTION_CONFIDENCE_1 | DOUBLE | 0.8 | Prediction confidence for interval 1 | Only valid when the upper and lower columns are provided in the result table. |
| PREDICTION_CONFIDENCE_2 | DOUBLE | 0.95 | Prediction confidence for interval 2 | Only valid when the upper and lower columns are provided in the result table. |

Output Tables

| Table | Column | Column Name | Column Data Type |
|---|---|---|---|
| FORECAST | 1st column | GROUP_ID | VARCHAR |
| | 2nd column | TIMESTAMP | INTEGER |
| | 3rd column | VALUE | DOUBLE |
| | 4th column | PI1_LOWER | DOUBLE |
| | 5th column | PI1_UPPER | DOUBLE |
| | 6th column | PI2_LOWER | DOUBLE |
| | 7th column | PI2_UPPER | DOUBLE |
| | 8th column | REASON | VARCHAR |
| STATISTICS | 1st column | GROUP_ID | VARCHAR |
| | 2nd column | STAT_NAME | VARCHAR, NVARCHAR |
| | 3rd column | STAT_VALUE | VARCHAR, NVARCHAR |
| ERRORMsgTBL | 1st column | GROUP_ID | VARCHAR |
| | 2nd column | ERROR_TIMESTAMP | VARCHAR, NVARCHAR |
| | 3rd column | ERRORCODE | INTEGER |
| | 4th column | MASSAGE | VARCHAR, NVARCHAR |
| PLCAE_HOLDER_1 | 1st column | OBJECT | VARCHAR, NVARCHAR |
| | 2nd column | KEY | VARCHAR, NVARCHAR |
| | 3rd column | VALUE | VARCHAR, NVARCHAR |
| PLACE_HOLDER_2 | 1st column | OJBECT | VARCHAR, NVARCHAR |
| | 2nd column | KEY | VARCHAR, NVARCHAR |
| | 3rd column | VALUE | VARCHAR, NVARCHAR |

### Example

Assume that:

- DM_PAL is a schema belonging to USER1;

- USER1 has been assigned the AFL__SYS_AFL_AFLPAL_EXECUTE or AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION role.

Example1: Single Exponential Smoothing

```
SET SCHEMA DM_PAL;
DROP TABLE META_DATA_TAB;
CREATE COLUMN TABLE META_DATA_TAB(
    "VARIABLE_NAME" NVARCHAR(50),
    "VARIABLE_TYPE" NVARCHAR(50)
);
INSERT INTO META_DATA_TAB VALUES ('ID',    'CONTINUOUS');
INSERT INTO META_DATA_TAB VALUES ('VALUE', 'TARGET');
DROP TABLE PAL_DATA_TBL;
CREATE COLUMN TABLE PAL_DATA_TBL(
    "ID" INTEGER,
    "VARIABLE_NAME" NVARCHAR(50),
    "VALUE" NVARCHAR(50)
);
INSERT INTO PAL_DATA_TBL VALUES (0,    'ID',    '1');
INSERT INTO PAL_DATA_TBL VALUES (0,    'VALUE',   '200.0');
INSERT INTO PAL_DATA_TBL VALUES (1,    'ID',   '2');
INSERT INTO PAL_DATA_TBL VALUES (1,    'VALUE',   '135.0');
INSERT INTO PAL_DATA_TBL VALUES (2,    'ID',    '3');
INSERT INTO PAL_DATA_TBL VALUES (2,    'VALUE',   '195.0');
INSERT INTO PAL_DATA_TBL VALUES (3,    'ID',   '4');
INSERT INTO PAL_DATA_TBL VALUES (3,    'VALUE',   '197.5');
INSERT INTO PAL_DATA_TBL VALUES (4,    'ID',    '5');
INSERT INTO PAL_DATA_TBL VALUES (4,    'VALUE',   '310.0');
INSERT INTO PAL_DATA_TBL VALUES (5,    'ID',   '6');
INSERT INTO PAL_DATA_TBL VALUES (5,    'VALUE',   '175.0');
INSERT INTO PAL_DATA_TBL VALUES (6,    'ID',    '7');
INSERT INTO PAL_DATA_TBL VALUES (6,    'VALUE',   '155.0');
INSERT INTO PAL_DATA_TBL VALUES (7,    'ID',   '8');
INSERT INTO PAL_DATA_TBL VALUES (7,    'VALUE',   '130.0');
INSERT INTO PAL_DATA_TBL VALUES (8,    'ID',    '9');
INSERT INTO PAL_DATA_TBL VALUES (8,    'VALUE',   '220.0');
INSERT INTO PAL_DATA_TBL VALUES (9,    'ID',   '10');
INSERT INTO PAL_DATA_TBL VALUES (9,    'VALUE',   '277.5');
INSERT INTO PAL_DATA_TBL VALUES (10,    'ID',    '11');
INSERT INTO PAL_DATA_TBL VALUES (10,    'VALUE',   '235.0');
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO PAL_PARAMETER_TBL VALUES ('FUNCTION',null,null,'SESM');
INSERT INTO PAL_PARAMETER_TBL VALUES ('ADAPTIVE_METHOD',0,null,null);
INSERT INTO PAL_PARAMETER_TBL VALUES ('MEASURE_NAME',null,null,'MSE');
INSERT INTO PAL_PARAMETER_TBL VALUES ('ALPHA', null,0.1,null);
INSERT INTO PAL_PARAMETER_TBL VALUES ('FORECAST_NUM', 12,null,null);
INSERT INTO PAL_PARAMETER_TBL VALUES ('EXPOST_FLAG', 1,null,null);
CALL _SYS_AFL.PAL_UNIFIED_EXPONENTIALSMOOTHING_PIVOT(META_DATA_TAB,
PAL_DATA_TBL, PAL_PARAMETER_TBL, ?, ?, ?, ?, ?);
```

**Expected Result**:

FORECAST:

Result 1 × Result 2 × Result 3 × Result 4 × Result 5 × Messages × History

Rows (22) SQL

| | GROUP_ID | TIMESTAMP | VALUE | PI1_LOWER | PI1_UPPER | PI2_LOWER | PI2_UPPER | REASON |
|---|---|---|---|---|---|---|---|---|
| 1 | NULL | 2 | 200 | NULL | NULL | NULL | NULL | [{"level":200.0}] |
| 2 | NULL | 3 | 193.5 | NULL | NULL | NULL | NULL | [{"level":193.5}] |
| 3 | NULL | 4 | 193.65 | NULL | NULL | NULL | NULL | [{"level":193.65000000000001}] |
| 4 | NULL | 5 | 194.035 | NULL | NULL | NULL | NULL | [{"level":194.035}] |
| 5 | NULL | 6 | 205.6315 | NULL | NULL | NULL | NULL | [{"level":205.63149999999999}] |
| 6 | NULL | 7 | 202.56834999999998 | NULL | NULL | NULL | NULL | [{"level":202.56834999999998}] |
| 7 | NULL | 8 | 197.81151499999999 | NULL | NULL | NULL | NULL | [{"level":197.81151499999999}] |
| 8 | NULL | 9 | 191.0303635 | NULL | NULL | NULL | NULL | [{"level":191.03036349999999}] |
| 9 | NULL | 10 | 193.92732715 | NULL | NULL | NULL | NULL | [{"level":193.92732715}] |
| 10 | NULL | 11 | 202.284594435 | NULL | NULL | NULL | NULL | [{"level":202.284594435}] |
| 11 | NULL | 12 | 205.55613499150002 | 126.700940203298 | 284.41132977970204 | 84.9575216121871 | 326.15474837081297 | [{"level":205.55613499150002}] |
| 12 | NULL | 13 | 205.55613499150002 | 126.30764502143097 | 284.80462496156906 | 84.35602853732595 | 326.75624144567405 | [{"level":205.55613499150002}] |
| 13 | NULL | 14 | 205.55613499150002 | 125.91629207360504 | 285.19597790939497 | 83.75750585310104 | 327.35476412989897 | [{"level":205.55613499150002}] |
| 14 | NULL | 15 | 205.55613499150002 | 125.52685286653129 | 285.58541711646876 | 83.16190998278736 | 327.95036000021264 | [{"level":205.55613499150002}] |
| 15 | NULL | 16 | 205.55613499150002 | 125.1392995968596 | 285.97297038614045 | 82.56919840483033 | 328.5430715781697 | [{"level":205.55613499150002}] |
| 16 | NULL | 17 | 205.55613499150002 | 124.75360512801403 | 286.358664854986 | 81.97932961741841 | 329.1329403655816 | [{"level":205.55613499150002}] |
| 17 | NULL | 18 | 205.55613499150002 | 124.36974296801866 | 286.74252701498136 | 81.39226310457079 | 329.7200068784292 | [{"level":205.55613499150002}] |
| 18 | NULL | 19 | 205.55613499150002 | 123.98768724826262 | 287.12458273473743 | 80.80795930366119 | 330.30431067933887 | [{"level":205.55613499150002}] |
| 19 | NULL | 20 | 205.55613499150002 | 123.60741270315617 | 287.50485727984386 | 80.22637957430457 | 330.88589040869545 | [{"level":205.55613499150002}] |
| 20 | NULL | 21 | 205.55613499150002 | 123.22889465063255 | 287.88337533236745 | 79.64748616853727 | 331.46478381446275 | [{"level":205.55613499150002}] |
| 21 | NULL | 22 | 205.55613499150002 | 122.85210897345293 | 288.2601610095471 | 79.07124220222539 | 332.04102778077464 | [{"level":205.55613499150002}] |
| 22 | NULL | 23 | 205.55613499150002 | 122.4770321012746 | 288.6352378817254 | 78.49761162764055 | 332.6146583553595 | [{"level":205.55613499150002}] |

STATISTIC:

Rows (1) SQL

| | GROUP_ID | STAT_NAME | STAT_VALUE |
|---|---|---|---|
| 1 | NULL | MSE | 3438.3321253085405 |

Example2: Massive Single Exponential Smoothing

```
SET SCHEMA DM_PAL;
DROP TABLE META_DATA_TAB;
CREATE COLUMN TABLE META_DATA_TAB(
    "VARIABLE_NAME" NVARCHAR(50),
    "VARIABLE_TYPE" NVARCHAR(50)
);
INSERT INTO META_DATA_TAB VALUES ('GROUP_ID', 'CATEGORICAL');
INSERT INTO META_DATA_TAB VALUES ('ID',    'CONTINUOUS');
INSERT INTO META_DATA_TAB VALUES ('VALUE', 'TARGET');
DROP TABLE PAL_DATA_TBL;
CREATE COLUMN TABLE PAL_DATA_TBL(
    "ID" INT,
    "VARIABLE_NAME" NVARCHAR(50),
    "VALUE" NVARCHAR(50)
);
INSERT INTO PAL_DATA_TBL VALUES (0,    'GROUP_ID',    'Hello1');
INSERT INTO PAL_DATA_TBL VALUES (0,    'ID',    '1');
INSERT INTO PAL_DATA_TBL VALUES (0,    'VALUE',    '200.0');
INSERT INTO PAL_DATA_TBL VALUES (1,    'GROUP_ID',    'Hello1');
INSERT INTO PAL_DATA_TBL VALUES (1,    'ID',    '2');
INSERT INTO PAL_DATA_TBL VALUES (1,    'VALUE',    '135.0');
INSERT INTO PAL_DATA_TBL VALUES (2,    'GROUP_ID',    'Hello1');
INSERT INTO PAL_DATA_TBL VALUES (2,    'ID',    '3');
INSERT INTO PAL_DATA_TBL VALUES (2,    'VALUE',    '195.0');
INSERT INTO PAL_DATA_TBL VALUES (3,    'GROUP_ID',    'Hello1');
INSERT INTO PAL_DATA_TBL VALUES (3,    'ID',    '4');
INSERT INTO PAL_DATA_TBL VALUES (3,    'VALUE',    '197.5');
INSERT INTO PAL_DATA_TBL VALUES (4,    'GROUP_ID',    'Hello1');
INSERT INTO PAL_DATA_TBL VALUES (4,    'ID',    '5');
INSERT INTO PAL_DATA_TBL VALUES (4,    'VALUE',    '310.0');
```

```
INSERT INTO PAL_DATA_TBL VALUES (5,    'GROUP_ID',    'Hello1');
INSERT INTO PAL_DATA_TBL VALUES (5,    'ID',   '6');
INSERT INTO PAL_DATA_TBL VALUES (5,    'VALUE',   '175.0');
INSERT INTO PAL_DATA_TBL VALUES (6,    'GROUP_ID',    'Hello1');
INSERT INTO PAL_DATA_TBL VALUES (6,    'ID',    '7');
INSERT INTO PAL_DATA_TBL VALUES (6,    'VALUE',   '155.0');
INSERT INTO PAL_DATA_TBL VALUES (7,    'GROUP_ID',    'Hello1');
INSERT INTO PAL_DATA_TBL VALUES (7,    'ID',   '8');
INSERT INTO PAL_DATA_TBL VALUES (7,    'VALUE',   '130.0');
INSERT INTO PAL_DATA_TBL VALUES (8,    'GROUP_ID',    'Hello1');
INSERT INTO PAL_DATA_TBL VALUES (8,    'ID',    '9');
INSERT INTO PAL_DATA_TBL VALUES (8,    'VALUE',   '220.0');
INSERT INTO PAL_DATA_TBL VALUES (9,    'GROUP_ID',    'Hello1');
INSERT INTO PAL_DATA_TBL VALUES (9,    'ID',   '10');
INSERT INTO PAL_DATA_TBL VALUES (9,    'VALUE',   '277.5');
INSERT INTO PAL_DATA_TBL VALUES (10,    'GROUP_ID',    'Hello1');
INSERT INTO PAL_DATA_TBL VALUES (10,    'ID',    '11');
INSERT INTO PAL_DATA_TBL VALUES (10,    'VALUE',   '235.0');
--full zero
INSERT INTO PAL_DATA_TBL VALUES (11,    'GROUP_ID',    'Hello2');
INSERT INTO PAL_DATA_TBL VALUES (11,    'ID',    '1');
INSERT INTO PAL_DATA_TBL VALUES (11,    'VALUE',   '0.0');
INSERT INTO PAL_DATA_TBL VALUES (12,    'GROUP_ID',    'Hello2');
INSERT INTO PAL_DATA_TBL VALUES (12,    'ID',   '2');
INSERT INTO PAL_DATA_TBL VALUES (12,    'VALUE',   '0.0');
INSERT INTO PAL_DATA_TBL VALUES (13,    'GROUP_ID',    'Hello2');
INSERT INTO PAL_DATA_TBL VALUES (13,    'ID',    '3');
INSERT INTO PAL_DATA_TBL VALUES (13,    'VALUE',   '0.0');
INSERT INTO PAL_DATA_TBL VALUES (14,    'GROUP_ID',    'Hello2');
INSERT INTO PAL_DATA_TBL VALUES (14,    'ID',   '4');
INSERT INTO PAL_DATA_TBL VALUES (14,    'VALUE',   '0.0');
INSERT INTO PAL_DATA_TBL VALUES (15,    'GROUP_ID',    'Hello2');
INSERT INTO PAL_DATA_TBL VALUES (15,    'ID',    '5');
INSERT INTO PAL_DATA_TBL VALUES (15,    'VALUE',   '0.0');
INSERT INTO PAL_DATA_TBL VALUES (16,    'GROUP_ID',    'Hello2');
INSERT INTO PAL_DATA_TBL VALUES (16,    'ID',   '6');
INSERT INTO PAL_DATA_TBL VALUES (16,    'VALUE',   '0.0');
INSERT INTO PAL_DATA_TBL VALUES (17,    'GROUP_ID',    'Hello2');
INSERT INTO PAL_DATA_TBL VALUES (17,    'ID',    '7');
INSERT INTO PAL_DATA_TBL VALUES (17,    'VALUE',   '0.0');
INSERT INTO PAL_DATA_TBL VALUES (18,    'GROUP_ID',    'Hello2');
INSERT INTO PAL_DATA_TBL VALUES (18,    'ID',   '8');
INSERT INTO PAL_DATA_TBL VALUES (18,    'VALUE',   '0.0');
INSERT INTO PAL_DATA_TBL VALUES (19,    'GROUP_ID',    'Hello2');
INSERT INTO PAL_DATA_TBL VALUES (19,    'ID',    '9');
INSERT INTO PAL_DATA_TBL VALUES (19,    'VALUE',   '0.0');
INSERT INTO PAL_DATA_TBL VALUES (20,    'GROUP_ID',    'Hello2');
INSERT INTO PAL_DATA_TBL VALUES (20,    'ID',   '10');
INSERT INTO PAL_DATA_TBL VALUES (20,    'VALUE',   '0.0');
INSERT INTO PAL_DATA_TBL VALUES (21,    'GROUP_ID',    'Hello2');
INSERT INTO PAL_DATA_TBL VALUES (21,    'ID',    '11');
INSERT INTO PAL_DATA_TBL VALUES (21,    'VALUE',   '0.0');
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL (
    "GROUP_ID" VARCHAR (50),
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO PAL_PARAMETER_TBL VALUES (null, 'FUNCTION',null,null,'MSESM');
INSERT INTO PAL_PARAMETER_TBL VALUES ('Hello1', 'ADAPTIVE_METHOD',0,null,null);
INSERT INTO PAL_PARAMETER_TBL VALUES ('Hello1', 'MEASURE_NAME',null,null,'MSE');
INSERT INTO PAL_PARAMETER_TBL VALUES ('Hello1', 'ALPHA', null,0.1,null);
INSERT INTO PAL_PARAMETER_TBL VALUES ('Hello1', 'FORECAST_NUM', 12,null,null);
INSERT INTO PAL_PARAMETER_TBL VALUES ('Hello1', 'EXPOST_FLAG', 1,null,null);
INSERT INTO PAL_PARAMETER_TBL VALUES ('Hello2', 'ADAPTIVE_METHOD',1,null,null);
```

```
INSERT INTO PAL_PARAMETER_TBL VALUES ('Hello2',
'MEASURE_NAME',null,null,'MAPE');
INSERT INTO PAL_PARAMETER_TBL VALUES ('Hello2', 'ALPHA', null,0.5,null);
INSERT INTO PAL_PARAMETER_TBL VALUES ('Hello2', 'FORECAST_NUM', 12,null,null);
INSERT INTO PAL_PARAMETER_TBL VALUES ('Hello2', 'EXPOST_FLAG', 0,null,null);
CALL _SYS_AFL.PAL_UNIFIED_EXPONENTIALSMOOTHING_PIVOT(META_DATA_TAB,
PAL_DATA_TBL, PAL_PARAMETER_TBL, ?, ?, ?, ?, ?);
```

**Expected Result**:

FORECAST:

Result 1 × Result 2 × Result 3 × Result 4 × Result 5 × Messages × History

Rows (34) SQL

| | GROUP_ID | TIMESTAMP | VALUE | PI1_LOWER | PI1_UPPER | PI2_LOWER | PI2_UPPER | REASON |
|---|---|---|---|---|---|---|---|---|
| 9 | Hello1 | 10 | 193.92732715 | *NULL* | *NULL* | *NULL* | *NULL* | { "level" : 193.92732715 } |
| 10 | Hello1 | 11 | 202.284594435 | *NULL* | *NULL* | *NULL* | *NULL* | { "level" : 202.284594435 } |
| 11 | Hello1 | 12 | 205.55613499150002 | 126.700940203298 | 284.41132977970204 | 84.9575216121871 | 326.15474837081297 | { "level" : 205.55613499150002 } |
| 12 | Hello1 | 13 | 205.55613499150002 | 126.30764502143097 | 284.80462496156906 | 84.35602853732595 | 326.75624144567405 | { "level" : 205.55613499150002 } |
| 13 | Hello1 | 14 | 205.55613499150002 | 125.91629207360504 | 285.19597790939497 | 83.75750585310104 | 327.35476412989897 | { "level" : 205.55613499150002 } |
| 14 | Hello1 | 15 | 205.55613499150002 | 125.52685286653129 | 285.58541711646876 | 83.16190998278736 | 327.95036000021264 | { "level" : 205.55613499150002 } |
| 15 | Hello1 | 16 | 205.55613499150002 | 125.1392995968596 | 285.97297038614045 | 82.56919840483033 | 328.5430715781697 | { "level" : 205.55613499150002 } |
| 16 | Hello1 | 17 | 205.55613499150002 | 124.75360512801403 | 286.358664854986 | 81.97932961741841 | 329.1329403655816 | { "level" : 205.55613499150002 } |
| 17 | Hello1 | 18 | 205.55613499150002 | 124.36974296801866 | 286.74252701498136 | 81.39226310457079 | 329.7200068784292 | { "level" : 205.55613499150002 } |
| 18 | Hello1 | 19 | 205.55613499150002 | 123.98768724826262 | 287.12458273473743 | 80.80795930366119 | 330.30431067933887 | { "level" : 205.55613499150002 } |
| 19 | Hello1 | 20 | 205.55613499150002 | 123.60741270315617 | 287.50485727984386 | 80.22637957430457 | 330.88589040869545 | { "level" : 205.55613499150002 } |
| 20 | Hello1 | 21 | 205.55613499150002 | 123.22889465063255 | 287.88337533236745 | 79.64748616853727 | 331.46478381446275 | { "level" : 205.55613499150002 } |
| 21 | Hello1 | 22 | 205.55613499150002 | 122.85210897345293 | 288.2601610095471 | 79.07124220222539 | 332.04102778077464 | { "level" : 205.55613499150002 } |
| 22 | Hello1 | 23 | 205.55613499150002 | 122.4770321012746 | 288.6352378817254 | 78.49761162764055 | 332.6146583553595 | { "level" : 205.55613499150002 } |
| 23 | Hello2 | 12 | 0 | 0 | 0 | 0 | 0 | null |
| 24 | Hello2 | 13 | 0 | 0 | 0 | 0 | 0 | null |
| 25 | Hello2 | 14 | 0 | 0 | 0 | 0 | 0 | null |
| 26 | Hello2 | 15 | 0 | 0 | 0 | 0 | 0 | null |
| 27 | Hello2 | 16 | 0 | 0 | 0 | 0 | 0 | null |
| 28 | Hello2 | 17 | 0 | 0 | 0 | 0 | 0 | null |
| 29 | Hello2 | 18 | 0 | 0 | 0 | 0 | 0 | null |
| 30 | Hello2 | 19 | 0 | 0 | 0 | 0 | 0 | null |

STATISTIC:

Rows (2) SQL

| | GROUP_ID | STAT_NAME | STAT_VALUE |
|---|---|---|---|
| 1 | Hello1 | MSE | 3438.3321253085405 |
| 2 | Hello2 | MAPE | 0 |

## 5.13.14.1 Parameter Tables

This topic lists all parameters for unified exponential smoothing with pivoted input data.

## _SYS_AFL.PAL_UNIFIED_EXPONENTIALSMOOTHING_PIVOT

### UNIFIED EXPONENTIAL SMOOTHING

Mandatory parameters

| Name | Data Type | Description |
| --- | --- | --- |
| FUNCTION | VARCHAR or NVARCHAR | The algorithm name, including:<br>• SESM;<br>• DESM;<br>• TESM;<br>• BESM;<br>• AESM;<br>• MSESM;<br>• MDESM;<br>• MTESM;<br>• MBESM;<br>• MAESM; |

Optional parameters

| Name | Data Type | Default Value | Description | Dependency |
| --- | --- | --- | --- | --- |
| FORECAST_NUM | INTEGER | 0 | Number of values to be forecast. | |
| PREDICTION_CONFIDENCE_1 | DOUBLE | 0.8 | Prediction confidence for interval 1 | Only valid when the upper and lower columns are provided in the result table. |
| PREDICTION_CONFIDENCE_2 | DOUBLE | 0.95 | Prediction confidence for interval 2 | Only valid when the upper and lower columns are provided in the result table. |
| DECOM_STATE | INTEGER | 0 | Specifies whether or not to decompose the forecast value.<br>• 0: non-decomposition<br>• 1: decomposition | |

### SESM & MSESM

Mandatory parameters

None

Optional parameters

| Name | Data Type | Default Value | Description | Denpendency |
|---|---|---|---|---|
| ALPHA | DOUBLE | 0.1 (Single exponential smoothing) 0.2 (Adaptive-response-rate single exponential smoothing) | The smoothing constant alpha for single exponential smoothing or the initialization value for adaptive-response-rate single exponential smoothing ($0 < \alpha < 1$). | |
| DELTA | DOUBLE | 0.2 | Value of weighted for At and Mt. | Only valid when ADAPTIVE_METHOD is 1. |
| ADAPTIVE_METHOD | INTEGER | 0 | • 0: Single exponential smoothing<br>• 1: Adaptive-response-rate single exponential smoothing | |
| MEASURE_NAME | VARCHAR or NVARCHAR | No default value | Measure name. Supported measures are MPE, MSE, RMSE, ET, MAD, MASE, WMAPE, SMAPE, and MAPE. For detailed information on measures, see Forecast Accuracy Measures. | |
| IGNORE_ZERO | INTEGER | 0 | • 0: Uses zero values in the input dataset when calculating MPE or MAPE.<br>• 1: Ignores zero values in the input dataset when calculating MPE or MAPE. | Only valid when MEASURE_NAME is MPE or MAPE. |
| EXPOST_FLAG | INTEGER | 1 | • 0: Does not output the expost forecast, and just outputs the forecast values.<br>• 1: Outputs the expost forecast and the forecast values. | |

### DESM & MDESM

Mandatory parameters

None

Optional parameters

| Name | Data Type | Default Value | Description | Denpendency |
|---|---|---|---|---|
| ALPHA | DOUBLE | 0.1 | Value of the smoothing constant alpha ($0 < \alpha < 1$). | |
| BETA | DOUBLE | 0.1 | Value of the smoothing constant beta ($0 < \beta < 1$). | |
| PHI | DOUBLE | 0.1 | Value of the damped smoothing constant $\Phi$ ($0 < \Phi < 1$). | |
| DAMPED | INTEGER | 0 | • 0: Uses the Holt's linear method.<br>• 1: Uses the additive damped trend Holt's linear method. | |
| MEASURE_NAME | VARCHAR or NVARCHAR | No default value | Measure name. Supported measures are MPE, MSE, RMSE, ET, MAD, MASE, WMAPE, SMAPE, and MAPE. For detailed information on measures, see Forecast Accuracy Measures. | |
| IGNORE_ZERO | INTEGER | 0 | • 0: Uses zero values in the input dataset when calculating MPE or MAPE.<br>• 1: Ignores zero values in the input dataset when calculating MPE or MAPE. | Only valid when MEASURE_NAME is MPE or MAPE. |
| EXPOST_FLAG | INTEGER | 1 | • 0: Does not output the expost forecast, and just outputs the forecast values.<br>• 1: Outputs the expost forecast and the forecast values. | |

## TESM & MTESM

Mandatory parameters

None

Optional parameters

| Name | Data Type | Default Value | Description | Denpendency |
|---|---|---|---|---|
| ALPHA | DOUBLE | 0.1 | Value of the smoothing constant alpha ($0 < \alpha < 1$). | |
| BETA | DOUBLE | 0.1 | Value of the smoothing constant beta ($0 < \beta < 1$). | |
| GAMMA | DOUBLE | 0.1 | Value of the smoothing constant gamma ( $0 < \gamma < 1$). | |
| CYCLE | INTEGER | 2 | A cycle of length L ($L > 1$). For example, quarterly data cycle is 4, and monthly data cycle is 12. | |
| SEASONAL | INTEGER | 2 | • 0: Multiplicative triple exponential smoothing<br>• 1: Additive triple exponential smoothing | When SEASONAL is set to 1, the default value of INITIAL_METHOD is 1; When SEASONAL is set to 0, the default value of INITIAL_METHOD is 0. |
| INITIAL_METHOD | INTEGER | 0 or 1 | Initialization method for trend and seasonal:<br>• 0: The first initialization method as the description part.<br>• 1: The second initialization method as the description part. | When SEASONAL is set to 1, the default value of INITIAL_METHOD is 1; When SEASONAL is set to 0, the default value of INITIAL_METHOD is 0. |
| PHI | DOUBLE | 0.1 | Value of the damped smoothing constant $\Phi$ ($0 < \Phi < 1$). | |
| DAMPED | INTEGER | 0 | • 0: Uses the Holt's linear method.<br>• 1: Uses the additive damped trend Holt's linear method. | |

| Name | Data Type | Default Value | Description | Denpendency |
|---|---|---|---|---|
| MEASURE_NAME | VARCHAR or NVARCHAR | No default value | Measure name. Supported measures are MPE, MSE, RMSE, ET, MAD, MASE, WMAPE, SMAPE, and MAPE. For detailed information on measures, see Forecast Accuracy Measures. | |
| IGNORE_ZERO | INTEGER | 0 | • 0: Uses zero values in the input dataset when calculating MPE or MAPE.<br>• 1: Ignores zero values in the input dataset when calculating MPE or MAPE. | Only valid when MEASURE_NAME is MPE or MAPE. |
| EXPOST_FLAG | INTEGER | 1 | • 0: Does not output the expost forecast, and just outputs the forecast values.<br>• 1: Outputs the expost forecast and the forecast values. | |
| LEVEL_START | DOUBLE | No default value | The initial value for level component S. If this value is not provided, it will be calculated in the way as described earlier. Note: LEVEL_START cannot be zero. If it is set to zero, 0.0000000001 will be used instead. | |
| TREND_START | DOUBLE | No default value | The initial value for trend component B. If this value is not provided, it will be calculated in the way as described earlier. | |

| Name | Data Type | Default Value | Description | Denpendency |
|---|---|---|---|---|
| SEASON_START | INTEGER or DOUBLE | No default value | A list of initial values for seasonal component C. If this parameter is not used, the initial values of C will be calculated in the way as described earlier. Two values must be provided for each cycle: Cycle ID: An INTEGER which represents which cycle the initial value is used for.Initial value: A DOUBLE precision number which represents the initial value for the corresponding cycle. For example: To give the initial value 0.5 to the 3rd cycle, insert tuple ('SEASON_START', 3, 0.5, NULL) into the parameter table. Note: The initial values of all cycles must be provided if this parameter is used. | |

### BESM & MBESM

Mandatory parameters

None

Optional parameters

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| ALPHA | DOUBLE | 0.1 (Brown exponential smoothing) 0.2 (Adaptive brown exponential smoothing) | The smoothing constant alpha for brown exponential smoothing or the initialization value for adaptive brown exponential smoothing ($0 < \alpha < 1$). | |
| DELTA | DOUBLE | 0.2 | Value of weighted for At and Mt. | Only valid when ADAPTIVE_METHOD is 1. |
| ADAPTIVE_METHOD | INTEGER | 0 | • 0: Brown exponential smoothing<br>• 1: Adaptive brown exponential smoothing | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| MEASURE_NAME | VARCHAR or NVARCHAR | No default value | Measure name. Supported measures are MPE, MSE, RMSE, ET, MAD, MASE, WMAPE, SMAPE, and MAPE. For detailed information on measures, see Forecast Accuracy Measures. | |
| IGNORE_ZERO | INTEGER | 0 | • 0: Uses zero values in the input dataset when calculating MPE or MAPE.<br>• 1: Ignores zero values in the input dataset when calculating MPE or MAPE. | Only valid when MEASURE_NAME is MPE or MAPE. |
| EXPOST_FLAG | INTEGER | 1 | • 0: Does not output the expost forecast, and just outputs the forecast values.<br>• 1: Outputs the expost forecast and the forecast values. | |

### AESM & MAESM

Mandatory Parameters

None.

Optional Parameters

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| MODELSELECTION | INTEGER | 0 | When this is set to 1, the algorithm will select the best model among Single/Double/Triple/Damped Double/Damped Triple Exponential Smoothing models. If FORECAST_MODEL_NAME is set, the model defined by FORECAST_MODEL_NAME will be used. When this is set to 0, the algorithm will not perform the model selection. | |
| FORECAST_MODEL_NAME | VARCHAR or NVARCHAR | No default value | Name of the statistical model used for calculating the forecast.<br>• SESM: Single Exponential Smoothing<br>• DESM: Double Exponential Smoothing<br>• TESM: Triple Exponential Smoothing | This parameter must be set unless MODELSELECTION is set to 1. |
| OPTIMIZER_TIME_BUDGET | INTEGER | 1 | Time budget for Nelder-Mead optimization process. The time unit is second and the value should be larger than zero. | |
| MAX_ITERATION | INTEGER | 100 | Maximum number of iterations for simulated annealing. | |
| OPTIMIZER_RANDOM_SEED | INTEGER | System time | Random seed for simulated annealing. The value should be larger than zero. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| THREAD_RATIO | DOUBLE | -1.0 | Specifies the ratio of total number of threads that can be used by this function. The value range is from 0 to 1, where 0 means only using one thread, and 1 means using at most all the currently available threads. Values outside the range are ignored and this function heuristically determines the number of threads to use. | |
| EXPOST_FLAG | INTEGER | 1 | • 0: Does not output the expost forecast, and just outputs the forecast values.<br>• 1: Outputs the expost forecast and the forecast values. | |
| ALPHA | DOUBLE | Computed automatically | Weight for smoothing. Value range: $0 < \alpha < 1$ | |
| BETA | DOUBLE | Computed automatically | Weight for the trend component. Value range: 0 Value range: $0 \leq \beta < 1$ Note: If it is not set, the optimized value will be computed automatically. | Only valid when the model is set by user or identified by the algorithm as a DESM/TESM. Value 0 is allowed under TESM model only. |
| GAMMA | DOUBLE | Computed automatically | Weight for the seasonal component. Value range: $0 < \gamma < 1$ | Only valid when the model is set by user or identified by the algorithm as TESM. |
| PHI | DOUBLE | Computed automatically | Value of the damped smoothing constant $\Phi$ ($0 < \Phi < 1$). | Only valid when the model is set by user or identified by the algorithm as a DAMPED model. |
| CYCLE | INTEGER | Computed automatically | Length of a cycle (L > 1). For example, the cycle of quarterly data is 4, and the cycle of monthly data is 12. | Only valid when the model is set by user or identified by the algorithm as a TESM. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| SEASONAL | INTEGER | If MODELSELECTION is set to 1, the default value will be computed automatically. Otherwise, the default value is 0. | • 0: Multiplicative triple exponential smoothing<br>• 1: Additive triple exponential smoothing | Only valid when the model is set by user or identified by the algorithm as a TESM. |
| INITIAL_METHOD | INTEGER | 0 or 1 | Initialization method for the trend and seasonal components. Refer to Triple Exponential Smoothing for detailed information on initialization method. | Only valid when the model is set by user or identified by the algorithm as a TESM. |
| TRAINING_RATIO | DOUBLE | 1.0 | The ratio of training data to the whole time series. Assuming the size of time series is N, and the training ratio is r, the first N*r time series is used to train, whereas only the latter N*(1-r) one is used to test. | If this parameter is set to 0.0 or 1.0, or the resulting training data (N*r) is less than 1 or equal to the size of time series, no train-and-test procedure is carried out. |
| DAMPED | INTEGER | If MODELSELECTION is set to 1, the default value will be computed automatically. Otherwise, the default value is 0. | For DESM:<br>• 0: Uses the Holt's linear method.<br>• 1: Uses the additive damped trend Holt's linear method.<br>For TESM:<br>• 0: Uses the Holt Winter method.<br>• 1: Uses the additive damped seasonal Holt Winter method. | |
| ACCURACY_MEASURE | VARCHAR Or NVARCHAR | MSE | The criterion used for the optimization. Available values are MSE and MAPE. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| SEASONALITY_CRITERION | DOUBLE | 0.5 | The criterion of the auto-correlation coefficient for accepting seasonality, in the range of (0, 1). The larger it is, the less probable a time series is regarded to be seasonal. | Only valid when FORECAST_MODEL_NAME is TESM or MODELSELECTION is set to 1, and CYCLE is not defined. |
| TREND_TEST_METHOD | INTEGER | 1 | The method to test trend:<br>• 1: MK test<br>• 2: Difference-sign test | Only valid when MODELSELECTION is set to 1. |
| TREND_TEST_ALPHA | DOUBLE | 0.05 | Tolerance probability for trend test. The value range is (0, 0.5). | Only valid when MODELSELECTION is set to 1. |
| ALPHA_MIN | DOUBLE | 0.0000000001 | Sets the minimum value of ALPHA. | Only valid when ALPHA is not defined. |
| BETA_MIN | DOUBLE | 0.0000000001 | Sets the minimum value of BETA. | Only valid when BETA is not defined. |
| GAMMA_MIN | DOUBLE | 0.0000000001 | Sets the minimum value of GAMMA. | Only valid when GAMMA is not defined. |
| PHI_MIN | DOUBLE | 0.0000000001 | Sets the minimum value of PHI. | Only valid when PHI is not defined. |
| ALPHA_MAX | DOUBLE | 1 | Sets the maximum value of ALPHA. | Only valid when ALPHA is not defined. |
| BETA_MAX | DOUBLE | 1 | Sets the maximum value of BETA. | Only valid when BETA is not defined. |
| GAMMA_MAX | DOUBLE | 1 | Sets the maximum value of GAMMA. | Only valid when GAMMA is not defined. |
| PHI_MAX | DOUBLE | 1 | Sets the maximum value of PHI. | Only valid when PHI is not defined. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| LEVEL_START | DOUBLE | No default value | The initial value for level component S. If this value is not provided, it will be calculated in the way as described in Triple Exponential Smoothing. Note: LEVEL_START cannot be zero. If it is set to zero, 0.0000000001 will be used instead. Refer to Triple Exponential Smoothing for detailed information | |
| TREND_START | DOUBLE | No default value | The initial value for trend component B. If this value is not provided, it will be calculated in the way as described in Triple Exponential Smoothing. Refer to Triple Exponential Smoothing for detailed information | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| SEASON_START | INTEGER or DOUBLE | No defalut value | A list of initial values for seasonal component C. If this parameter is not used, the initial values of C will be calculated in the way as described in Triple Exponential Smoothing. Two values must be provided for each cycle:<br>• Cycle ID: An INTEGER which represents which cycle the initial value is used for.<br>• Initial value: A DOUBLE precision number which represents the initial value for the corresponding cycle.<br>For example: To give the initial value 0.5 to the 3rd cycle, insert tuple ('SEASON_START', 3, 0.5, NULL) into the parameter table. Note: The initial values of all cycles must be provided if this parameter is used. Refer to Triple Exponential Smoothing for detailed information | |

## 5.14 Text Processing Algorithms

This section describes the text processing related functions that are provided by the Predictive Analysis Library.

## 5.14.1 Conditional Random Field

Conditional random fields (CRFs) are a probabilistic framework for labeling and segmenting structured data, such as sequences.

The underlying idea is that of defining a conditional probability distribution over label sequences given an observation sequences, rather than a joint distribution over both label and observation sequences. Let Y, X be random vectors, then a linear-chain conditional random field is a distribution $p(y|x)$ that takes the form:

$$p(y|x)=\frac{\exp\left(\sum_{t=1}^{T}\sum_{k=1}^{K}\theta_k f_k(y_{t-1},\ y_t,\ x_t,t)\right)}{Z(x)}$$

where $Z(x)=\sum_y \exp\left(\sum_{t=1}^{T}\sum_{k=1}^{K}\theta_k f_k(y_{t-1},\ y_t,\ x_t,t)\right)$ is a normalization factor, $f_k$ is feature function of the entire observation sequence and the labels at position $t$ and $t-1$ in the labels sequence; $\theta_k$ is parameter to be estimated from training data. $f_k$ includes node feature function and edge feature function.

Assuming the training data $\{x^{(i)},\ y^{(i)},\ i=1,\ldots,N\}$ are independently and identically distributed, the product of the above $p(y|x)$ over all training sequences, as a function of the parameters $\lambda$, is known as the likelihood. For CRF, the log-likelihood is given by

$$L(\theta)=\sum_{i=1}^{N}\sum_{t=1}^{T}\sum_{k=1}^{K}\theta_k f_k\left(y_{t-1}^{(i)},\ y_t^{(i)},x_t,\ t\right)+\sum_{i=1}^{N}\log\frac{1}{Z(x^{(i)})}$$

$N$ is the length of total sentences, $T$ is the length of a sentence, $K$ is the number of feature function.

As a measure to avoid overfitting, we use regularization, which is a penalty on weights vectors whose norm is too large. A common choice of penalty is based on the Euclidean norm of $\theta$ and on a regularization parameter $\frac{1}{2}\lambda$. Then the regularized log likelihood is

$$l(\theta)=L(\theta)-\frac{1}{2}\lambda\sum_{k=1}^{K}\theta_k^2$$

L-BFGS is provided to minimize objective function which is opposite in sign with regularized log likelihood function.

In PAL, linear-chain CRF can be applied to Named-entity recognition problem. The named-entity recognition task is, given a sentence, first to segment which words are part of entity, and then to classify each entity by type (PERSON, ORGANIZATION, LOCATION, and so on). An example sentence is “U.N. official Ekeus heads for Baghdad.” In this sentence, the token U.N. is labeled as an organization, Ekeus a person, and Baghdad a location.

### Prerequisite

- No missing or null data in input data.
- The data table must have Document ID and word position columns in training and inference phase.

# _SYS_AFL.PAL_CRF

This function trains the CRF model for English text.

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<Model table>` | OUT |
| 4 | `<STATISTICS table>` | OUT |
| 5 | `<Optimal Param table>` | OUT |

### Input Tables

| Table | Column | Data Type | Description |
|---|---|---|---|
| DATA | Document ID | INTEGER, VRACHAR, or NVARCHAR | |
| | Word position | INTEGER | |
| | Word | VARCHAR or NVARCHAR | |
| | Label | VARCHAR or NVARCHAR | |

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| THREAD_RATIO | DOUBLE | 1.0 | Specifies the ratio of total number of threads that can be used by this function. The range of this parameter is from 0 to 1, where 0 means only using 1 thread, and 1 means using at most all the currently available threads. Values outside this range are ignored and this function heuristically determines the number of threads to use. | |
| ENET_LAMBDA | DOUBLE | 1.0 | Penalized weights. The value should be greater than 0. | |
| EXIT_THRESHOLD | DOUBLE | 1e-4 | Convergence tolerance in optimization. | |
| MAX_ITERATION | INTEGER | 1000 | Maximum iteration number in optimization. | |
| LBFGS_M | INTEGER | 25 | Number of previous iterations of updates to store. | |
| USE_CLASS_FEATURE | INTEGER | 1 | To include a feature for the class. This is the same thing as having a bias vector in a model. | |
| USE_WORD | INTEGER | 1 | Gives you feature for current word. | |
| USE_NGRAMS | INTEGER | 1 | Make feature from letter n-grams, that is, substrings of the word. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| NO_MIDNGRAMS | INTEGER | 1 | Not to include character n-gram features for n-grams that contain neither the beginning or the end of the word. | |
| MAX_NGRAM_LENGTH | INTEGER | 6 | If this number is positive, n-grams above this size will not be used in the model. | |
| USE_PREV | INTEGER | 1 | Give you feature for previous word and current word, and together with other options enables other previous features. | |
| USE_NEXT | INTEGER | 1 | Give you feature for next word and current word. | |
| USE_DISJUNCTIVE | INTEGER | 1 | Include in features giving disjunctions of words anywhere in left or right disjunctionWidth words | |
| DISJUNCTION_WIDTH | INTEGER | 4 | | |
| USE_SEQUENCES | INTEGER | 1 | Not to use any class combination features if this is 0. | |
| USE_PREVSEQUENCES | INTEGER | 1 | Not to use any class combination features using the previous class if this is 0. | |
| USE_TYPE_SEQS | INTEGER | 1 | To use basic zeroth order word shape features. | |
| USE_TYPE_SEQS2 | INTEGER | 1 | Add additional first and second order word shape features. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| USE_TYPE_YSEQUENCES | INTEGER | 1 | Some first order word shape patterns. | |
| WORD_SHAPE | INTEGER | 0 | Word shape. For example, whether capitalized or numeric. Only supports chris2UseLC currently. Do not use word shape if this is 0. | |

### Output Tables

| Table | Column | Column Name | Data Type | Description |
|---|---|---|---|---|
| MODEL | 1st column | ROW_INDEX | INTEGER | |
| | 2nd column | MODEL_CONTENT | NCLOB | |
| STATISTICS | 1st column | STAT_NAME | NVARCHAR (256) | |
| | 2nd column | STAT_VALUE | NVARCHAR (1000) | |
| OPTIMAL_PARAM | 1st column | PARAM_NAME | NVARCHAR (256) | |
| | 2nd column | INT_VALUE | INTEGER | |
| | 3rd column | DOUBLE_VALUE | DOUBLE | |
| | 4th column | STRING_VALUE | NVARCHAR (1000) | |

### Examples

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

Example:

```
SET SCHEMA DM_PAL;
--prepare train data
DROP TABLE PAL_CRF_TRAIN_TBL;
CREATE COLUMN TABLE PAL_CRF_TRAIN_TBL("DOC_ID" INTEGER, "WORD_POSITION" INTEGER,
"WORD" VARCHAR(500), "LABEL" VARCHAR(100));
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(1,1,'RECORD','O');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(1,2,'#497321','O');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(1,3,'78554939','O');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(1,4,'|','O');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(1,5,'LRH','O');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(1,6,'|','O');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(1,7,'62413233','O');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(1,8,'|','O');
```

```
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(1,9,'|','O');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(1,10,'7368393','O');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(1,11,'|','O');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(1,12,'6','O');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(1,13,'g','O');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(1,14,'24','O');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(1,15,'g','O');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(1,16,'2007','O');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(1,17,'12:00:00','O');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(1,18,'AM','O');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(1,19,'|','O');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(1,20,'INCACERATED','O');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(1,21,'HERNIA','O');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(1,22,'SEPSIS','O');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(1,23,'Unsigned','O');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(1,24,'Admission','O');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(1,25,'systolic','O');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(1,26,'less','O');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(1,27,'than','O');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(1,28,'90','O');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(1,29,'heart','O');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(1,30,'PHYSICAL','O');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(1,31,'EXAMINATION','O');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(1,32,':','O');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(1,33,'VITAL','O');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(1,34,'SIGNS','O');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(1,35,':','O');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(1,36,'Blood','O');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(1,37,'pressure','O');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(1,38,'114g58','O');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(1,39,'_','O');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(1,40,'pulse','O');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(1,41,'68','O');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(1,42,'_','O');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(1,43,'respiratory','O');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(1,44,'rate','O');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(1,45,'20','O');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(1,46,'_','O');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(1,47,'she','O');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(1,48,'was','O');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(1,49,'afebrile','O');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(1,50,'at','O');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(1,51,'98.4','O');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(1,52,'_','O');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(1,53,'and','O');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(1,54,'O2','OxygenSaturation');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(1,55,'saturation','OxygenSaturation');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(1,56,'96%','OxygenSaturation');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(1,57,'on','OxygenSaturation');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(1,58,'room','OxygenSaturation');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(1,59,'air','OxygenSaturation');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(3,1,'PHYSICAL','O');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(3,2,'EXAMINATION','O');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(3,3,':','O');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(3,4,'VITAL','O');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(3,5,'SIGNS','O');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(3,6,':','O');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(3,7,'Heart','O');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(3,8,'rate','O');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(3,9,'88','O');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(3,10,'and','O');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(3,11,'irregularly','O');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(3,12,'irregular','O');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(3,13,'_','O');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(3,14,'temperature','O');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(3,15,'100.6','O');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(3,16,'_','O');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(3,17,'pulse','O');
```

```
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(3,18,'106','O');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(3,19,'_','O');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(3,20,'respiratory','O');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(3,21,'rate','O');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(3,22,'22','O');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(3,23,'_','O');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(3,24,'blood','O');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(3,25,'pressure','O');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(3,26,'108g64','O');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(3,27,'98%','OxygenSaturation');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(3,28,'saturation','OxygenSaturation');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(3,29,'on','OxygenSaturation');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(3,30,'2','OxygenSaturation');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(3,31,'liters','OxygenSaturation');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(3,32,'of','OxygenSaturation');
INSERT INTO PAL_CRF_TRAIN_TBL VALUES(3,33,'oxygen','OxygenSaturation');
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 1.0, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('ENET_LAMBDA', NULL, 0.1, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MAX_ITERATION', 1000, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('EXIT_THRESHOLD', NULL, 1e-4, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('WORD_SHAPE', 0, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('LBFGS_M', 25, NULL, NULL);
DROP TABLE PAL_CRF_MODEL;
CREATE COLUMN TABLE PAL_CRF_MODEL("ID" INTEGER, "CONTENT" NCLOB);
DROP TABLE PAL_CRF_STATS;
CREATE COLUMN TABLE PAL_CRF_STATS("NAME" VARCHAR(100), "VALUE" VARCHAR(1000));
CALL _SYS_AFL.PAL_CRF(PAL_CRF_TRAIN_TBL, "#PAL_PARAMETER_TBL", PAL_CRF_MODEL,
                      PAL_CRF_STATS, ?)with overview;

select * from PAL_CRF_MODEL;
select * from PAL_CRF_STATS;
```

### Expected Result

MODEL:

| ID | CONTENT |
|---|---|
| 0 | {"classIndex":[["O","OxygenSaturation"]],"defaultSymbol":"O","disJunctionWidth":4,"edgeLabel":[[0,0],[1,0],[0,1],[1,1]],"featureIndex":[["syst... |

STATISTIC:

| | NAME | VALUE |
|---|---|---|
| 1 | obj | 0.44251900977373015 |
| 2 | iter | 22 |
| 3 | solut... | Converged |
| 4 | num... | 2 |
| 5 | num... | 92 |
| 6 | num... | 963 |
| 7 | iter 1 | obj=26.6557 |
| 8 | iter 2 | obj=14.8484 |
| 9 | iter 3 | obj=5.36967 |
| 10 | iter 4 | obj=2.4382 |
| 11 | iter 5 | obj=1.80108 |
| 12 | iter 6 | obj=1.24094 |
| 13 | iter 7 | obj=0.836052 |
| 14 | iter 8 | obj=0.584655 |
| 15 | iter 9 | obj=0.495138 |
| 16 | iter 10 | obj=0.463937 |
| 17 | iter 11 | obj=0.453706 |
| 18 | iter 12 | obj=0.447952 |
| 19 | iter 13 | obj=0.443964 |
| 20 | iter 14 | obj=0.442732 |
| 21 | iter 15 | obj=0.44254 |
| 22 | iter 16 | obj=0.442523 |
| 23 | iter 17 | obj=0.442519 |
| 24 | iter 18 | obj=0.442519 |
| 25 | iter 19 | obj=0.442519 |
| 26 | iter 20 | obj=0.442519 |
| 27 | iter 21 | obj=0.442519 |

## _SYS_AFL.PAL_CRF_INFERENCE

This function predicts based on CRF models.

**Overview of All Tables**

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<Model table>` | IN |
| 3 | `<PARAMETER table>` | IN |
| 4 | `<Result table>` | OUT |

### Input Tables

| Table | Column | Reference for Output Table | Column Data Type | Description |
|---|---|---|---|---|
| DATA | 1st column | DOCUMENT_ID | INTEGER, VRACHAR, or NVARCHAR | Document ID |
| | 2nd column | WORD_POSITION | INTEGER | Word position |
| | 3rd column | | VARCHAR or NVARCHAR | Word |
| MODEL | 1st column | | INTEGER | ID |
| | 2nd column | | NCLOB | |

### Parameter Table

Mandatory Parameters

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL uses its default value.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| THREAD_RATIO | DOUBLE | 1.0 | Specifies the ratio of total number of threads that can be used by this function. The range of this parameter is from 0 to 1, where 0 means only using 1 thread, and 1 means using at most all the currently available threads. Values outside this range are ignored and this function heuristically determines the number of threads to use. | |

### Output Tables

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| RESULT | 1st column | DOCUMENT_ID (in input table) data type | DOCUMENT_ID (in input table) column name | |
| | 2nd column | WORD_POSITION (in input table) data type | WORD_POSITION (in input table) column name | |
| | 3rd column | VARCHAR or NVARCHAR | LABEL | |

### Examples

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

Example:

```
SET SCHEMA DM_PAL;
--prepare inference data table
DROP TABLE PAL_CRF_INFERENCE_TBL;
CREATE COLUMN TABLE PAL_CRF_INFERENCE_TBL("DOC_ID" INTEGER, "WORD_POSITION"
INTEGER, "WORD" VARCHAR(1000));
INSERT INTO PAL_CRF_INFERENCE_TBL VALUES(2,1,'GENERAL');
INSERT INTO PAL_CRF_INFERENCE_TBL VALUES(2,2,'PHYSICAL');
INSERT INTO PAL_CRF_INFERENCE_TBL VALUES(2,3,'EXAMINATION');
INSERT INTO PAL_CRF_INFERENCE_TBL VALUES(2,4,':');
INSERT INTO PAL_CRF_INFERENCE_TBL VALUES(2,5,'VITAL');
```

```
INSERT INTO PAL_CRF_INFERENCE_TBL VALUES(2,6,'SIGNS');
INSERT INTO PAL_CRF_INFERENCE_TBL VALUES(2,7,':');
INSERT INTO PAL_CRF_INFERENCE_TBL VALUES(2,8,'Blood');
INSERT INTO PAL_CRF_INFERENCE_TBL VALUES(2,9,'pressure');
INSERT INTO PAL_CRF_INFERENCE_TBL VALUES(2,10,'86g52');
INSERT INTO PAL_CRF_INFERENCE_TBL VALUES(2,11,'_');
INSERT INTO PAL_CRF_INFERENCE_TBL VALUES(2,12,'heart');
INSERT INTO PAL_CRF_INFERENCE_TBL VALUES(2,13,'rate');
INSERT INTO PAL_CRF_INFERENCE_TBL VALUES(2,14,'132');
INSERT INTO PAL_CRF_INFERENCE_TBL VALUES(2,15,'and');
INSERT INTO PAL_CRF_INFERENCE_TBL VALUES(2,16,'irregular');
INSERT INTO PAL_CRF_INFERENCE_TBL VALUES(2,17,'_');
INSERT INTO PAL_CRF_INFERENCE_TBL VALUES(2,18,'temperature');
INSERT INTO PAL_CRF_INFERENCE_TBL VALUES(2,19,'in');
INSERT INTO PAL_CRF_INFERENCE_TBL VALUES(2,20,'the');
INSERT INTO PAL_CRF_INFERENCE_TBL VALUES(2,21,'emergency');
INSERT INTO PAL_CRF_INFERENCE_TBL VALUES(2,22,'department');
INSERT INTO PAL_CRF_INFERENCE_TBL VALUES(2,23,'on');
INSERT INTO PAL_CRF_INFERENCE_TBL VALUES(2,24,'admission');
INSERT INTO PAL_CRF_INFERENCE_TBL VALUES(2,25,'100.8');
INSERT INTO PAL_CRF_INFERENCE_TBL VALUES(2,26,'PHYSICAL');
INSERT INTO PAL_CRF_INFERENCE_TBL VALUES(2,27,'EXAMINATION');
INSERT INTO PAL_CRF_INFERENCE_TBL VALUES(2,28,':');
INSERT INTO PAL_CRF_INFERENCE_TBL VALUES(2,29,'VITAL');
INSERT INTO PAL_CRF_INFERENCE_TBL VALUES(2,30,'SIGNS');
INSERT INTO PAL_CRF_INFERENCE_TBL VALUES(2,31,':');
INSERT INTO PAL_CRF_INFERENCE_TBL VALUES(2,32,'Blood');
INSERT INTO PAL_CRF_INFERENCE_TBL VALUES(2,33,'pressure');
INSERT INTO PAL_CRF_INFERENCE_TBL VALUES(2,34,'194g100');
INSERT INTO PAL_CRF_INFERENCE_TBL VALUES(2,35,'_');
INSERT INTO PAL_CRF_INFERENCE_TBL VALUES(2,36,'heart');
INSERT INTO PAL_CRF_INFERENCE_TBL VALUES(2,37,'rate');
INSERT INTO PAL_CRF_INFERENCE_TBL VALUES(2,38,'98');
INSERT INTO PAL_CRF_INFERENCE_TBL VALUES(2,39,'_');
INSERT INTO PAL_CRF_INFERENCE_TBL VALUES(2,40,'oxygen');
INSERT INTO PAL_CRF_INFERENCE_TBL VALUES(2,41,'saturation');
INSERT INTO PAL_CRF_INFERENCE_TBL VALUES(2,42,'99%');
INSERT INTO PAL_CRF_INFERENCE_TBL VALUES(2,43,'on');
INSERT INTO PAL_CRF_INFERENCE_TBL VALUES(2,44,'2');
INSERT INTO PAL_CRF_INFERENCE_TBL VALUES(2,45,'liters');
INSERT INTO PAL_CRF_INFERENCE_TBL VALUES(2,46,'.');
INSERT INTO PAL_CRF_INFERENCE_TBL VALUES(3,1,'GENERAL');
INSERT INTO PAL_CRF_INFERENCE_TBL VALUES(3,2,'patient');
INSERT INTO PAL_CRF_INFERENCE_TBL VALUES(3,3,'PHYSICAL');
INSERT INTO PAL_CRF_INFERENCE_TBL VALUES(3,4,'EXAMINATION');
INSERT INTO PAL_CRF_INFERENCE_TBL VALUES(3,5,':');
INSERT INTO PAL_CRF_INFERENCE_TBL VALUES(3,6,'VITAL');
INSERT INTO PAL_CRF_INFERENCE_TBL VALUES(3,7,'SIGNS');
INSERT INTO PAL_CRF_INFERENCE_TBL VALUES(3,8,':');
INSERT INTO PAL_CRF_INFERENCE_TBL VALUES(3,9,'Blood');
INSERT INTO PAL_CRF_INFERENCE_TBL VALUES(3,10,'pressure');
INSERT INTO PAL_CRF_INFERENCE_TBL VALUES(3,11,'156g80');
INSERT INTO PAL_CRF_INFERENCE_TBL VALUES(3,12,'_');
INSERT INTO PAL_CRF_INFERENCE_TBL VALUES(3,13,'pulse');
INSERT INTO PAL_CRF_INFERENCE_TBL VALUES(3,14,'112');
INSERT INTO PAL_CRF_INFERENCE_TBL VALUES(3,15,'per');
INSERT INTO PAL_CRF_INFERENCE_TBL VALUES(3,16,'minute');
INSERT INTO PAL_CRF_INFERENCE_TBL VALUES(3,17,'_');
INSERT INTO PAL_CRF_INFERENCE_TBL VALUES(3,18,'respiratory');
INSERT INTO PAL_CRF_INFERENCE_TBL VALUES(3,19,'rate');
INSERT INTO PAL_CRF_INFERENCE_TBL VALUES(3,20,'21');
INSERT INTO PAL_CRF_INFERENCE_TBL VALUES(3,21,'per');
INSERT INTO PAL_CRF_INFERENCE_TBL VALUES(3,22,'minute');
INSERT INTO PAL_CRF_INFERENCE_TBL VALUES(3,23,'_');
INSERT INTO PAL_CRF_INFERENCE_TBL VALUES(3,24,'temperature');
INSERT INTO PAL_CRF_INFERENCE_TBL VALUES(3,25,'98.6');
INSERT INTO PAL_CRF_INFERENCE_TBL VALUES(3,26,'and');
INSERT INTO PAL_CRF_INFERENCE_TBL VALUES(3,27,'oxygen');
```

```
INSERT INTO PAL_CRF_INFERENCE_TBL VALUES(3,28,'saturation');
INSERT INTO PAL_CRF_INFERENCE_TBL VALUES(3,29,'94%');
INSERT INTO PAL_CRF_INFERENCE_TBL VALUES(3,30,'on');
INSERT INTO PAL_CRF_INFERENCE_TBL VALUES(3,31,'room');
INSERT INTO PAL_CRF_INFERENCE_TBL VALUES(3,32,'air');
DROP TABLE PAL_CRF_INFERENCE_RESULT;
CREATE COLUMN TABLE PAL_CRF_INFERENCE_RESULT("DOC_ID" INTEGER, "WORD_POSITION"
INTEGER, "LABEL" VARCHAR(500));
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 1.0, NULL);
CALL _SYS_AFL.PAL_CRF_INFERENCE(PAL_CRF_INFERENCE_TBL, PAL_CRF_MODEL,
"#PAL_PARAMETER_TBL",
                               PAL_CRF_INFERENCE_RESULT)WITH overview;

SELECT * FROM PAL_CRF_INFERENCE_RESULT;
```

### Expected Result

Result:

| | DOC_ID | WORD_POSITION | LABEL |
|---|---|---|---|
| 1 | 2 | 1 | O |
| 2 | 2 | 2 | O |
| 3 | 2 | 3 | O |
| 4 | 2 | 4 | O |
| 5 | 2 | 5 | O |
| 6 | 2 | 6 | O |
| 7 | 2 | 7 | O |
| 8 | 2 | 8 | O |
| 9 | 2 | 9 | O |
| 10 | 2 | 10 | O |
| 11 | 2 | 11 | O |
| 12 | 2 | 12 | O |
| 13 | 2 | 13 | O |
| 14 | 2 | 14 | O |
| 15 | 2 | 15 | O |
| 16 | 2 | 16 | O |
| 17 | 2 | 17 | O |
| 18 | 2 | 18 | O |
| 19 | 2 | 19 | O |
| 20 | 2 | 20 | O |
| 21 | 2 | 21 | O |
| 22 | 2 | 22 | O |
| 23 | 2 | 23 | O |
| 24 | 2 | 24 | O |
| 25 | 2 | 25 | O |
| 26 | 2 | 26 | O |
| 27 | 2 | 27 | O |
| 28 | 2 | 28 | O |
| 29 | 2 | 29 | O |
| 30 | 2 | 30 | O |
| 31 | 2 | 31 | O |
| 32 | 2 | 32 | O |
| 33 | 2 | 33 | O |
| 34 | 2 | 34 | O |
| 35 | 2 | 35 | O |
| 36 | 2 | 36 | O |
| 37 | 2 | 37 | O |
| 38 | 2 | 38 | O |
| 39 | 2 | 39 | O |

## 5.14.2 Latent Dirichlet Allocation

Latent Dirichlet allocation (LDA) is a generative model in which each item (word) of a collection (document) is generated from a finite mixture over several latent groups (topics).

In the context of text modeling, it posits that each document in the text corpora consists of several topics with different probabilities and each word belongs to certain topics with different probabilities. In PAL, the parameter inference is done via Gibbs sampling.

### _SYS_AFL.PAL_LATENT_DIRICHLET_ALLOCATION

This function performs LDA estimation.

**Overview of All Tables**

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<DOCUMENT_TOPIC_DISTRIBUTION table>` | OUT |
| 4 | `<WORD_TOPIC_ASSIGNMENT table>` | OUT |
| 5 | `<TOPIC_TOP_WORDS table>` | OUT |
| 6 | `<TOPIC_WORD_DISTRIBUTION table>` | OUT |
| 7 | `<DICTIONARY table>` | OUT |
| 8 | `<STATISTICS table>` | OUT |
| 9 | `<CV_PARAMETER table>` | OUT |

**Input Table**

| Table | Column | Reference for Output Table | Data Type | Description |
|---|---|---|---|---|
| DATA | 1st column | DOCUMENT_ID | INTEGER, VARCHAR, or NVARCHAR | Document ID |
| | 2nd column | | VARCHAR or NVARCHAR | Document texts separated by certain delimit |

### Parameter Table

Mandatory Parameter

The following parameter is mandatory and must be given a value.

| Name | Data Type | Description |
|---|---|---|
| TOPICS | INTEGER | Expected number of topics in the corpus |

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| ALPHA | DOUBLE | 50/(value of parameter TOPICS) | Specifies the prior weight related to document-topic distribution | |
| BETA | DOUBLE | 0.1 | Specifies the prior weight related to topic-word distribution | |
| BURNIN | INTEGER | 0 | Number of omitted Gibbs iterations at the beginning | |
| ITERATION | INTEGER | 2000 | Number of Gibbs iterations | |
| THIN | INTEGER | 1 | Number of omitted in-between Gibbs iterations | |
| SEED | INTEGER | 0 | Indicates the seed used to initialize the random number generator.<br>• 0: uses the system time<br>• Not 0: uses the specified seed | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| MAX_TOP_WORDS | INTEGER | 0 | Specifies the maximum number of words to be output for each topic. | Only valid when the topic top words output table is provided. It cannot be used together with parameter THRESHOLD_TOP_WORDS. |
| THRESHOLD_TOP_WORDS | DOUBLE | 0 | The algorithm outputs top words for each topic if the probability is larger than this threshold. | Only valid when the topic top words output table is provided. It cannot be used together with parameter MAX_TOP_WORDS. |
| INIT | INTEGER | 0 | Specifies initialization method for Gibbs sampling:<br>• 0: Uniform. Assigns each word in each document a topic by a uniform distribution. Each topic has the same probability to be assigned for each word.<br>• 1: Initialization by Gibbs sampling. Assigns each word in each document a topic by one round of Gibbs sampling using the prior distribution of document-topic and topic-word given by parameters ALPHA and BETA. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| DELIMIT | VARCHAR | ' ' | Specifies the delimit to separate words in a document.<br><br>For example, if the words are separated by `,` and `:`, then the delimit can be `,:`.<br><br>For example, if the words are separated by `,` or `:`, then the delimit should be `','` or `':'`. | |
| OUTPUT_WORD_ASSIGNMENT | INTEGER | 0 | Controls whether to output the word-topic assignment or not. Note that if this parameter is set to 1, the procedure would take more time to return to write the WORD_TOPIC_ASSIGNMENT table. | |

## Output Tables

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| DOCUMENT_TOPIC_DISTRIBUTION | 1st column | DOCUMENT_ID (in input table) data type | DOCUMENT_ID (in input table) column name | Document ID |
| | 2nd column | INTEGER | TOPIC_ID | Topic ID |
| | 3rd column | DOUBLE | PROBABILITY | Probability of topic given document |
| WORD_TOPIC_ASSIGNMENT | 1st column | DOCUMENT_ID (in input table) data type | DOCUMENT_ID (in input table) column name | Document ID |
| | 2nd column | INTEGER | WORD_ID | Word ID |
| | 3rd column | INTEGER | TOPIC_ID | Topic ID |

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| TOPIC_TOP_WORDS | 1st column | INTEGER | TOPIC_ID | Topic ID |
| | 2nd column | NVARCHAR(5000) | WORDS | Topic top words separated by space |
| TOPIC_WORD_DISTRIBUTION | 1st column | INTEGER | TOPIC_ID | Topic ID |
| | 2nd column | INTEGER | WORD_ID | Word ID |
| | 3rd column | DOUBLE | PROBABILITY | Probability of word given topic |
| DICTIONARY | 1st column | INTEGER | WORD_ID | Word ID |
| | 2nd column | NVARCHAR(5000) | WORD | Word text |
| STATISTICS | 1st column | NVARCHAR(256) | STAT_NAME | Statistics name |
| | 2nd column | NVARCHAR(1000) | STAT_VALUE | Statistics value |
| CV_PARAMETER | 1st column | NVARCHAR(256) | PARAM_NAME | Parameter name |
| | 2nd column | INTEGER | INT_VALUE | INTEGER value |
| | 3rd column | DOUBLE | DOUBLE_VALUE | DOUBLE value |
| | 4th column | NVARCHAR(1000) | STRING_VALUE | STRING value |

### Example

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_LATENT_DIRICHLET_ALLOCATION_DATA_TBL;
CREATE COLUMN TABLE PAL_LATENT_DIRICHLET_ALLOCATION_DATA_TBL ("DOCUMENT_ID"
INTEGER, "TEXT" NVARCHAR (5000));
INSERT INTO PAL_LATENT_DIRICHLET_ALLOCATION_DATA_TBL VALUES (10 , 'cpu harddisk
graphiccard cpu monitor keyboard cpu memory memory');
INSERT INTO PAL_LATENT_DIRICHLET_ALLOCATION_DATA_TBL VALUES (20 , 'tires
mountainbike wheels valve helmet mountainbike rearfender tires mountainbike
mountainbike');
INSERT INTO PAL_LATENT_DIRICHLET_ALLOCATION_DATA_TBL VALUES (30 , 'carseat toy
strollers toy toy spoon toy strollers toy carseat');
INSERT INTO PAL_LATENT_DIRICHLET_ALLOCATION_DATA_TBL VALUES (40 , 'sweaters
sweaters sweaters boots sweaters rings vest vest shoe sweaters');
DROP TABLE PAL_LDA_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_LDA_PARAMETER_TBL ("PARAM_NAME" VARCHAR(256),
"INT_VALUE" INTEGER, "DOUBLE_VALUE" DOUBLE, "STRING_VALUE" VARCHAR(1000));
INSERT INTO PAL_LDA_PARAMETER_TBL VALUES ('TOPICS', 6, NULL, NULL);
INSERT INTO PAL_LDA_PARAMETER_TBL VALUES ('BURNIN', 50, NULL, NULL);
INSERT INTO PAL_LDA_PARAMETER_TBL VALUES ('THIN', 10, NULL, NULL);
INSERT INTO PAL_LDA_PARAMETER_TBL VALUES ('ITERATION', 100, NULL, NULL);
```

```
INSERT INTO PAL_LDA_PARAMETER_TBL VALUES ('SEED', 1, NULL, NULL);
INSERT INTO PAL_LDA_PARAMETER_TBL VALUES ('ALPHA', NULL, 0.1, NULL);
INSERT INTO PAL_LDA_PARAMETER_TBL VALUES ('MAX_TOP_WORDS', 5, NULL, NULL);
INSERT INTO PAL_LDA_PARAMETER_TBL VALUES ('OUTPUT_WORD_ASSIGNMENT', 1, NULL,
NULL);
INSERT INTO PAL_LDA_PARAMETER_TBL VALUES ('DELIMIT', NULL, NULL, ' '||char(13)||
char(10));
DROP TABLE PAL_LDA_TOPIC_WORD_DISTRIBUTION_TBL;
DROP TABLE PAL_LDA_DICTIONARY_TBL;
DROP TABLE PAL_LDA_CV_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_LDA_TOPIC_WORD_DISTRIBUTION_TBL ("TOPIC_ID" INTEGER,
"WORD_ID" INTEGER, "PROBABILITY" DOUBLE);
CREATE COLUMN TABLE PAL_LDA_DICTIONARY_TBL ("WORD_ID" INTEGER, "WORD"
NVARCHAR(5000));
CREATE COLUMN TABLE PAL_LDA_CV_PARAMETER_TBL ("PARAM_NAME" VARCHAR(256),
"INT_VALUE" INTEGER, "DOUBLE_VALUE" DOUBLE, "STRING_VALUE" VARCHAR(1000));
DO BEGIN
lt_data = SELECT * FROM PAL_LATENT_DIRICHLET_ALLOCATION_DATA_TBL;
lt_param = SELECT * FROM PAL_LDA_PARAMETER_TBL;
CALL _SYS_AFL.PAL_LATENT_DIRICHLET_ALLOCATION (:lt_data, :lt_param,
lt_document_topic_distribution, lt_word_topic_assignment, lt_topic_top_words,
lt_topic_word_distribution, lt_dictionary, lt_statistics, lt_cv_parameter);
INSERT INTO PAL_LDA_TOPIC_WORD_DISTRIBUTION_TBL SELECT *
FROM :lt_topic_word_distribution;
INSERT INTO PAL_LDA_DICTIONARY_TBL SELECT * FROM :lt_dictionary;
INSERT INTO PAL_LDA_CV_PARAMETER_TBL SELECT * FROM :lt_cv_parameter;
END;
```

### Expected Results

DOCUMENT_TOPIC_DISTRIBUTION:

| | DOCUMENT_ID | TOPIC_ID | PROBABILITY |
|---|---|---|---|
| 1 | 10 | 0 | 0.010416666666666668 |
| 2 | 10 | 1 | 0.010416666666666668 |
| 3 | 10 | 2 | 0.010416666666666668 |
| 4 | 10 | 3 | 0.010416666666666668 |
| 5 | 10 | 4 | 0.9479166666666666 |
| 6 | 10 | 5 | 0.010416666666666668 |
| 7 | 20 | 0 | 0.009433962264150945 |
| 8 | 20 | 1 | 0.009433962264150945 |
| 9 | 20 | 2 | 0.009433962264150945 |
| 10 | 20 | 3 | 0.9528301886792453 |
| 11 | 20 | 4 | 0.009433962264150945 |
| 12 | 20 | 5 | 0.009433962264150945 |
| 13 | 30 | 0 | 0.1037735849056604 |
| 14 | 30 | 1 | 0.8584905660377359 |
| 15 | 30 | 2 | 0.009433962264150945 |
| 16 | 30 | 3 | 0.009433962264150945 |
| 17 | 30 | 4 | 0.009433962264150945 |
| 18 | 30 | 5 | 0.009433962264150945 |
| 19 | 40 | 0 | 0.009433962264150945 |
| 20 | 40 | 1 | 0.009433962264150945 |
| 21 | 40 | 2 | 0.9528301886792453 |
| 22 | 40 | 3 | 0.009433962264150945 |

WORD_TOPIC_ASSIGNMENT:

| | DOCUMENT_ID | WORD_ID | TOPIC_ID |
|---|---|---|---|
| 1 | 10 | 0 | 4 |
| 2 | 10 | 1 | 4 |
| 3 | 10 | 2 | 4 |
| 4 | 10 | 0 | 4 |
| 5 | 10 | 3 | 4 |
| 6 | 10 | 4 | 4 |
| 7 | 10 | 0 | 4 |
| 8 | 10 | 5 | 4 |
| 9 | 10 | 5 | 4 |
| 10 | 20 | 6 | 3 |
| 11 | 20 | 7 | 3 |
| 12 | 20 | 8 | 3 |
| 13 | 20 | 9 | 3 |
| 14 | 20 | 10 | 3 |
| 15 | 20 | 7 | 3 |
| 16 | 20 | 11 | 3 |
| 17 | 20 | 6 | 3 |
| 18 | 20 | 7 | 3 |
| 19 | 20 | 7 | 3 |

TOPIC_TOP_WORDS:

| | TOPIC_ID | WORDS |
|---|---|---|
| 1 | 0 | spoon strollers tires graphiccard valve |
| 2 | 1 | toy strollers carseat graphiccard cpu |
| 3 | 2 | sweaters vest shoe rings boots |
| 4 | 3 | mountainbike tires rearfender helmet valve |
| 5 | 4 | cpu memory graphiccard keyboard harddisk |
| 6 | 5 | strollers tires graphiccard cpu valve |

TOPIC_WORD_DISTRIBUTION:

| | TOPIC_ID | WORD_ID | PROBABILITY |
|---|---|---|---|
| 1 | 0 | 0 | 0.03225806451612903 |
| 2 | 0 | 1 | 0.03225806451612903 |
| 3 | 0 | 2 | 0.03225806451612903 |
| 4 | 0 | 3 | 0.03225806451612903 |
| 5 | 0 | 4 | 0.03225806451612903 |
| 6 | 0 | 5 | 0.03225806451612903 |
| 7 | 0 | 6 | 0.03225806451612903 |
| 8 | 0 | 7 | 0.03225806451612903 |
| 9 | 0 | 8 | 0.03225806451612903 |
| 10 | 0 | 9 | 0.03225806451612903 |
| 11 | 0 | 10 | 0.03225806451612903 |
| 12 | 0 | 11 | 0.03225806451612903 |
| 13 | 0 | 12 | 0.03225806451612903 |
| 14 | 0 | 13 | 0.03225806451612903 |
| 15 | 0 | 14 | 0.03225806451612903 |
| 16 | 0 | 15 | 0.3548387096774194 |
| 17 | 0 | 16 | 0.03225806451612903 |
| 18 | 0 | 17 | 0.03225806451612903 |
| 19 | 0 | 18 | 0.03225806451612903 |
| 20 | 0 | 19 | 0.03225806451612903 |
| 21 | 0 | 20 | 0.03225806451612903 |

DICTIONARY:

| | WORD_ID | WORD |
|---|---|---|
| 1 | 17 | boots |
| 2 | 12 | carseat |
| 3 | 0 | cpu |
| 4 | 2 | graphiccard |
| 5 | 1 | harddisk |
| 6 | 10 | helmet |
| 7 | 4 | keyboard |
| 8 | 5 | memory |
| 9 | 3 | monitor |
| 10 | 7 | mountainbike |
| 11 | 11 | rearfender |
| 12 | 18 | rings |
| 13 | 20 | shoe |
| 14 | 15 | spoon |
| 15 | 14 | strollers |
| 16 | 16 | sweaters |
| 17 | 6 | tires |
| 18 | 13 | toy |
| 19 | 9 | valve |
| 20 | 19 | vest |
| 21 | 8 | wheels |

STATISTICS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | DOCUMENTS | 4 |
| 2 | VOCABULARY_SIZE | 21 |
| 3 | LOG_LIKELIHOOD | -64.95765414596762 |

CV_PARAMETER:

| | PARAM_NAME | INT_VALUE | DOUBLE_VALUE | STRING_VALUE |
|---|---|---|---|---|
| 1 | TOPICS | 6 | ? | ? |
| 2 | BURNIN | 50 | ? | ? |
| 3 | THIN | 10 | ? | ? |
| 4 | SEED | 1 | ? | ? |
| 5 | ITERATION | 100 | ? | ? |
| 6 | INIT | 0 | ? | ? |
| 7 | ALPHA | ? | 0.1 | ? |
| 8 | BETA | ? | 0.1 | ? |
| 9 | DELIMIT | ? | ? | |

## _SYS_AFL.PAL_LATENT_DIRICHLET_ALLOCATION_INFERENCE

This function inferences the topic assignment for new documents based on the previous LDA estimation results.

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<TOPIC_WORD_DISTRIBUTION table>` | IN |
| 3 | `<DICTIONARY table>` | IN |
| 4 | `<CV_PARAMETER table>` | IN |
| 5 | `<PARAMETER table>` | IN |
| 6 | `<DOCUMENT_TOPIC_DISTRIBUTION table>` | OUT |
| 7 | `<WORD_TOPIC_ASSIGNMENT table>` | OUT |
| 8 | `<STATISTICS table>` | OUT |

### Input Tables

| Table | Column | Reference for Output Table | Data Type | Description |
|---|---|---|---|---|
| DATA | 1st column | DOCUMENT_ID | INTEGER, VARCHAR, or NVARCHAR | Document ID |
| | 2nd column | | VARCHAR or NVARCHAR | Document texts separated by certain delimit |
| TOPIC_WORD_DISTRIBUTION | 1st column | | INTEGER | Topic ID |
| | 2nd column | | INTEGER | Word ID |
| | 3rd column | | DOUBLE or DECIMAL(p, s) | Probability of word given topic |
| DICTIONARY | 1st column | | INTEGER | Word ID |
| | 2nd column | | NVARCHAR | Word text |
| CV_PARAMETER | 1st column | | NVARCHAR | Parameter name |
| | 2nd column | | INTEGER | INTEGER value |
| | 3rd column | | DOUBLE | DOUBLE value |
| | 4th column | | NVARCHAR | STRING value |

### Parameter Table

Mandatory Parameter

None.

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| BURNIN | INTEGER | 0 | Number of omitted Gibbs iterations at the beginning. This value takes precedence over the corresponding one in the general information table. |
| ITERATION | INTEGER | 2000 | Number of Gibbs iterations. This value takes precedence over the corresponding one in the general information table. |
| THIN | INTEGER | 1 | Number of omitted in-between Gibbs iterations. This value takes precedence over the corresponding one in the general information table. |
| SEED | INTEGER | 0 | Indicates the seed used to initialize the random number generator. This value takes precedence over the corresponding one in the general information table.<br>• 0: uses the system time<br>• Not 0: uses the specified seed |
| INIT | INTEGER | 0 | Specifies initialization method for Gibbs sampling. This value takes precedence over the corresponding one in the general information table.<br>• 0: Uniform<br>• 1: Initialization by sampling |

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| DELIMIT | VARCHAR | ' ' | Specifies the delimit to separate each word in the document. This value takes precedence over the corresponding one in the general information table.<br><br>For example, if the words are separated by `,` or `:`, then the delimit should be `','` or `':'`. |
| OUTPUT_WORD_ASSIGNMENT | INTEGER | 0 | Controls whether to output the word-topic assignment or not. Note that if this parameter is set to 1, the procedure would take more time to return to write the WORD_TOPIC_ASSIGNMENT table. |

## Output Tables

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| DOCUMENT_TOPIC_DISTRIBUTION | 1st column | DOCUMENT_ID (in input table) data type | DOCUMENT_ID (in input table) column name | Document ID |
| | 2nd column | INTEGER | TOPIC_ID | Topic ID |
| | 3rd column | DOUBLE or DECIMAL(p, s) | PROBABILITY | Probability of topic given document |
| WORD_TOPIC_ASSIGNMENT | 1st column | DOCUMENT_ID (in input table) data type | DOCUMENT_ID (in input table) column name | Document ID |
| | 2nd column | INTEGER | WORD_ID | Word ID |
| | 3rd column | INTEGER | TOPIC_ID | Topic ID |
| STATISTICS | 1st column | NVARCHAR(256) | STAT_NAME | Statistics name |
| | 2nd column | NVARCHAR(1000) | STAT_VALUE | Statistics value |

## Example

Assume that:

- DM_PAL is a schema belonging to USER1;
- USER1 has been assigned the AFL__SYS_AFL_AFLPAL_EXECUTE or AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION role.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_LATENT_DIRICHLET_ALLOCATION_INFERENCE_DATA_TBL;
CREATE COLUMN TABLE PAL_LATENT_DIRICHLET_ALLOCATION_INFERENCE_DATA_TBL
("DOCUMENT_ID" INTEGER, "TEXT" NVARCHAR (5000));
INSERT INTO PAL_LATENT_DIRICHLET_ALLOCATION_INFERENCE_DATA_TBL VALUES (10, 'toy
toy spoon cpu');
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL ("PARAM_NAME"
VARCHAR(256), "INT_VALUE" INTEGER, "DOUBLE_VALUE" DOUBLE, "STRING_VALUE"
VARCHAR(1000));
INSERT INTO #PAL_PARAMETER_TBL VALUES ('BURNIN', 2000, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('THIN', 100, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('ITERATION', 1000, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SEED', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('OUTPUT_WORD_ASSIGNMENT', 1, NULL, NULL);
CALL _SYS_AFL.PAL_LATENT_DIRICHLET_ALLOCATION_INFERENCE
(PAL_LATENT_DIRICHLET_ALLOCATION_INFERENCE_DATA_TBL,
PAL_LDA_TOPIC_WORD_DISTRIBUTION_TBL, PAL_LDA_DICTIONARY_TBL,
PAL_LDA_CV_PARAMETER_TBL, #PAL_PARAMETER_TBL, ?, ?, ?);
```

**Expected Result**

DOCUMENT_TOPIC_DISTRIBUTION:

| | DOCUMENT_ID | TOPIC_ID | PROBABILITY |
|---|---|---|---|
| 1 | 10 | 0 | 0.23913043478260873 |
| 2 | 10 | 1 | 0.4565217391304348 |
| 3 | 10 | 2 | 0.02173913043478261 |
| 4 | 10 | 3 | 0.02173913043478261 |
| 5 | 10 | 4 | 0.23913043478260873 |
| 6 | 10 | 5 | 0.02173913043478261 |

WORD_TOPIC_ASSIGNMENT:

| | DOCUMENT_ID | WORD_ID | TOPIC_ID |
|---|---|---|---|
| 1 | 10 | 13 | 1 |
| 2 | 10 | 13 | 1 |
| 3 | 10 | 15 | 0 |
| 4 | 10 | 0 | 4 |

STATISTICS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | DOCUMENTS | 1 |
| 2 | VOCABULARY_SIZE | 21 |
| 3 | LOG_LIKELIHOOD | -7.9250929918753... |
| 4 | PERPLEXITY | 7.251970666272191 |

> **Note**
>
> The wrapper procedure generation approach to call PAL functions is still supported. For more information, refer to the older versions (prior to SAP HANA Platform 2.0 SPS 02) of this document.

## 5.14.3 TF-IDF Calculation

TFIDF, short for term frequency–inverse document frequency, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus.

TFIDF is calculated by multiplying two metrics:

- the number of times a word appears in a document
- the inverse document frequency of the word across a set of documents

The TF-IDF value increases proportionally to the number of times a word appears in the document but is offset by the number of documents in the corpus that contain the word, it helps to adjust for the fact that some words appear more frequently in general. This function supports English, German, Spanish, French and Russian.

### Prerequisites

The first column of the input document table is an ID column.

### _SYS_AFL.PAL_TEXT_COLLECT

Its use is primarily compute inverse document frequency of documents which provided by user. The results analysis is used as input for PAL_TEXT_TFIDF.

**Overview of All Tables**

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<PARAMETER table>` | IN |
| 3 | `<TERM-IDF Result Table>` | OUT |
| 4 | `<EXTEND_OUT Table>` | OUT |

**Input Table:**

| Table | Column | Reference | Data Type | Description |
|---|---|---|---|---|
| DATA | 1st column | ID | INTEGER, BIGINT, VARCHAR or NVARCHAR | The ID column that indicates the document No.This must be the first column. |
| | 2nd column | | VARCHAR or NVARCHAR | Document content |

**Parameter Table**

Mandatory parameters

None.

Optional Parameters

The following parameter is optional. If not specified, PAL will use its default value.

| Name | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| LANGUAGE | NVARCHAR | Auto detection | Language of document<br>• EN<br>• DE<br>• ES<br>• FR<br>• RU | |

**Output tables:**

| Table | Column | Column data type | Description |
|---|---|---|---|
| TERM_IDF_Result | 1st column | NVARCHAR | TM_TERM |
| | 2nd column | DOUBLE | IDF_VALUE |

| Table | Column | Column data type | Description |
|---|---|---|---|
| EXTEND_OUT | 1st column | INTEGER | Index |
| | 2nd column | NCLOB | Some info |

**Example**

Assume that:

- DM_PAL is a schema belonging to USER1;
- USER1 has been assigned the AFL__SYS_AFL_AFLPAL_EXECUTE or AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION role.

**Example:**

```
set schema DM_PAL;
drop table PAL_TFIDF_DATA_TAB;
create column table PAL_TFIDF_DATA_TAB (
"ID" nvarchar(1000),
"CONTENT" nvarchar(1000)
);
INSERT INTO PAL_TFIDF_DATA_TAB VALUES('doc1','term1 term2 term2 term3 term3
term3');
INSERT INTO PAL_TFIDF_DATA_TAB VALUES('doc2','term2 term3 term3 term4 term4
term4');
INSERT INTO PAL_TFIDF_DATA_TAB VALUES('doc3','term3 term4 term4 term5 term5
term5');
INSERT INTO PAL_TFIDF_DATA_TAB VALUES('doc5','term3 term4 term4 term5 term5
term5 term5 term5 term5');
INSERT INTO PAL_TFIDF_DATA_TAB VALUES('doc4','term4 term6');
INSERT INTO PAL_TFIDF_DATA_TAB VALUES('doc6','term4 term6 term6 term6');
drop table PAL_PARAMETER_TAB;
create column table PAL_PARAMETER_TAB (
"PARAM_NAME" nvarchar(256),
"INT_VALUE" integer,
"DOUBLE_VALUE" double,
"STRING_VALUE" nvarchar(1000)
);
```

```
drop table PAL_TM_TERM_IDF_TAB;
create column table PAL_TM_TERM_IDF_TAB ("TM_TERMS" nvarchar(1000),"IDF_VALUE"
double);
drop table PAL_TM_EXTEND_OUT_TAB;
create column table PAL_TM_EXTEND_OUT_TAB ("INDEX" integer,"INFO" NCLOB);
DO BEGIN
    lt_data = SELECT * FROM PAL_TFIDF_DATA_TAB;
    lt_param = SELECT * FROM PAL_PARAMETER_TAB;
    CALL _SYS_AFL.PAL_TEXT_COLLECT(:lt_data,:lt_param,lt_termidf,lt_extendout);
    INSERT INTO PAL_TM_TERM_IDF_TAB SELECT * FROM :lt_termidf;
    INSERT INTO PAL_TM_EXTEND_OUT_TAB SELECT * FROM :lt_extendout;
END;
--------------------------------------------
select * from PAL_TM_TERM_IDF_TAB;
select * from PAL_TM_EXTEND_OUT_TAB;
```

**Expected Result**

PAL_TM_TERM_IDF_TAB:

| | TM_TERMS | IDF_VALUE |
|---|---|---|
| 1 | term5 | 1.0986122886681098 |
| 2 | term2 | 1.0986122886681098 |
| 3 | term1 | 1.791759469228055 |
| 4 | term4 | 0.1823215567939546 |
| 5 | term3 | 0.4054651081081644 |
| 6 | term6 | 1.0986122886681098 |

## _SYS_AFL.PAL_TEXT_TFIDF

Its use is primarily compute term frequency - inverse document frequency by document.

**Overview of All Tables**

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | `<DATA table>` | IN |
| 2 | `<TERM-IDF table>` | IN |
| 3 | `<PARAMETER table>` | IN |
| 4 | `<TERM-TFIDF Result Table>` | OUT |

**Input Table:**

| Table | Column | Reference | Data Type | Description |
|---|---|---|---|---|
| DATA | 1st column | ID | INTEGER, BIGINT, VARCHAR or NVARCHAR | The ID column that indicates the document No. This must be the first column. |

| Table | Column | Reference | Data Type | Description |
|---|---|---|---|---|
| | 2nd column | | VARCHAR, NVARCHAR, CLOB or NCLOB | Document content |

| Table | Column | Column data type | Description |
|---|---|---|---|
| TERM_IDF_Result | 1st column | NVARCHAR | TM_TERM |
| | 2rd column | DOUBLE | IDF_VALUE |

### Parameter Table

Mandatory parameters

None.

Optional Parameters

The following parameter is optional. If not specified, PAL will use its default value.

| Name | DATA TYPE | DEFAULT VALUE | DESCRIPTION | DEPENDENCY |
|---|---|---|---|---|
| LANGUAGE | NVARCHAR | Auto detection | Language of document<br>• EN<br>• DE<br>• ES<br>• FR<br>• RU | |

### Output tables:

| Table | Column | Column data type | Description |
|---|---|---|---|
| TERM_TFIDF_Result | 1st column | INTEGER | Doc ID |
| | 2nd column | NVARCHAR | Term |
| | 3rd column | DOUBLE | TF_Value |
| | 4th column | DOUBLE | TFIDF Value |

### Example

Assume that:

- DM_PAL is a schema belonging to USER1;
- USER1 has been assigned the AFL__SYS_AFL_AFLPAL_EXECUTE or AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION role.

**Example:**

```
set schema DM_PAL;
-----------------PAL_TEXT_COLLECT----------------------------
drop table PAL_TFIDF_DATA_TAB;
create column table PAL_TFIDF_DATA_TAB (
"ID" nvarchar(1000),
```

```
"CONTENT" nvarchar(1000)
);
INSERT INTO PAL_TFIDF_DATA_TAB VALUES('doc1','term1 term2 term2 term3 term3
term3');
INSERT INTO PAL_TFIDF_DATA_TAB VALUES('doc2','term2 term3 term3 term4 term4
term4');
INSERT INTO PAL_TFIDF_DATA_TAB VALUES('doc3','term3 term4 term4 term5 term5
term5');
INSERT INTO PAL_TFIDF_DATA_TAB VALUES('doc5','term3 term4 term4 term5 term5
term5 term5 term5 term5');
INSERT INTO PAL_TFIDF_DATA_TAB VALUES('doc4','term4 term6');
INSERT INTO PAL_TFIDF_DATA_TAB VALUES('doc6','term4 term6 term6 term6');
drop table PAL_PARAMETER_TAB;
create column table PAL_PARAMETER_TAB (
"PARAM_NAME" nvarchar(256),
"INT_VALUE" integer,
"DOUBLE_VALUE" double,
"STRING_VALUE" nvarchar(1000)
);
drop table PAL_TM_TERM_IDF_TAB;
create column table PAL_TM_TERM_IDF_TAB ("TM_TERMS" nvarchar(1000),"IDF_VALUE"
double);
drop table PAL_TM_EXTEND_OUT_TAB;
create column table PAL_TM_EXTEND_OUT_TAB ("INDEX" integer,"INFO" NCLOB);
DO BEGIN
lt_data = SELECT * FROM PAL_TFIDF_DATA_TAB;
lt_param = SELECT * FROM PAL_PARAMETER_TAB;
CALL _SYS_AFL.PAL_TEXT_COLLECT(:lt_data,:lt_param,lt_termidf,lt_extendout);
INSERT INTO PAL_TM_TERM_IDF_TAB SELECT * FROM :lt_termidf;
INSERT INTO PAL_TM_EXTEND_OUT_TAB SELECT * FROM :lt_extendout;
END;
-----------------PAL_TEXT_TFIDF----------------------------
drop table PAL_TM_TERM_TFIDF_TAB;
create column table PAL_TM_TERM_TFIDF_TAB (
"ID" nvarchar(1000),
"TERM" nvarchar(1000),
"TF_VALUE" double,
"TFIDF_VALUE" double
);
DO BEGIN
    lt_data = SELECT * FROM PAL_TFIDF_DATA_TAB;
    lt_param = SELECT * FROM PAL_PARAMETER_TAB;
    lt_termidf = SELECT * FROM PAL_TM_TERM_IDF_TAB;
    CALL _SYS_AFL.PAL_TEXT_TFIDF(:lt_data,:lt_termidf,:lt_param,lt_tfidf);
    INSERT INTO PAL_TM_TERM_TFIDF_TAB SELECT * FROM :lt_tfidf;
END;
---------------------------------------------
select * from PAL_TM_TERM_TFIDF_TAB;
```

**Expected Result**

| | ID | TERM | TF_VALUE | TFIDF_VALUE |
|---|---|---|---|---|
| 1 | doc1 | term2 | 2 | 2.1972245773362196 |
| 2 | doc1 | term1 | 1 | 1.791759469228055 |
| 3 | doc1 | term3 | 3 | 1.2163953243244932 |
| 4 | doc2 | term2 | 1 | 1.0986122886681098 |
| 5 | doc2 | term4 | 3 | 0.5469646703818638 |
| 6 | doc2 | term3 | 2 | 0.8109302162163288 |
| 7 | doc3 | term5 | 3 | 3.295836866004329 |
| 8 | doc3 | term4 | 2 | 0.3646431135879092 |
| 9 | doc3 | term3 | 1 | 0.4054651081081644 |
| 10 | doc5 | term5 | 6 | 6.591673732008658 |
| 11 | doc5 | term4 | 2 | 0.3646431135879092 |
| 12 | doc5 | term3 | 1 | 0.4054651081081644 |
| 13 | doc4 | term4 | 1 | 0.1823215567939546 |
| 14 | doc4 | term6 | 1 | 1.0986122886681098 |
| 15 | doc6 | term4 | 1 | 0.1823215567939546 |
| 16 | doc6 | term6 | 3 | 3.295836866004329 |

ⓘ Note

Current release supports English, German, Spanish, French and Russian.

# 6 Model Evaluation and Parameter Selection

You can use some resampling methods to evaluate model performance and perform model selection for the optimal parameters of the model. PAL provides model evaluation and parameter selection for some training algorithms, such as SVM and logistic regression.

PAL provides two kinds of resampling methods:

| | |
|---|---|
| Cross validation | Divides training data into k folds as evenly as possible. Each time select one fold to leave out; use remaining data as training data set; use data in selected fold as evaluation data set. |
| Bootstrap | Randomly sample entities of same quantity from original data set with put-back. Use these entities as training data set; use the remaining unselected data as evaluation data set. |

For classification problems, to make training and evaluation data set have similar class distribution to the original one, both resampling methods provide their stratified version.

To reduce variability, PAL can perform multiple rounds of evaluation and combine the results.

To accelerate the process of parameter selection, a technic called early stop could be used to reduce parameter search space by eliminating not so promising ones with a few resources and focuse on the rest.

- Successive Halving
  A method iteratively eliminates parameter search space by a specific rate and the same time increase the resource used for evaluating until there is only one left. This method is not perfect. At early stage, it eliminates parameters with a few resources to evaluate them and it may discard parameters that could be actually good ones. For a large parameter search space, this method probably could gain a considerably acceleration and give a pretty good result.
- Hyperband
  Hyperband method means to ease the defect of successive halving when it is hard to decide the configuration of search space and resources used. Hyperband method employs successive halving algorithm and detects a good configuration for this purpose.

While activating parameter selection, there are two search strategies:

| | |
|---|---|
| Grid | Each parameter to be selected has determined candidates when it is defined via either value set or range with step. Every possible combination of all parameters is evaluated and the one with best result is chosen. |
| Random | Each parameter to be selected has either a discrete value set or a range. Each time a proper value of each parameter is chosen, and the combination of parameters is evaluated. In this case, you must specify the number of trying rounds. The combination with best result is chosen. |

Algorithms that support model evaluation and parameter selection:

- _SYS_AFL.PAL_FRM
- _SYS_AFL.PAL_ALS
- _SYS_AFL.PAL_LINEAR_REGRESSION
- _SYS_AFL.PAL_POLYNOMIAL_REGRESSION

- _SYS_AFL.PAL_LOGISTIC_REGRESSION
- _SYS_AFL.PAL_MULTICLASS_LOGISTIC_REGRESSION
- _SYS_AFL.PAL_MULTILAYER_PERCEPTRON
- _SYS_AFL.PAL_NAIVE_BAYES
- _SYS_AFL.PAL_DECISION_TREE
- _SYS_AFL.PAL_SVM
- _SYS_AFL.PAL_KNN_CV
- _SYS_AFL.PAL_HGBT
- _SYS_AFL.PAL_GLM

When you enable model evaluation or parameter selection on a specific training algorithm, evaluation results and some extra information are provided in the STATISTIC table. And for parameter selection, selected optimal parameters are written into the last OPTIMAL_PARAM table.

At last a final model is trained using optimal parameters and the whole data set.

## Prerequisites

- Follow prerequisites of corresponding training algorithm.
- Set the RESAMPLING_METHOD parameter to enable model evaluation and parameter selection.
- Set the PARAM_SEARCH_STRATEGY parameter to enable parameter selection.

### Overview of All Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1, 2, ... n | Algorithm specific tables | IN or OUT |
| n + 1 | `<STATISTIC table>` | OUT |
| n + 2 | `<OPTIMAL_PARAM table>` | OUT |

### Parameter Table

Mandatory Parameters

The following parameters are mandatory and must be given a value.

| Name | Data type | Description |
| --- | --- | --- |
| RESAMPLING_METHOD | NVARCHAR | Specifies the resampling method for model evaluation or parameter selection.<br>• cv<br>• cv_sha<br>• cv_hyperband<br>• stratified_cv<br>• stratified_cv_sha<br>• stratified_cv_hyperband<br>• bootstrap<br>• bootstrap_sha<br>• bootstrap_hyperband<br>• stratified_bootstrap<br>• stratified_bootstrap_sha<br>• stratified_bootstrap_hyperband<br>Options like stratified_* can only be applied to classification algorithms. Options like *_sha and *_hyperband can only be used in parameter selection, can not be used in model evaluation.<br>If no value is specified for this parameter, neither model evaluation nor parameter selection is activated. |
| EVALUATION_METRIC | NVARCHAR | Specifies the evaluation metric for model evaluation or parameter selection.<br>• ACCURACY<br>• ERROR_RATE<br>• F1_SCORE<br>• RMSE<br>• MAE<br>• AUC<br>• NLL (negative log likelihood)<br>Some evaluation metrics may not be supported by specific algorithms. Please check the corresponding algorithm section for details. |

Optional Parameters

The following parameters are optional. If a parameter is not specified, PAL will use its default value.

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| FOLD_NUM | INTEGER | No default value | Specifies the fold number for the cross validation method. | Mandatory and valid only when RESAMPLING_METHOD is set to cv or stratified_cv. |
| REPEAT_TIMES | INTEGER | 1 | Specifies the number of repeat times for resampling. | |
| PARAM_SEARCH_STRATEGY | NVARCHAR | No default value | Specifies the parameter search method:<br>• grid<br>• random | When *_sha is specified in RESAMPLING_METHOD, PARAM_SEARCH_STRATEGY is mandatory. When *_hyperband is specified in RESAMPLING_METHOD, PARAM_SEARCH_STRATEGY is set to random whatever other value is provided. Otherwise if this parameter is not set, only model evaluation is activated. |
| RANDOM_SEARCH_TIMES | INTEGER | No default value | Specifies the number of times to randomly select candidate parameters for selection. | Mandatory and valid when PARAM_SEARCH_STRATEGY is set to random. |
| REDUCTION_RATE | DOUBLE | 3.0 | Specifies reduction rate in SHA or Hyperband method. | Only valid when RESAMPLING_METHOD is specified as *_sha or *_hyperband. |
| RESOURCE | NVARCHAR | No default value | Specifies resource definition for SHA or Hyperband method. It could be some parameter name depending on specific algorithm. If nothing is specified, size of data set is used as resource by default. | Only valid when RESAMPLING_METHOD is specified as *_sha or *_hyperband. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| MIN_RESOURCE_RATE | DOUBLE | Algorithm specified | Specifies the minimum resource rate that should be used in SHA or Hyperband iteration. | Only valid when RESAMPLING_METHOD is specified as *_sha or *_hyperband. |
| MAX_RESOURCE | INTEGER or DOUBLE | No default value | Specifies the maximum resource that should be used in SHA or Hyperband iteration. Data type of this parameter depends on specified parameter RESOURCE. | Only valid when RESAMPLING_METHOD is specified as *_sha or *_hyperband. It is valid and mandatory if RESOURCE is set. |
| AGGRESSIVE_ELIMINATION | INTEGER | 0 | Specifies whether to apply aggressive elimination while using SHA method. Aggressive elimination happens when the data size and parameters size to be searched does not match and there are still bunch of parameters to be searched while data size reaches its upper limits. If aggressive elimination is applied, lower bound of limit of data size will be used multiple times first to reduce number of parameters.<br>• 0: does not apply aggressive elimination<br>• 1: applies aggressive elimination | Only valid when RESAMPLING_METHOD is specified as *_sha. |
| SEED | INTEGER | 0 | Specifies the seed for random generation. Use system time when 0 is specified. | |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| TIMEOUT | INTEGER | 0 | Specifies maximum running time for model evaluation or parameter selection, in seconds. No timeout when 0 is specified. | |
| PROGRESS_INDICATOR_ID | NVARCHAR | No default value | Sets an ID of progress indicator for model evaluation or parameter selection. No progress indicator is active if no value is provided. | |
| THREAD_RATIO | DOUBLE | 0 | Specifies the ratio of total number of threads that can be used by this function. The range of this parameter is from 0 to 1, where 0 means only using one thread, and 1 means using at most all currently available threads. Values outside this range are ignored and this function heuristically determines the number of threads to use (share the same parameter with primary algorithm). | |

### Progress

To check progress, run the following SQL statement:

```
SELECT * FROM _SYS_AFL.FUNCTION_PROGRESS_IN_AFLPAL WHERE EXECUTION_ID =
<specified id>
```

The result is similar to this:

| | EXECUTION_ID | FUNCTION_NAME | HOST | PORT | CONN... | PROGRESS_TIMESTAMP | PROGRESS_ELAPSEDTIME | PROGRESS_CURRENT | PROGRESS_MAX | PROGRESS_LEVEL | PROGRESS_MESSAGE |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | TEST | PAL_SVM | pvgl... | 30,... | 203,048 | Mar 2, 2018 4:57:56.999 PM | 6,113,063 | 0 | 20 | n/a | COEF_CONST=0.00... |
| 2 | TEST | PAL_SVM | pvgl... | 30,... | 203,048 | Mar 2, 2018 4:57:56.999 PM | 6,303,104 | 1 | 20 | n/a | COEF_CONST=0.00... |
| 3 | TEST | PAL_SVM | pvgl... | 30,... | 203,048 | Mar 2, 2018 4:57:56.999 PM | 6,547,177 | 3 | 20 | n/a | COEF_CONST=0.00... |
| 4 | TEST | PAL_SVM | pvgl... | 30,... | 203,048 | Mar 2, 2018 4:57:56.999 PM | 6,546,908 | 2 | 20 | n/a | COEF_CONST=0.00... |
| 5 | TEST | PAL_SVM | pvgl... | 30,... | 203,048 | Mar 2, 2018 4:57:57.999 PM | 6,670,226 | 4 | 20 | n/a | COEF_CONST=0.00... |
| 6 | TEST | PAL_SVM | pvgl... | 30,... | 203,048 | Mar 2, 2018 4:57:57.999 PM | 6,670,915 | 5 | 20 | n/a | COEF_CONST=0.00... |
| 7 | TEST | PAL_SVM | pvgl... | 30,... | 203,048 | Mar 2, 2018 4:57:57.999 PM | 6,696,194 | 6 | 20 | n/a | COEF_CONST=0.00... |
| 8 | TEST | PAL_SVM | pvgl... | 30,... | 203,048 | Mar 2, 2018 4:57:57.999 PM | 6,718,541 | 7 | 20 | n/a | COEF_CONST=0.00... |
| 9 | TEST | PAL_SVM | pvgl... | 30,... | 203,048 | Mar 2, 2018 4:57:57.999 PM | 6,767,644 | 8 | 20 | n/a | COEF_CONST=0.00... |
| 10 | TEST | PAL_SVM | pvgl... | 30,... | 203,048 | Mar 2, 2018 4:57:57.999 PM | 7,444,722 | 9 | 20 | n/a | COEF_CONST=0.00... |

Column description:

| Column Name | Description |
|---|---|
| EXECUTION_ID | ID specified by the PROGRESS_INDICATOR_ID parameter. |
| FUNCTION_NAME | Function used to perform model evaluation or parameter selection. |
| PROGRESS_TIMESTAMP | Job finish timestamp. |
| PROGRESS_ELAPSEDTIME | Time elapsed since function starts, in microseconds. |
| PROGRESS_CURRENT | Job number. |
| PROGRESS_MAX | Total job number. |
| PROGRESS_MESSAGE | Parameters used to perform parameter selection for job. May only contain part information due to length limits. |

### Specify Candidate Parameters

When parameter selection is enabled, you must specify candidate parameters for parameter selection to work. There are three kinds of candidate parameters:

- Fixed parameter
  This kind of parameters is specified like ordinary parameter using its original name. Such parameters do not engage in parameter selection.
- Parameter value set
  This kind of parameters makes use of original parameter name with a suffix _VALUES. For instance, if the original parameter name is XXX, then XXX_VALUES is used.
  Value set takes form of {value1, value2, value3} as a string. Values should be surrounded by double quotes.
  Examples:
  {1, 2, 3}
  {0.1}
  {“a”, “b”}
  {“1,2,3”, “3,2,1”} – two values of string type
- Parameter range
  Like the parameter value set, this kind of parameters makes use of the original parameter name with a suffix _RANGE. For instance, if the original parameter name is XXX, then XXX_RANGE is used.
  Range of parameter takes form of [range start, step, range end] as a string. Only parameter of INTEGER or DOUBLE type is supported. When using grid search, algorithm takes range start as the first value, and adds

one step to it as the second value and so on until it reaches the range end. When using random search, step can be omitted for it is not used.
Example:
[0, 0.1, 1]
[0, 1, 10]
[0.1, , 0.5] – only for random parameter search

Only one of these three kinds of candidate parameter can exist for each parameter in parameter selection. For the searchable parameter list of each algorithm, see the corresponding algorithm section.

### Output Tables

| Table | Column | Data Type | Column Name | Description |
|---|---|---|---|---|
| STATISTICS | 1st column | VARCHAR or NVARCHAR | STAT_NAME | Statistic names. |
| | 2nd column | VARCHAR or NVARCHAR | STAT_VALUE | Statistic values. |
| OPTIMAL_PARAM | 1st column | NVARCHAR | PARAM_NAME | Optimal parameters selected. |
| | 2nd column | INTEGER | INT VALUE | |
| | 3rd column | DOUBLE | DOUBLE_VALUE | |
| | 4th column | NVARCHAR | STRING VALUE | |

Possible statistics

| Name | Description |
|---|---|
| timeout | Indicates whether the program runs out of time. |
| TEST_[n]_[evaluation metric] | The $n^{th}$ round of evaluation result(s) of specified metric of parameters evaluated or selected. For bootstrap, there is one result in each row. For cross validation there are k results in each row, using comma(,) as delimiter (k equals to fold number). |
| TEST_[evaluation metric].MEAN | Mean value of all evaluation results of specified metric of parameters evaluated or selected. |
| TEST_[evaluation metric].VAR | Variance value of all evaluation results of specified metric of parameters evaluated or selected. |

| Name | Description |
| --- | --- |
| EVAL_RESULTS_[line number] | JSON formatted evaluation results of full details with similar name notation.<br><br>Example:<br><br>`{ "candidates": [{ "TEST_ACCURACY": [[1.0], [1.0]], "TEST_ACCURACY.MEAN": 1.0, "TEST_ACCURACY.VAR": 0.0, "parameters": "COEF_CONST=0.00599155;COEF_LIN=0.00500022;SVM_C=10;" }, { "TEST_ACCURACY": [[1.0], [1.0]], "TEST_ACCURACY.MEAN": 1.0, "TEST_ACCURACY.VAR": 0.0, "parameters": "COEF_CONST=0.0041903;COEF_LIN=0.00669262;SVM_C=10;" }, ... ], "version": "1.0" }` |

### Example

Assume that:

- DM_PAL is a schema belonging to USER1; and
- USER1 has been assigned the AFL__SYS_AFL_AFLPAL_EXECUTE or AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION role.

#### Example 1: Model evaluation

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_SVM_DATA_TBL;
CREATE COLUMN TABLE PAL_SVM_DATA_TBL (
    "ID" INTEGER,
    "ATTRIBUTE1" DOUBLE,
    "ATTRIBUTE2" DOUBLE,
    "ATTRIBUTE3" DOUBLE,
    "ATTRIBUTE4" VARCHAR(10),
    "LABEL" INTEGER
);
INSERT INTO PAL_SVM_DATA_TBL VALUES (0, 1, 10, 100, 'A', 1);
INSERT INTO PAL_SVM_DATA_TBL VALUES (1, 1.1, 10.1, 100,'A', 1);
INSERT INTO PAL_SVM_DATA_TBL VALUES (2, 1.2, 10.2, 100, 'A', 1);
INSERT INTO PAL_SVM_DATA_TBL VALUES (3, 1.3, 10.4, 100, 'A', 1);
INSERT INTO PAL_SVM_DATA_TBL VALUES (4, 1.2, 10.3, 100, 'AB', 1);
INSERT INTO PAL_SVM_DATA_TBL VALUES (5, 4, 40, 400, 'AB', 2);
INSERT INTO PAL_SVM_DATA_TBL VALUES (6, 4.1, 40.1, 400, 'AB', 2);
INSERT INTO PAL_SVM_DATA_TBL VALUES (7, 4.2, 40.2, 400, 'AB', 2);
INSERT INTO PAL_SVM_DATA_TBL VALUES (8, 4.3, 40.4, 400, 'AB', 2);
INSERT INTO PAL_SVM_DATA_TBL VALUES (9, 4.2, 40.3, 400, 'AB', 2);
INSERT INTO PAL_SVM_DATA_TBL VALUES (10, 9, 90, 900, 'B', 3);
INSERT INTO PAL_SVM_DATA_TBL VALUES (11, 9.1, 90.1, 900, 'A', 3);
INSERT INTO PAL_SVM_DATA_TBL VALUES (12, 9.2, 90.2, 900, 'B', 3);
INSERT INTO PAL_SVM_DATA_TBL VALUES (13, 9.3, 90.4, 900, 'A', 3);
```

```
INSERT INTO PAL_SVM_DATA_TBL VALUES (14, 9.2, 90.3, 900, 'A', 3);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR (1000)
);
-- specify SVM parameter
INSERT INTO #PAL_PARAMETER_TBL VALUES ('HAS_ID', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 1, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('TYPE', 1, NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('KERNEL_TYPE', 3, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SVM_C', NULL, 101, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('COEF_LIN', NULL, 0.0065, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('COEF_CONST', NULL, 0.0055, NULL);
-- specify parameter selection settings
INSERT INTO #PAL_PARAMETER_TBL VALUES ('RESAMPLING_METHOD', NULL, NULL, 'cv');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('EVALUATION_METRIC', NULL, NULL,
'F1_SCORE');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FOLD_NUM', 3, NULL, NULL); -- mandatory
for cross validation
INSERT INTO #PAL_PARAMETER_TBL VALUES ('REPEAT_TIMES', 2, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SEED', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PROGRESS_INDICATOR_ID', NULL, NULL,
'TEST');
-- call procedure
CALL _SYS_AFL.PAL_SVM(PAL_SVM_DATA_TBL, "#PAL_PARAMETER_TBL", ?, ?, ?);
```

**Expected Results**

STATISTICS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | timeout | FALSE |
| 2 | TEST_1_F1_SCORE | 1, 1, 0.555555 |
| 3 | TEST_2_F1_SCORE | 1, 0.822222, 1 |
| 4 | TEST_F1_SCORE.... | 0.896296 |
| 5 | TEST_F1_SCORE.... | 0.0329218 |
| 6 | EVAL_RESULTS_1 | {"candidates":[{"TEST_F1_SCORE":[[1.0,1.0,... |

**Example 2: Gird parameter selection**

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_SVM_DATA_TBL;
CREATE COLUMN TABLE PAL_SVM_DATA_TBL (
    "ID" INTEGER,
    "ATTRIBUTE1" DOUBLE,
    "ATTRIBUTE2" DOUBLE,
    "ATTRIBUTE3" DOUBLE,
    "ATTRIBUTE4" VARCHAR(10),
    "LABEL" INTEGER
);
INSERT INTO PAL_SVM_DATA_TBL VALUES (0, 1, 10, 100, 'A', 1);
INSERT INTO PAL_SVM_DATA_TBL VALUES (1, 1.1, 10.1, 100,'A', 1);
INSERT INTO PAL_SVM_DATA_TBL VALUES (2, 1.2, 10.2, 100, 'A', 1);
```

```
INSERT INTO PAL_SVM_DATA_TBL VALUES (3, 1.3, 10.4, 100, 'A', 1);
INSERT INTO PAL_SVM_DATA_TBL VALUES (4, 1.2, 10.3, 100, 'AB', 1);
INSERT INTO PAL_SVM_DATA_TBL VALUES (5, 4, 40, 400, 'AB', 2);
INSERT INTO PAL_SVM_DATA_TBL VALUES (6, 4.1, 40.1, 400, 'AB', 2);
INSERT INTO PAL_SVM_DATA_TBL VALUES (7, 4.2, 40.2, 400, 'AB', 2);
INSERT INTO PAL_SVM_DATA_TBL VALUES (8, 4.3, 40.4, 400, 'AB', 2);
INSERT INTO PAL_SVM_DATA_TBL VALUES (9, 4.2, 40.3, 400, 'AB', 2);
INSERT INTO PAL_SVM_DATA_TBL VALUES (10, 9, 90, 900, 'B', 3);
INSERT INTO PAL_SVM_DATA_TBL VALUES (11, 9.1, 90.1, 900, 'A', 3);
INSERT INTO PAL_SVM_DATA_TBL VALUES (12, 9.2, 90.2, 900, 'B', 3);
INSERT INTO PAL_SVM_DATA_TBL VALUES (13, 9.3, 90.4, 900, 'A', 3);
INSERT INTO PAL_SVM_DATA_TBL VALUES (14, 9.2, 90.3, 900, 'A', 3);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR (1000)
);
-- specify SVM parameter
INSERT INTO #PAL_PARAMETER_TBL VALUES ('HAS_ID', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 1, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('TYPE', 1, NULL, NULL); -- fixed
INSERT INTO #PAL_PARAMETER_TBL VALUES ('KERNEL_TYPE', 3, NULL, NULL); -- fixed
-- specify parameter selection settings
INSERT INTO #PAL_PARAMETER_TBL VALUES ('RESAMPLING_METHOD', NULL, NULL,
'bootstrap');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('EVALUATION_METRIC', NULL, NULL,
'ACCURACY');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('REPEAT_TIMES', 2, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PARAM_SEARCH_STRATEGY', NULL, NULL,
'grid');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SEED', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PROGRESS_INDICATOR_ID', NULL, NULL,
'TEST');
-- specify SVM parameter candidates
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SVM_C_VALUES', NULL, NULL, '{1, 10, 21,
100}');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('COEF_LIN_RANGE', NULL, NULL, '[0.005,
0.001, 0.007]'); -- equivalent to {0.005, 0.006, 0.007}
INSERT INTO #PAL_PARAMETER_TBL VALUES ('COEF_CONST_RANGE', NULL, NULL, '[0.003,
0.002, 0.006]'); -- equivalent to {0.003, 0.005}
-- call procedure
CALL _SYS_AFL.PAL_SVM(PAL_SVM_DATA_TBL, "#PAL_PARAMETER_TBL", ?, ?, ?);
```

### Expected Results

STATISTICS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | timeout | FALSE |
| 2 | TEST_1_ACCURACY | 1 |
| 3 | TEST_2_ACCURACY | 0.75 |
| 4 | TEST_ACCURACY.MEAN | 0.875 |
| 5 | TEST_ACCURACY.VAR | 0.03125 |
| 6 | EVAL_RESULTS_1 | {"candidates":[{"TEST_ACCU... |
| 7 | EVAL_RESULTS_2 | "COEF_CONST=0.003;COEF... |
| 8 | EVAL_RESULTS_3 | ACCURACY.VAR":0.031250,... |
| 9 | EVAL_RESULTS_4 | 0]],"TEST_ACCURACY.MEAN... |

OPTIMAL PARAMETERS:

| | PARAM_NAME | INT_VALUE | DOUBLE_VALUE | STRING_VALUE |
|---|---|---|---|---|
| 1 | COEF_CONST | ? | 0.003 | ? |
| 2 | COEF_LIN | ? | 0.005 | ? |
| 3 | SVM_C | ? | 10 | ? |

**Example 3: Random parameter selection**

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_SVM_DATA_TBL;
CREATE COLUMN TABLE PAL_SVM_DATA_TBL (
    "ID" INTEGER,
    "ATTRIBUTE1" DOUBLE,
    "ATTRIBUTE2" DOUBLE,
    "ATTRIBUTE3" DOUBLE,
    "ATTRIBUTE4" VARCHAR(10),
    "LABEL" INTEGER
);
INSERT INTO PAL_SVM_DATA_TBL VALUES (0, 1, 10, 100, 'A', 1);
INSERT INTO PAL_SVM_DATA_TBL VALUES (1, 1.1, 10.1, 100,'A', 1);
INSERT INTO PAL_SVM_DATA_TBL VALUES (2, 1.2, 10.2, 100, 'A', 1);
INSERT INTO PAL_SVM_DATA_TBL VALUES (3, 1.3, 10.4, 100, 'A', 1);
INSERT INTO PAL_SVM_DATA_TBL VALUES (4, 1.2, 10.3, 100, 'AB', 1);
INSERT INTO PAL_SVM_DATA_TBL VALUES (5, 4, 40, 400, 'AB', 2);
INSERT INTO PAL_SVM_DATA_TBL VALUES (6, 4.1, 40.1, 400, 'AB', 2);
INSERT INTO PAL_SVM_DATA_TBL VALUES (7, 4.2, 40.2, 400, 'AB', 2);
INSERT INTO PAL_SVM_DATA_TBL VALUES (8, 4.3, 40.4, 400, 'AB', 2);
INSERT INTO PAL_SVM_DATA_TBL VALUES (9, 4.2, 40.3, 400, 'AB', 2);
INSERT INTO PAL_SVM_DATA_TBL VALUES (10, 9, 90, 900, 'B', 3);
INSERT INTO PAL_SVM_DATA_TBL VALUES (11, 9.1, 90.1, 900, 'A', 3);
INSERT INTO PAL_SVM_DATA_TBL VALUES (12, 9.2, 90.2, 900, 'B', 3);
INSERT INTO PAL_SVM_DATA_TBL VALUES (13, 9.3, 90.4, 900, 'A', 3);
INSERT INTO PAL_SVM_DATA_TBL VALUES (14, 9.2, 90.3, 900, 'A', 3);
DROP TABLE #PAL_PARAMETER_TBL;
```

```
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR (1000)
);
-- specify SVM parameter
INSERT INTO #PAL_PARAMETER_TBL VALUES ('HAS_ID', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('THREAD_RATIO', NULL, 1, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('TYPE', 1, NULL, NULL); -- fixed
INSERT INTO #PAL_PARAMETER_TBL VALUES ('KERNEL_TYPE', 3, NULL, NULL); -- fixed
-- specify parameter selection settings
INSERT INTO #PAL_PARAMETER_TBL VALUES ('RESAMPLING_METHOD', NULL, NULL,
'stratified_bootstrap'); -- stratified version
INSERT INTO #PAL_PARAMETER_TBL VALUES ('EVALUATION_METRIC', NULL, NULL,
'ACCURACY');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('REPEAT_TIMES', 2, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PARAM_SEARCH_STRATEGY', NULL, NULL,
'random');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('RANDOM_SEARCH_TIMES', 10, NULL, NULL);
-- it is mandatory when using grid search
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SEED', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PROGRESS_INDICATOR_ID', NULL, NULL,
'TEST');
-- specify SVM parameter candidates
INSERT INTO #PAL_PARAMETER_TBL VALUES ('SVM_C_VALUES', NULL, NULL, '{1, 10, 21,
100}');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('COEF_LIN_RANGE', NULL, NULL, '[0.005, ,
0.007]'); -- omit step
INSERT INTO #PAL_PARAMETER_TBL VALUES ('COEF_CONST_RANGE', NULL, NULL, '[0.003,
0.002, 0.006]'); -- step is ignored
-- call procedure
CALL _SYS_AFL.PAL_SVM(PAL_SVM_DATA_TBL, "#PAL_PARAMETER_TBL", ?, ?, ?);
```

**Expected Results**

STATISTICS:

| | STAT_NAME | STAT_VALUE |
|---|---|---|
| 1 | timeout | FALSE |
| 2 | TEST_1_ACCURACY | 1 |
| 3 | TEST_2_ACCURACY | 1 |
| 4 | TEST_ACCURACY.MEAN | 1 |
| 5 | TEST_ACCURACY.VAR | 0 |
| 6 | EVAL_RESULTS_1 | {"candidates":[{"TEST_ACCU... |
| 7 | EVAL_RESULTS_2 | _CONST=0.00440771;COEF... |

OPTIMAL PARAMETERS:

| | PARAM_NAME | INT_VALUE | DOUBLE_VALUE | STRING_VALUE |
|---|---|---|---|---|
| 1 | COEF_CONST | ? | 0.005991554424690797 | ? |
| 2 | COEF_LIN | ? | 0.005000228762595243 | ? |
| 3 | SVM_C | ? | 10 | ? |

## 6.1 Successive Halving and Hyperband for Model Evaluation and Parameter Selection

The purpose of successive halving is to accelerate parameter selection. It belongs to a general idea called early stopping.

An observation in parameter selection is that most of the candidates in search space only have poor performance, yet they all cost the same resource. It leads to an intuitive optimization that to filter out those unpromising candidates at a relatively low cost and focuses on the promising ones.

Successive halving is based on such idea. It begins with a low resource configuration then reduce the candidates by a certain rate. It then increases the resource configuration and repeat last step. This process continues until there is only one candidate left.

The definition of resource in successive halving can be a wide range. The default resource is the size of data set. Besides, each algorithm can define their own specific resource that usually is some regular parameter of it, like `ITER_NUM` of HGBT.

Some parameters can be sensitive to successive halving because the nature of it. One classic example is parameter `learning rate`, by which the speed of convergence is influenced directly. So, if the resource is `number of iteration`, it is not very fair for small `learning rate`.

Because of the nature of successive halving, it will not always pick the best candidate in sense of the evaluation metric, but hopefully a relatively good one.

It is quite hard for successive halving to determine how many resources is needed for a specific search task. On other hand, hyperband can ease this problem by some level. Basically hyperband will try several configurations regarding resource and search space, then calls successive halving to detect the best one. It only supports random search.

### Relevant Parameters

- RESAMPLING_METHOD
  To activate successive halving, choose options like *_sha.
  To activate hyperband choose options like *_hyperband.

- REDUCTION_RATE
  The rate of elimination of candidates in each iteration, also the rate of increasement of resource
- RESOURCE
  The options can vary on different algorithms.
  If not set, size of data set is used.
- MIN_RESOURCE_RATE
  Minimum resource by rate that should be used in iteration. Usually it indicates the resource used in the first iteration.
- MAX_RESOURCE
  Maximum resource that should be used in iteration. Usually it indicates the resource used in the last iteration. Only valid when RESOURCE is set.
- AGGRESSIVE_ELIMINATION
  It is used to balance the reduction of candidates and increasing of resource when their quantities do not match. It happens, when there are still bunch of candidates to be searched while resource reaches its upper limits. If aggressive elimination is applied, lower bound of limit of resource will be used multiple times first to reduce number of candidates to a certain level.

## Example

```
set schema DM_PAL;
drop table PAL_HGBT_DATA_TAB;
create column table PAL_HGBT_DATA_TAB (
    "ATT1" double,
    "ATT2" double,
    "ATT3" double,
    "ATT4" double,
    "LABEL" nvarchar(50)
);
insert into PAL_HGBT_DATA_TAB values (1.0, 10.0, 100, 1.0, 'A');
insert into PAL_HGBT_DATA_TAB values (1.1, 10.1, 100, 1.0, 'A');
insert into PAL_HGBT_DATA_TAB values (1.2, 10.2, 100, 1.0, 'A');
insert into PAL_HGBT_DATA_TAB values (1.3, 10.4, 100, 1.0, 'A');
insert into PAL_HGBT_DATA_TAB values (1.2, 10.3, 100, 1.0, 'A');
insert into PAL_HGBT_DATA_TAB values (4.0, 40.0, 400, 4.0, 'B');
insert into PAL_HGBT_DATA_TAB values (4.1, 40.1, 400, 4.0, 'B');
insert into PAL_HGBT_DATA_TAB values (4.2, 40.2, 400, 4.0, 'B');
insert into PAL_HGBT_DATA_TAB values (4.3, 40.4, 400, 4.0, 'B');
insert into PAL_HGBT_DATA_TAB values (4.2, 40.3, 400, 4.0, 'A');
insert into PAL_HGBT_DATA_TAB values (9.0, 90.0, 900, 2.0, 'A');
insert into PAL_HGBT_DATA_TAB values (9.1, 90.1, 900, 1.0, 'B');
insert into PAL_HGBT_DATA_TAB values (9.2, 90.2, 900, 2.0, 'B');
insert into PAL_HGBT_DATA_TAB values (9.3, 90.4, 900, 1.0, 'B');
insert into PAL_HGBT_DATA_TAB values (9.2, 90.3, 900, 1.0, 'B');
insert into PAL_HGBT_DATA_TAB values (1.0, 10.0, 100, 1.0, 'A');
insert into PAL_HGBT_DATA_TAB values (1.1, 10.1, 100, 1.0, 'A');
insert into PAL_HGBT_DATA_TAB values (1.2, 10.2, 100, 1.0, 'A');
insert into PAL_HGBT_DATA_TAB values (1.3, 10.4, 100, 1.0, 'A');
insert into PAL_HGBT_DATA_TAB values (1.2, 10.3, 100, 1.0, 'A');
drop table PAL_PARAMETER_TAB;
create column table PAL_PARAMETER_TAB (
    "PARAM_NAME" nvarchar(100),
    "INT_VALUE" integer,
    "DOUBLE_VALUE" double,
    "STRING_VALUE" nvarchar(100)
);
insert into PAL_PARAMETER_TAB values ('RESAMPLING_METHOD', null, null,
'bootstrap_sha');
```

```
insert into PAL_PARAMETER_TAB values ('RESOURCE', null, null, 'ITER_NUM');
insert into PAL_PARAMETER_TAB values ('MAX_RESOURCE', 10, null, null);
insert into PAL_PARAMETER_TAB values ('MIN_RESOURCE_RATE', null, 0.0, null);
insert into PAL_PARAMETER_TAB values ('AGGRESSIVE_ELIMINATION', 1, null, null);
insert into PAL_PARAMETER_TAB values ('PARAM_SEARCH_STRATEGY', null, null,
'grid');
insert into PAL_PARAMETER_TAB values ('EVALUATION_METRIC', null, null,
'ERROR_RATE');
insert into PAL_PARAMETER_TAB values ('SEED', 1, null, null);
insert into PAL_PARAMETER_TAB values ('MAX_DEPTH_VALUES', null, null, '{3, 5,
6}');
insert into PAL_PARAMETER_TAB values ('ETA_RANGE', null, null, '[0.1, 0.1,
1.0]');
insert into PAL_PARAMETER_TAB values ('GAMMA_RANGE', null, null, '[0.1, 0.1,
1.0]');
insert into PAL_PARAMETER_TAB values ('ALPHA_RANGE', null, null, '[0.1, 0.1,
1.0]');
do begin
    LT_DATA_TAB = select * from PAL_HGBT_DATA_TAB;
    LT_PARAM_TAB = select * from PAL_PARAMETER_TAB;
    call _SYS_AFL.PAL_HGBT(:LT_DATA_TAB, :LT_PARAM_TAB, LT_MODEL_TAB,
LT_IMP_TAB, LT_CONFUSE_TAB, LT_STAT_TAB, LT_CV_TAB);
    select * from :LT_MODEL_TAB;
    select * from :LT_IMP_TAB;
    select * from :LT_CONFUSE_TAB;
    select * from :LT_STAT_TAB;
    select * from :LT_CV_TAB;
    insert into PAL_HGBT_MODEL_TAB select * from :LT_MODEL_TAB;
end;
```

# 7 Model Comparison

Model comparison functions are integrations of partition, training, testing and measuring. They help user to compare different models with the same dataset easier.

Model comparison functions are composed of several steps:

1. (Optional) Partition data into training set and testing set.
2. Call training functions with training dataset.
3. Call predict functions with testing dataset.
4. Calculate measurements on testing data

Model Comparison accepts three formats of input data:

- **One unpartitioned data table**
  Raw data table without any partitioning. Model comparison function will do partition on the dataset and a partition table will be returned.
- **One unpartitioned data table with one partition table returned from PAL_PARTITION function (_WITH_PARTITION_TABLE)**
  If PAL_PARTITION function is already called with the input dataset, and the user has the partition table returned from PAL_PARTITION function, model comparison function will partition the dataset based on provided partition table.
- **Two data tables, one for training and one for testing (_WITH_TESTING_TABLE)**
  The data is already partitioned into two tables (training table and testing table). The training table and testing table should have the same shape (same number of columns and the columns should be ordered in the same way). Model comparison functions will use these two tables for training and testing.
  Model Comparison accepts two configuration tables: one is called parameter table and the other is called configuration table. Mandatory parameters must be provided through the parameter table. Configuration table is used to let the program know which models to run and which parameters to use for the specific model.

**Functions enabled in model comparison:**

We only provide model comparison for regression functions now.

| Function | Alias in Configuration Table |
|---|---|
| Multiple Linear Regression | MLR |
| Support Vector Machine(Regression) | SVM |
| Bi-Variate Geometric Regression | GEOR |
| Bi-Variate Natural Logarithmic Regression | LOGARITHMIC |
| Exponential Regression | EXP |
| Polynomial Regression | POLYNOMIAL |

**Model Comparison function names:**

Regression:

PAL_MODEL_COMPARISON_REGRESSION

PAL_MODEL_COMPARISON_REGRESSION_WITH_PARTITION_TABLE

PAL_MODEL_COMPARISON_REGRESSION_WITH_TESTING_TABLE

**Overview of All Tables**

- **One unpartitioned data table**

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | DATA table | IN |
| 2 | Config table | IN |
| 3 | Parameter table | IN |
| 4 | Model table | OUT |
| 5 | Fitting table | OUT |
| 6 | Training State table | OUT |
| 7 | Evaluation table | OUT |
| 8 | Partition table | OUT |
| 9 | Message table | OUT |

- **With partition table returned from PAL_PARTITION**

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | DATA table | IN |
| 2 | Config table | IN |
| 3 | Parameter table | IN |
| 4 | Partition table | IN |
| 5 | Model table | OUT |
| 6 | Fitting table | OUT |
| 7 | Training State table | OUT |
| 8 | Evaluation table | OUT |

| Position | Table Name | Parameter Type |
|---|---|---|
| 9 | Message table | OUT |

- Training and Testing Tables

| Position | Table Name | Parameter Type |
|---|---|---|
| 1 | Training DATA table | IN |
| 2 | Testing Data table | IN |
| 3 | Configuration table | IN |
| 4 | Parameter table | IN |
| 5 | Model table | OUT |
| 6 | Fitting table | OUT |
| 7 | Training State table | OUT |
| 8 | Evaluation table | OUT |
| 9 | Message table | OUT |

### Data Table

- **One unpartitioned data table**

| Table | Column | Data Type | Description |
|---|---|---|---|
| Data table | All columns | INTEGER, DOUBLE, DECIMAL(p, s), VARCHAR, or NVARCHAR | As input data table defined in corresponding functions. |

- **With two data tables for training and testing purpose**

| Table | Column | Data Type | Description |
|---|---|---|---|
| Training data table | All columns | INTEGER, DOUBLE, DECIMAL(p, s), VARCHAR, or NVARCHAR | As input data table defined in corresponding functions. |

| Table | Column | Data Type | Description |
|---|---|---|---|
| Testing data table | All columns | INTEGER, DOUBLE, DECIMAL(p, s), VARCHAR, or NVARCHAR | As input data table defined in corresponding functions. Testing data table should have exactly the same number of columns as training data table. |

### Configuration Table

| Column Position | Data Type | Description |
|---|---|---|
| 1 | INTEGER or VARCHAR or NVARCHAR | ID column. Assign a ID for each model and configuration pair. |
| 2 | VARCHAR or NVARCHAR | Model name column, which indicates the model to use. Use "Alias in configuration table" mentioned earlier in this document. |
| 3 | VARCHAR or NVARCHAR | Parameter settings for this specific model in Json format. For example, to set<br>• ENET_ALPHA parameter to 0.5 and<br>• ALG parameter to 4<br>for MLR model, the configuration string should be:<br>`{"ALG": 4, "ENET_ALPHA":0.5}` |

### Parameter Table

#### Mandatory Parameters

The following parameters are mandatory and must be given a value.

Mandatory Parameters for Model Selection Regression functions:

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| HAS_ID | INTEGER | 2 | Specifies whether the ID column exists.<br>• 0: Has no ID column<br>• 1: Has an ID column<br>All other values are invalid. |

| Name | Data Type | Default Value | Description |
|---|---|---|---|
| DEPENDENT_VARIABLE | VARCHAR or NVARCHAR | No default value | Column name of dependent variable. See PAL document for regression functions for details. |

### Optional Parameters

The following parameter is optional. If it is not specified, PAL will use its default value.

Optional parameters for all model comparison functions:

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| TRAINING_PERCENT | Double | 0.8 | Defines the percentage of the data to be used as training dataset. | The total of TRAINING_PERCENT and TESTING_PERCENT should be 1.<br>This parameter is only valid for procedure PAL_MODEL_COMPARISON_REGRESSION. |
| TESTING_PERCENT | Double | 0.2 | Defines the percentage the data to be used as testing dataset. | The total of TRAINING_PERCENT and TESTING_PERCENT should be 1.<br>Append one more paragraph below: This parameter is only valid for procedure PAL_MODEL_COMPARISON_REGRESSION. |
| RANDOM_SEED | INTEGER | 0 | Indicates the seed used to initialize the random number generator.<br>• 0: Uses the system time<br>• Positive value: Uses the specified value as seed | This parameter is only valid for procedure PAL_MODEL_COMPARISON_REGRESSION. |

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| PARTITION_METHOD | INTEGER | 0 | Indicates the partition method:<br>• 0: Random partition<br>• Not 0: Stratified partition | This parameter is only valid for procedure PAL_MODEL_COMPARISON_REGRESSION. |
| STRATIFIED_COLUMN | VARCHAR or NVARCHAR | None | Indicates which column is used for stratification. The column type must be integer or varchar or nvarchar. | This parameter is only valid for procedure PAL_MODEL_COMPARISON_REGRESSION. |
| THREAD_RATIO | Double | 1.0 | Specifies the ratio of total number of threads that can be used by this function. The value range is from 0 to 1, where 0 means only using 1 thread, and 1 means using at most all the currently available threads. Values outside the range will be ignored and this function heuristically determines the number of threads to use. | |

Optional parameters for model comparison regression functions:

| Name | Data Type | Default Value | Description | Dependency |
|---|---|---|---|---|
| CATEGORICAL_VARIABLE | VARCHAR or NVARCHAR | No default variable | Column name of categorical variable. See PAL document for regression functions for detail. | |

### Output Table

Partition Table

| Table | Column | Column Name | Data Type | Description |
|---|---|---|---|---|
| OUTPUT | 1st column | DATA_ID (in the input table) column name | DATA_ID (in the input table) data type | ID. Key of the data table. |
| | 2nd column | PARTITION_TYPE | INTEGER | Indicates which partition the data belongs to.<br>• 1: Training data<br>• 2: Testing data |

### Other Output Tables

| Table | Column | Column Name | Data Type | Description |
|---|---|---|---|---|
| Model Table | 1st column | MODEL_ID | Data Type of Configuration Table ID column | Model ID refers to the ID column in the configuration table. |
| | 2nd column | ROW_INDEX | NVARCHAR(10) | Row index for the corresponding model content. |
| | 3rd column | MODEL_CONTENT | NVARCHAR(5000) | Model content in PMML or Json. |
| Fitting Table | 1st column | MODEL_ID | Data Type of Configuration Table ID column | Model ID refers to the ID column in configuration table. |
| | 2nd column | DATA_ID | Data Type of Data Table ID column | Data ID refers to the ID column in data table. |
| | 3rd column | ACTUAL_VALUE | INTEGER or DOUBLE or NVARCHAR(100) | Actual value of the data. |
| | 4th column | PREDICTED_VALUE | DOUBLE, NVARCHAR(100) | Predicted value of the data. |
| Training Stat Table | 1st column | MODEL_ID | Data Type of Configuration Table ID column | Model ID refers to the ID column in configuration table. |
| | 2nd column | STAT_NAME | NVARCHAR(256) | Stat name. |
| | 3rd column | INT_VALUE | Integer | The integer value of the statistic. |
| | 4th column | DOUBLE_VALUE | Double | The double value of the statistic. |

| Table | Column | Column Name | Data Type | Description |
|---|---|---|---|---|
| | 5th column | STRING_VALUE | NVARCHAR(1000) | The string value of the statistic. (Almost all values are treated as string value) |
| Testing Stat Table | 1st column | MODEL_ID | Data Type of Configuration Table ID column | Model ID refers to the ID column in configuration table. |
| | 2nd column | STAT_NAME | NVARCHAR(20) | Stat name. |
| | 3rd column | STAT_VALUE | NVARCHAR(256) | Stat value. |
| Error Message Table | 1st column | MODEL_ID | Data Type of Configuration Table ID column | Model ID refers to the ID column in configuration table. |
| | 2nd column | TIME_STAMP | NVARCHAR(20) | Time stamp of the message. |
| | 3rd column | ERROR_CODE | Integer | Error code of the message. |
| | 4th column | MESSAGE_CONTENT | NVARCHAR(5000) | Error message content. |

## Examples

### Example 1: Use Model Comparison Regression with unpartitioned data

Models compared in this example:

- Multiple Linear Regression with cyclical coordinate descent method to solve elastic net regularized multiple linear regression and set ALPHA to 0.5
- Multiple Linear Regression with SVD
- Support Vector Machine with linear kernel

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_MODEL_COMP_DATA_TBL;
CREATE COLUMN TABLE PAL_MODEL_COMP_DATA_TBL
    ("ID" INTEGER, "Y" DOUBLE, "X1" DOUBLE);
INSERT INTO PAL_MODEL_COMP_DATA_TBL VALUES (0, 6.879, 0.02);
INSERT INTO PAL_MODEL_COMP_DATA_TBL VALUES (1, 3.449, 0.50);
INSERT INTO PAL_MODEL_COMP_DATA_TBL VALUES (2,  6.635, 0.54);
INSERT INTO PAL_MODEL_COMP_DATA_TBL VALUES (3, 11.844, 1.04);
INSERT INTO PAL_MODEL_COMP_DATA_TBL VALUES (4,  2.786, 1.50);
INSERT INTO PAL_MODEL_COMP_DATA_TBL VALUES (5,  2.389, 0.04);
INSERT INTO PAL_MODEL_COMP_DATA_TBL VALUES (6, 0.011, 2.00);
INSERT INTO PAL_MODEL_COMP_DATA_TBL VALUES (7,  8.839, 2.04);
INSERT INTO PAL_MODEL_COMP_DATA_TBL VALUES (8,  4.689, 1.54);
INSERT INTO PAL_MODEL_COMP_DATA_TBL VALUES (9, 5.507, 1.00);
DROP TABLE PAL_MC_CONFIG_TBL;
CREATE COLUMN TABLE PAL_MC_CONFIG_TBL
("MODEL_ID" INT,
```

```
"MODEL_NAME" VARCHAR(20),
"MODEL_CONFIG" VARCHAR(100)
);
INSERT INTO PAL_MC_CONFIG_TBL VALUES (1, 'MLR', '{"ALG": 4, "ENET_ALPHA":0.5}');
--Cyclical coordinate descent method to solve elastic net regularized multiple
linear regression
INSERT INTO PAL_MC_CONFIG_TBL VALUES (2, 'MLR', '{"ALG": 2}'); --SVD (solving
the least square problem)
INSERT INTO PAL_MC_CONFIG_TBL VALUES (3, 'SVM', '{"KERNEL_TYPE": 0}');
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('HAS_ID',1,NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('DEPENDENT_VARIABLE',NULL,NULL,'Y');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('TRAINING_PERCENT',NULL,0.6,NULL);
CALL _SYS_AFL.PAL_MODEL_COMPARISON_REGRESSION(PAL_MODEL_COMP_DATA_TBL,
PAL_MC_CONFIG_TBL, #PAL_PARAMETER_TBL, ?, ?, ?, ?, ?, ?);
```

**Expected Results:**

Model Table:

| | MODEL_ID | ROW_INDEX | MODEL_CONTENT |
|---|---|---|---|
| 1 | 1 | 1 | <PMML version="4.0" xmlns="http://www.dmg.org/PMML-4_0" xmlns:xsi="http://www.... |
| 2 | 2 | 1 | <PMML version="4.0" xmlns="http://www.dmg.org/PMML-4_0" xmlns:xsi="http://www.... |
| 3 | 3 | 0 | {"C":100.0,"CONTENT":"{\"impute_model\":{\"column_statistics\":[{\"COLUMN_NAME\":\... |

Fitting Table:

| | MODEL_ID | DATA_ID | ACTUAL_VALUE | PREDICTED_VALUE |
|---|---|---|---|---|
| 1 | 1 | 1 | 3.449 | 5.863089896579159 |
| 2 | 1 | 6 | 0.011 | 5.901894022047961 |
| 3 | 1 | 7 | 8.839 | 5.90292879872713 |
| 4 | 1 | 9 | 5.507 | 5.87602460506876 |
| 5 | 2 | 1 | 3.449 | 5.863089896579155 |
| 6 | 2 | 6 | 0.011 | 5.901894022047962 |
| 7 | 2 | 7 | 8.839 | 5.90292879872713 |
| 8 | 2 | 9 | 5.507 | 5.876024605068757 |
| 9 | 3 | 1 | 3.449 | 6.0594615107365... |
| 10 | 3 | 6 | 0.011 | 4.5837748460759... |
| 11 | 3 | 7 | 8.839 | 4.54442320168499 |
| 12 | 3 | 9 | 5.507 | 5.567565955849714 |

Training Stat Table:

| | MODEL_ID | STAT_NAME | INT_VALUE | DOUBLE_VALUE | STRING_VALUE |
|---|---|---|---|---|---|
| 1 | 3 | obj | ? | ? | -344.20054327942256 |
| 2 | 3 | iter | ? | ? | 843 |
| 3 | 2 | R2 | ? | ? | 0.000026034476705... |
| 4 | 2 | F | ? | ? | 0.000104140618067... |
| 5 | 2 | AIC | ? | ? | 36.87405019918088 |
| 6 | 2 | BIC | ? | ? | 36.24932860686504 |
| 7 | 2 | MSE | ? | ? | 15.078397264930672 |
| 8 | 2 | RMSE | ? | ? | 3.883091199666919 |
| 9 | 2 | MAE | ? | ? | 2.5887144561882036 |
| 10 | 2 | MAPE | ? | ? | 59.80975993802132 |
| 11 | 2 | F_PVALUE | ? | ? | 0.9923464677500982 |
| 12 | 1 | R2 | ? | ? | 0.000026034476705... |
| 13 | 1 | F | ? | ? | 0.000104140618067... |
| 14 | 1 | AIC | ? | ? | 36.87405019918088 |
| 15 | 1 | BIC | ? | ? | 36.24932860686504 |
| 16 | 1 | MSE | ? | ? | 15.078397264930672 |
| 17 | 1 | RMSE | ? | ? | 3.883091199666919 |
| 18 | 1 | MAE | ? | ? | 2.5887144561882027 |
| 19 | 1 | MAPE | ? | ? | 59.80975993802134 |
| 20 | 1 | F_PVALUE | ? | ? | 0.992346467750099 |

Testing Stat Table:

| | MODEL_ID | STAT_NAME | STAT_VALUE |
|---|---|---|---|
| 1 | 1 | MSE | 12.321788916464012 |
| 2 | 1 | RMSE | 3.5102405781461776 |
| 3 | 1 | MAE | 2.902519931242187 |
| 4 | 1 | MAPE | 13415.873542336769 |
| 5 | 1 | R2 | -0.1995720346331... |
| 6 | 2 | MSE | 12.321788916464008 |
| 7 | 2 | RMSE | 3.510240578146177 |
| 8 | 2 | MAE | 2.9025199312421863 |
| 9 | 2 | MAPE | 13415.873542336773 |
| 10 | 2 | R2 | -0.1995720346331... |
| 11 | 3 | MSE | 11.542959300893997 |
| 12 | 3 | RMSE | 3.3974930906322673 |
| 13 | 3 | MAE | 2.884594777744323 |
| 14 | 3 | MAPE | 10424.013597374002 |
| 15 | 3 | R2 | -0.1237500713682... |

Partition Table:

| | ID | PARTITION_TYPE |
|---|---|---|
| 1 | 2 | 1 |
| 2 | 3 | 1 |
| 3 | 0 | 1 |
| 4 | 5 | 1 |
| 5 | 4 | 1 |
| 6 | 8 | 1 |
| 7 | 1 | 2 |
| 8 | 9 | 2 |
| 9 | 6 | 2 |
| 10 | 7 | 2 |

Error message Table:

| | MODEL_ID | TIME_STAMP | ERROR_CODE | MESSAGE_CONTENT |
|---|---|---|---|---|
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

**Example 2: Use Model Comparison Regression with partition table**

Models compared in this example:

- Bi-Variate Geometric Regression with SVD
- Bi-Variate Geometric Regression with LU
- Bi-Variate Natural Logarithmic Regression
- Polynomial Regression
- Exponential Regression

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_MODEL_COMP_DATA_TBL;
CREATE COLUMN TABLE PAL_MODEL_COMP_DATA_TBL
    ("ID" INTEGER, "Y" DOUBLE, "X1" DOUBLE);
INSERT INTO PAL_MODEL_COMP_DATA_TBL VALUES (0, 6.879, 0.02);
INSERT INTO PAL_MODEL_COMP_DATA_TBL VALUES (1, 3.449, 0.50);
INSERT INTO PAL_MODEL_COMP_DATA_TBL VALUES (2,  6.635, 0.54);
INSERT INTO PAL_MODEL_COMP_DATA_TBL VALUES (3, 11.844, 1.04);
INSERT INTO PAL_MODEL_COMP_DATA_TBL VALUES (4,  2.786, 1.50);
INSERT INTO PAL_MODEL_COMP_DATA_TBL VALUES (5,  2.389, 0.04);
INSERT INTO PAL_MODEL_COMP_DATA_TBL VALUES (6, 0.011, 2.00);
INSERT INTO PAL_MODEL_COMP_DATA_TBL VALUES (7,  8.839, 2.04);
INSERT INTO PAL_MODEL_COMP_DATA_TBL VALUES (8,  4.689, 1.54);
INSERT INTO PAL_MODEL_COMP_DATA_TBL VALUES (9, 5.507, 1.00);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
```

```
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PARTITION_METHOD',0,null,null);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('TRAINING_PERCENT', null,0.6,null);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('TESTING_PERCENT', null,0.4,null);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('VALIDATION_PERCENT', null,0.0,null);
DROP TABLE PAL_MLR_PARTITION_TBL;
CREATE COLUMN TABLE PAL_MLR_PARTITION_TBL("ID" INTEGER, "TYPE" INT);
CALL "_SYS_AFL"."PAL_PARTITION"(PAL_MODEL_COMP_DATA_TBL, #PAL_PARAMETER_TBL,
PAL_MLR_PARTITION_TBL) with overview;
DROP TABLE PAL_MC_CONFIG_TBL;
CREATE COLUMN TABLE PAL_MC_CONFIG_TBL
("MODEL_ID" INT,
"MODEL_NAME" VARCHAR(20),
"MODEL_CONFIG" VARCHAR(100)
);
INSERT INTO PAL_MC_CONFIG_TBL VALUES (1, 'GEOR', '{"ALG": 2, "ADJUSTED_R2":1}');
--Use svd and output adjusted R square
INSERT INTO PAL_MC_CONFIG_TBL VALUES (2, 'GEOR', '{"ADJUSTED_R2":1}'); --use
Doolittle decomposition and output adjusted R square
INSERT INTO PAL_MC_CONFIG_TBL VALUES (3, 'LOGARITHMIC', '{}'); --Do not have
specific parameter setting
INSERT INTO PAL_MC_CONFIG_TBL VALUES (4, 'POLYNOMIAL', '{"POLYNOMIAL_NUM": 2}');
INSERT INTO PAL_MC_CONFIG_TBL VALUES (5, 'EXP', '{}');
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('HAS_ID',1,NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('DEPENDENT_VARIABLE',NULL,NULL,'Y');
CALL
_SYS_AFL.PAL_MODEL_COMPARISON_REGRESSION_WITH_PARTITION_TABLE(PAL_MODEL_COMP_DATA
_TBL, PAL_MC_CONFIG_TBL, #PAL_PARAMETER_TBL,
PAL_MLR_PARTITION_TBL, ?, ?, ?, ?, ?);
```

**Expected Results:**

Model Table:

| | MODEL_ID | ROW_INDEX | MODEL_CONTENT |
|---|---|---|---|
| 1 | 3 | 1 | <PMML version="4.0" xmlns="http://www.dmg.org/PMML-4_0" xmlns:xsi="http://www.... |
| 2 | 1 | 1 | <PMML version="4.0" xmlns="http://www.dmg.org/PMML-4_0" xmlns:xsi="http://www.... |
| 3 | 2 | 1 | <PMML version="4.0" xmlns="http://www.dmg.org/PMML-4_0" xmlns:xsi="http://www.... |
| 4 | 5 | 1 | <PMML version="4.0" xmlns="http://www.dmg.org/PMML-4_0" xmlns:xsi="http://www.... |
| 5 | 4 | 1 | <PMML version="4.0" xmlns="http://www.dmg.org/PMML-4_0" xmlns:xsi="http://www.... |

Fitting Table:

| | MODEL_ID | DATA_ID | ACTUAL_VALUE | PREDICTED_VALUE |
|---|---|---|---|---|
| 1 | 3 | 1 | 3.449 | 5.5527899550137665 |
| 2 | 3 | 3 | 11.844 | 5.175403675707274 |
| 3 | 3 | 4 | 2.786 | 4.986679387388047 |
| 4 | 3 | 5 | 2.389 | 6.8542879368280225 |
| 5 | 1 | 1 | 3.449 | 2.4777326502973867 |
| 6 | 1 | 3 | 11.844 | 1.745285353345658 |
| 7 | 1 | 4 | 2.786 | 1.46473777810065 |
| 8 | 1 | 5 | 2.389 | 8.296742013776221 |
| 9 | 2 | 1 | 3.449 | 2.4777326502973867 |
| 10 | 2 | 3 | 11.844 | 1.745285353345658 |
| 11 | 2 | 4 | 2.786 | 1.46473777810065 |
| 12 | 2 | 5 | 2.389 | 8.296742013776223 |
| 13 | 5 | 1 | 3.449 | 6.3046432044340905 |
| 14 | 5 | 3 | 11.844 | 2.769650602893286 |
| 15 | 5 | 4 | 2.786 | 1.3744002926324792 |
| 16 | 5 | 5 | 2.389 | 12.70492952147331 |
| 17 | 4 | 1 | 3.449 | 6.211851971522245 |
| 18 | 4 | 3 | 11.844 | 5.378410341853854 |
| 19 | 4 | 4 | 2.786 | 4.8639447294599725 |
| 20 | 4 | 5 | 2.389 | 7.1173239504152495 |

Training Stat Table:

| | MODEL_ID | STAT_NAME | INT_VALUE | DOUBLE_VALUE | STRING_VALUE |
|---|---|---|---|---|---|
| 1 | 3 | R2 | ? | ? | 0.09216876798435258 |
| 2 | 3 | F | ? | ? | 0.40610529681694746 |
| 3 | 3 | MSE | ? | ? | 10.23430179163209 |
| 4 | 3 | RMSE | ? | ? | 3.199109531046427 |
| 5 | 3 | MAE | ? | ? | 1.8146733578018956 |
| 6 | 3 | MAPE | ? | ? | 7327.43839023573 |
| 7 | 1 | R2 | ? | ? | 0.10558852752040138 |
| 8 | 1 | F | ? | ? | 0.4722145489823624 |
| 9 | 1 | Adjusted R2 | ? | ? | -0.11801434059949... |
| 10 | 1 | MSE | ? | ? | 7.5888766241487025 |
| 11 | 1 | RMSE | ? | ? | 2.7547915754460814 |
| 12 | 1 | MAE | ? | ? | 1.7576447161434425 |
| 13 | 1 | MAPE | ? | ? | 69.65517488122475 |
| 14 | 2 | R2 | ? | ? | 0.10558852752040138 |
| 15 | 2 | F | ? | ? | 0.4722145489823625 |
| 16 | 2 | Adjusted R2 | ? | ? | -0.11801434059949... |
| 17 | 2 | MSE | ? | ? | 7.5888766241487025 |
| 18 | 2 | RMSE | ? | ? | 2.7547915754460814 |
| 19 | 2 | MAE | ? | ? | 1.7576447161434425 |
| 20 | 2 | MAPE | ? | ? | 69.65517488122475 |
| 21 | 5 | R2 | ? | ? | 0.22756980285175676 |
| 22 | 5 | F | ? | ? | 1.1784614516207577 |
| 23 | 5 | MSE | ? | ? | 6.55389342298333 |
| 24 | 5 | RMSE | ? | ? | 2.560057308534973 |
| 25 | 5 | MAE | ? | ? | 1.5700750664827676 |
| 26 | 5 | MAPE | ? | ? | 62.11770698046395 |
| 27 | 4 | R2 | ? | ? | 0.12413984383429011 |
| 28 | 4 | F | ? | ? | 0.21260216535778306 |
| 29 | 4 | MSE | ? | ? | 13.165174097485975 |
| 30 | 4 | RMSE | ? | ? | 3.628384502431623 |
| 31 | 4 | MAE | ? | ? | 1.6394091299734246 |
| 32 | 4 | MAPE | ? | ? | 6825.611559377176 |
| 33 | 4 | F_PVALUE | ? | ? | 0.819694753542237 |

Testing Stat Table:

| | MODEL_ID | STAT_NAME | STAT_VALUE |
|---|---|---|---|
| 1 | 3 | MSE | 18.419473809010942 |
| 2 | 3 | RMSE | 4.291791445190568 |
| 3 | 3 | MAE | 3.8595884008806403 |
| 4 | 3 | MAPE | 95.80041598971165 |
| 5 | 3 | R2 | -0.2096131139605... |
| 6 | 1 | MSE | 34.893636834876105 |
| 7 | 1 | RMSE | 5.907083615023247 |
| 8 | 1 | MAE | 4.5747465580081315 |
| 9 | 1 | MAPE | 102.03490507905553 |
| 10 | 1 | R2 | -1.291477006720677 |
| 11 | 2 | MSE | 34.893636834876105 |
| 12 | 2 | RMSE | 5.907083615023247 |
| 13 | 2 | MAE | 4.5747465580081315 |
| 14 | 2 | MAPE | 102.03490507905555 |
| 15 | 2 | R2 | -1.291477006720677 |
| 16 | 5 | MSE | 49.72738267941156 |
| 17 | 5 | RMSE | 7.0517645082214395 |
| 18 | 5 | MAE | 5.914380457595409 |
| 19 | 5 | MAPE | 160.47224590367622 |
| 20 | 5 | R2 | -2.2656141448798... |
| 21 | 4 | MSE | 19.028025580707926 |
| 22 | 4 | RMSE | 4.36211251353148 |
| 23 | 4 | MAE | 4.008677577385903 |
| 24 | 4 | MAPE | 101.80033385117628 |
| 25 | 4 | R2 | -0.2495769159236... |

Error message Table:

| | MODEL_ID | TIME_STAMP | ERROR_CODE | MESSAGE_CONTENT |
|---|---|---|---|---|
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

**Example 3: Use Model Comparison Regression with training and testing tables**

Models compared in this example:

- Multiple Linear Regression with cyclical coordinate descent method to solve elastic net regularized multiple linear regression and set ALPHA to 0.5
- Multiple Linear Regression with ADMM and use cross validation

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_MODEL_COMP_TRAINING_DATA_TBL;
```

```
CREATE COLUMN TABLE PAL_MODEL_COMP_TRAINING_DATA_TBL
    ("ID" INTEGER, "Y" DOUBLE, "X1" DOUBLE);
INSERT INTO PAL_MODEL_COMP_TRAINING_DATA_TBL VALUES (0, 6.879, 0.02);
INSERT INTO PAL_MODEL_COMP_TRAINING_DATA_TBL VALUES (1, 3.449, 0.50);
INSERT INTO PAL_MODEL_COMP_TRAINING_DATA_TBL VALUES (2,  6.635, 0.54);
INSERT INTO PAL_MODEL_COMP_TRAINING_DATA_TBL VALUES (3, 11.844, 1.04);
INSERT INTO PAL_MODEL_COMP_TRAINING_DATA_TBL VALUES (4,  2.786, 1.50);
INSERT INTO PAL_MODEL_COMP_TRAINING_DATA_TBL VALUES (5,  2.389, 0.04);
DROP TABLE PAL_MODEL_COMP_TESTING_DATA_TBL;
CREATE COLUMN TABLE PAL_MODEL_COMP_TESTING_DATA_TBL
    ("ID" INTEGER, "Y" DOUBLE, "X1" DOUBLE);
INSERT INTO PAL_MODEL_COMP_TESTING_DATA_TBL VALUES (6, 0.011, 2.00);
INSERT INTO PAL_MODEL_COMP_TESTING_DATA_TBL VALUES (7,  8.839, 2.04);
INSERT INTO PAL_MODEL_COMP_TESTING_DATA_TBL VALUES (8,  4.689, 1.54);
INSERT INTO PAL_MODEL_COMP_TESTING_DATA_TBL VALUES (9, 5.507, 1.00);
DROP TABLE PAL_MC_CONFIG_TBL;
CREATE COLUMN TABLE PAL_MC_CONFIG_TBL
("MODEL_ID" INT,
"MODEL_NAME" VARCHAR(20),
"MODEL_CONFIG" VARCHAR(1000)
);
INSERT INTO PAL_MC_CONFIG_TBL VALUES (1, 'MLR', '{"ALG": 4, "ENET_ALPHA":0.5}');
--Cyclical coordinate descent method to solve elastic net regularized multiple
linear regression
INSERT INTO PAL_MC_CONFIG_TBL VALUES (2, 'MLR', '{"ALG": 6, "RESAMPLING_METHOD":
"cv", "EVALUATION_METRIC": "RMSE", "FOLD_NUM": 2, "PARAM_SEARCH_STRATEGY":
"grid", "ENET_LAMBDA_VALUES": "{0.01,0.001,0.05}", "ENET_ALPHA_VALUES":
"{0.1,0.5,0.03}"}');
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" NVARCHAR(256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" NVARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('HAS_ID',1,NULL,NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('DEPENDENT_VARIABLE',NULL,NULL,'Y');
CALL
_SYS_AFL.PAL_MODEL_COMPARISON_REGRESSION_WITH_TESTING_TABLE(PAL_MODEL_COMP_TRAINI
NG_DATA_TBL, PAL_MODEL_COMP_TESTING_DATA_TBL, PAL_MC_CONFIG_TBL,
#PAL_PARAMETER_TBL, ?, ?, ?, ?, ?);
```

**Expected Results:**

Model Table:

| | MODEL_ID | ROW_INDEX | MODEL_CONTENT |
|---|---|---|---|
| 1 | 1 | 1 | <PMML version="4.0" xmlns="http://www.dmg.org/PMML-4_0" xmlns:xsi="http://www.... |
| 2 | 2 | 1 | <PMML version="4.0" xmlns="http://www.dmg.org/PMML-4_0" xmlns:xsi="http://www.... |

Fitting Table:

| | MODEL_ID | DATA_ID | ACTUAL_VALUE | PREDICTED_VALUE |
|---|---|---|---|---|
| 1 | 1 | 6 | 0.011 | 6.851895136778116 |
| 2 | 1 | 7 | 8.839 | 6.886006958886578 |
| 3 | 1 | 8 | 4.689 | 6.459609182530795 |
| 4 | 1 | 9 | 5.507 | 5.99909958406655 |
| 5 | 2 | 6 | 0.011 | 6.7469872141353... |
| 6 | 2 | 7 | 8.839 | 6.778087325545908 |
| 7 | 2 | 8 | 4.689 | 6.389335932913614 |
| 8 | 2 | 9 | 5.507 | 5.969484428870737 |

Training Stat Table:

| | MODEL_ID | STAT_NAME | INT_VALUE | DOUBLE_VALUE | STRING_VALUE |
|---|---|---|---|---|---|
| 1 | 1 | R2 | ? | ? | 0.018787630034529434 |
| 2 | 1 | F | ? | ? | 0.07658945447330882 |
| 3 | 1 | AIC | ? | ? | 37.16533542092462 |
| 4 | 1 | BIC | ? | ? | 36.540613828608784 |
| 5 | 1 | MSE | ? | ? | 15.82847624318109 |
| 6 | 1 | RMSE | ? | ? | 3.978501758599723 |
| 7 | 1 | MAE | ? | ? | 2.8515383405321817 |
| 8 | 1 | MAPE | ? | ? | 66.42531112978214 |
| 9 | 1 | F_PVALUE | ? | ? | 0.7956856272668358 |
| 10 | 2 | timeout | ? | ? | FALSE |
| 11 | 2 | TEST_RMSE | ? | ? | 5.37179, 1.96365 |
| 12 | 2 | TEST_RMS... | ? | ? | 3.66772 |
| 13 | 2 | TEST_RMS... | ? | ? | 5.80771 |
| 14 | 2 | EVAL_RESU... | ? | ? | {"candidates":[{"TES... |
| 15 | 2 | EVAL_RESU... | ? | ? | meters":"ENET_ALP... |
| 16 | 2 | R2 | ? | ? | 0.018641180273849756 |
| 17 | 2 | F | ? | ? | 0.06365290284168483 |
| 18 | 2 | AIC | ? | ? | 37.16623087742609 |
| 19 | 2 | BIC | ? | ? | 36.54150928511026 |
| 20 | 2 | MSE | ? | ? | 15.830838704793573 |
| 21 | 2 | RMSE | ? | ? | 3.978798650948999 |
| 22 | 2 | MAE | ? | ? | 2.846016870919403 |
| 23 | 2 | MAPE | ? | ? | 66.28568010705624 |
| 24 | 2 | ENET_ALP... | 0 | 0.5 | |
| 25 | 2 | ENET_LAM... | 0 | 0.05 | |

Testing Stat Table:

| | MODEL_ID | STAT_NAME | STAT_VALUE |
|---|---|---|---|
| 1 | 1 | MSE | 13.497311742233192 |
| 2 | 1 | RMSE | 3.6738687704153494 |
| 3 | 1 | MAE | 2.764149236122221 |
| 4 | 1 | MAPE | 15564.686943605131 |
| 5 | 1 | R2 | -0.3580737192249... |
| 6 | 2 | MSE | 13.181479733106295 |
| 7 | 2 | RMSE | 3.6306307624304477 |
| 8 | 2 | MAE | 2.7399300625934417 |
| 9 | 2 | MAPE | 15326.055969799547 |
| 10 | 2 | R2 | -0.3262953059026... |

Error message Table:

| | MODEL_ID | TIME_STAMP | ERROR_CODE | MESSAGE_CONTENT |
|---|---|---|---|---|
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

# 8 End-to-End Scenarios

This section provides end-to-end scenarios of predictive analysis with PAL algorithms.

## 8.1 Scenario: Predict Segmentation of New Customers for a Supermarket

We wish to predict segmentation/clustering of new customers for a supermarket. First use the K-means function in PAL to perform segmentation/clustering for existing customers in the supermarket. The output can then be used as the training data for the C4.5 Decision Tree function to predict new customers' segmentation/clustering.

### Technology Background

- K-means clustering is a method of cluster analysis whereby the algorithm partitions N observations or records into K clusters, in which each observation belongs to the cluster with the nearest center. It is one of the most commonly used algorithms in clustering method.
- Decision trees are powerful and popular tools for classification and prediction. Decision tree learning, used in statistics, data mining, and machine learning uses a decision tree as a predictive model which maps the observations about an item to the conclusions about the item's target value.

### Implementation Steps

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

**Step 1**

Input customer data and use the K-means function to partition the data set into K clusters. In this example, nine rows of data will be input. K equals 3, which means the customers will be partitioned into three levels.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_KMEANS_DATA_TBL;
CREATE COLUMN TABLE PAL_KMEANS_DATA_TBL(
    "ID" INT,
    "AGE" DOUBLE,
    "INCOME" DOUBLE
```

```
);
INSERT INTO PAL_KMEANS_DATA_TBL VALUES (0 , 20, 100000);
INSERT INTO PAL_KMEANS_DATA_TBL VALUES (1 , 21, 101000);
INSERT INTO PAL_KMEANS_DATA_TBL VALUES (2 , 22, 102000);
INSERT INTO PAL_KMEANS_DATA_TBL VALUES (3 , 30, 200000);
INSERT INTO PAL_KMEANS_DATA_TBL VALUES (4 , 31, 201000);
INSERT INTO PAL_KMEANS_DATA_TBL VALUES (5 , 32, 202000);
INSERT INTO PAL_KMEANS_DATA_TBL VALUES (6 , 40, 400000);
INSERT INTO PAL_KMEANS_DATA_TBL VALUES (7 , 41, 401000);
INSERT INTO PAL_KMEANS_DATA_TBL VALUES (8 , 42, 402000);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FUNCTION', NULL, NULL, 'KMEANS');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('N_CLUSTERS', 3, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('INIT', 1, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('DISTANCE_LEVEL', 2, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('MAX_ITER', 100, NULL, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('TOL', NULL, 0.000001, NULL);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('NORMALIZATION', 0, NULL, NULL);
DROP TABLE PAL_KMEANS_RESASSIGN_TBL;
CREATE COLUMN TABLE PAL_KMEANS_RESASSIGN_TBL (
    "ID" INT,
    "CENTER_ASSIGN" INT,
    "DISTANCE" DOUBLE,
    "SLIGHT_SILHOUETTE" DOUBLE
);
DROP TABLE PAL_KMEANS_CENTERS_TBL;
CREATE COLUMN TABLE PAL_KMEANS_CENTERS_TBL (
    "CENTER_ID" INT,
    "V000" NVARCHAR(1000),
    "V001" NVARCHAR(1000)
);
CALL "_SYS_AFL"."PAL_UNIFIED_CLUSTERING"(PAL_KMEANS_DATA_TBL,
#PAL_PARAMETER_TBL,
            PAL_KMEANS_RESASSIGN_TBL, PAL_KMEANS_CENTERS_TBL, ?, ?, ?, ?, ?)with
overview;
DROP TABLE PAL_KMEANS_RESULT_TBL;
CREATE COLUMN TABLE PAL_KMEANS_RESULT_TBL(
    "AGE" DOUBLE,
    "INCOME" DOUBLE,
    "LEVEL" VARCHAR (10)
);
TRUNCATE TABLE PAL_KMEANS_RESULT_TBL;
INSERT INTO PAL_KMEANS_RESULT_TBL(
    SELECT PAL_KMEANS_DATA_TBL.AGE, PAL_KMEANS_DATA_TBL.INCOME,
PAL_KMEANS_RESASSIGN_TBL.CENTER_ASSIGN
    FROM PAL_KMEANS_RESASSIGN_TBL
    INNER JOIN PAL_KMEANS_DATA_TBL
        ON PAL_KMEANS_RESASSIGN_TBL.ID = PAL_KMEANS_DATA_TBL.ID
);
SELECT * FROM PAL_KMEANS_RESULT_TBL;
```

The result should show the following in PAL_KMEANS_RESULT_TBL:

| | AGE | INCOME | LEVEL |
|---|---|---|---|
| 1 | 20 | 100,000 | 0 |
| 2 | 21 | 101,000 | 0 |
| 3 | 22 | 102,000 | 0 |
| 4 | 30 | 200,000 | 1 |
| 5 | 31 | 201,000 | 1 |
| 6 | 32 | 202,000 | 1 |
| 7 | 40 | 400,000 | 2 |
| 8 | 41 | 401,000 | 2 |
| 9 | 42 | 402,000 | 2 |

### Step 2

Use the above output as the training data of C4.5 Decision Tree. The C4.5 Decision Tree function will generate a tree model which maps the observations about an item to the conclusions about the item's target value.

```
SET SCHEMA DM_PAL;
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FUNCTION', null, null, 'DT');
INSERT INTO #PAL_PARAMETER_TBL VALUES ('ALGORITHM', 1, null, null);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('PERCENTAGE', NULL, 1.0, NULL);
DROP TABLE PAL_C45_TREEMODEL_TBL;
CREATE COLUMN TABLE PAL_C45_TREEMODEL_TBL (
    "ROW_INDEX" INTEGER,
    "PART_INDEX" INTEGER,
    "TREEMODEL" NCLOB
);
CALL "_SYS_AFL"."PAL_UNIFIED_CLASSIFICATION"(PAL_KMEANS_RESULT_TBL,
#PAL_PARAMETER_TBL, PAL_C45_TREEMODEL_TBL, ?, ?, ?, ?, ?, ?, ?) with overview;
SELECT * FROM PAL_C45_TREEMODEL_TBL;
```

### Step 3

Use the above tree model to map each new customer to the corresponding level he or she belongs to.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_PCDT_DATA_TBL;
CREATE COLUMN TABLE PAL_PCDT_DATA_TBL (
    "ID" INTEGER,
    "AGE" DOUBLE,
    "INCOME" DOUBLE
);
INSERT INTO PAL_PCDT_DATA_TBL VALUES (10, 20, 100003);
INSERT INTO PAL_PCDT_DATA_TBL VALUES (11, 30, 200003);
INSERT INTO PAL_PCDT_DATA_TBL VALUES (12, 40, 400003);
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('FUNCTION', NULL, NULL, 'DT');
```

```
CALL "_SYS_AFL"."PAL_UNIFIED_CLASSIFICATION_PREDICT"(PAL_PCDT_DATA_TBL,
PAL_C45_TREEMODEL_TBL, #PAL_PARAMETER_TBL, ?, ?)
```

The expected prediction result is as follows:

| | ID | SCORE | CONFIDENCE | REASON_CODE |
|---|---|---|---|---|
| 1 | 10 | 0 | 1 | [{"attr":"AGE","pct":100,"val":0.66666666666666674}] |
| 2 | 11 | 1 | 1 | [{"attr":"AGE","pct":100,"val":0.66666666666666674}] |
| 3 | 12 | 2 | 1 | [{"attr":"AGE","pct":100,"val":0.66666666666666674}] |

## 8.2 Scenario: Analyze the Cash Flow of an Investment on a New Product

We wish to do an analysis of the cash flow of an investment required to create a new product. Projected estimates are given for the product revenue, product costs, overheads, and capital investment for each year of the analysis, from which the cash flow can be calculated. For capital investment appraisal the cash flows are summed for each year and discounted for future values, in other words the net present value of the cash flow is derived as a single value measuring the benefit of the investment.

The projected estimates are single point estimates of each data point and the analysis provides a single point value of project net present value (NPV). This is referred to as deterministic modeling, which is in contrast to probabilistic modeling whereby we examine the probability of outcomes, for example, what is the probability of a NPV greater than zero. Probabilistic modeling is also called Monte Carlo Simulation.

Monte Carlo Simulation is used in our example to estimate the net present value (NPV) of the investment. The equations used in the simulation are:

For each year `i=0, 1, ..., k`

```
Product margin(i) = product revenue(i) – product cost(i)
```

```
Total profit(i) = product margin(i) – overhead(i)
```

```
Cash flow(i) = total profit(i) – capital investment(i)
```

Suppose the simulation covers `k` years' time periods and the discount rate is `r`, the net present value of the investment is defined as:

$$\text{NPV of investment} = \sum_{i=0}^{k} \frac{\text{Cash flow}_i}{(1+r)^i} = \sum_{i=0}^{k} \frac{\text{product revenue}_i - \text{product cost}_i - \text{overhead}_i - \text{capital flow}_i}{(1+r)^i}$$

### Technology Background

Monte Carlo Simulation is a computational algorithm that repeatedly generates random samples to compute numerical results based on a formula or model in order to obtain the unknown probability distribution of an event or outcome.

In PAL, the Random Distribution Sampling, Distribution Fitting, and Cumulative Distribution algorithms may be used for Monte Carlo Simulation.

## Implementation Steps

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

**Step 1**

Input the given estimates (single point deterministic values) for product revenue, product costs, overheads, and capital investment. In this example, the time periods are 5 (from year 1 to year 5).

The probability distribution for each variable is assumed as follows:

- Product Revenue:
  Normal distribution and the mean and standard deviation are listed in the following table.
- Product Costs:
  Normal distribution and the mean and standard deviation are listed in the following table.
- Overheads:
  Uniform distribution and min and max values are listed in the following table.
- Capital Investment (for year 1 and year 2)
  Gamma distribution and shape and scale values are listed in the following table.

| | Year 1 | Year 2 | Year 3 | Year 4 | Year 5 |
|---|---|---|---|---|---|
| **Product Revenue** | | | | | |
| Mean | £0 | £3,000 | £8,000 | £18,000 | £30,000 |
| Standard Deviation | 0 | 300 | 800 | 1800 | 3000 |
| **Product Costs** | | | | | |
| Mean | £1,000 | £1,000 | £2,500 | £7,000 | £10,000 |
| Standard Deviation | 75 | 75 | 187.5 | 525 | 750 |
| **Overheads** | | | | | |
| Min | £1,400 | £1,800 | £2,200 | £2,600 | £3,000 |
| Max | £1,500 | £2,200 | £2,800 | £3,400 | £4,000 |
| **Capital Investment** | | | | | |
| Mean | £10,000 | £2,000 | | | |
| Standard Deviation | 500 | 100 | | | |

Run the Random Distribution Sampling algorithm for each variable and generate 1,000 sample sets. The number of sample sets is a choice for the analysis. The larger the value then the more smooth the output distribution and the closer it will be to a normal distribution.

```
SET SCHEMA DM_PAL;
----------------------------------------
---Random sampling process---------------
----------------------------------------
DROP TABLE PAL_DISTRRANDOM_DISTRPARAM_TAB;
CREATE COLUMN TABLE PAL_DISTRRANDOM_DISTRPARAM_TAB (
    NAME VARCHAR(50),
    VAL VARCHAR(50)
);
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
DROP TABLE PAL_DISTRRANDOM_RESULT_TAB_REVENUE;
CREATE COLUMN TABLE PAL_DISTRRANDOM_RESULT_TAB_REVENUE (
    ID INTEGER,
    RANDOM DOUBLE
);
DROP TABLE PAL_DISTRRANDOM_RESULT_TAB_COSTS;
CREATE COLUMN TABLE PAL_DISTRRANDOM_RESULT_TAB_COSTS (
    ID INTEGER,
    RANDOM DOUBLE
);
DROP TABLE PAL_DISTRRANDOM_RESULT_TAB_OVERHEADS;
CREATE COLUMN TABLE PAL_DISTRRANDOM_RESULT_TAB_OVERHEADS (
    ID INTEGER,
    RANDOM DOUBLE
);
DROP TABLE PAL_DISTRRANDOM_RESULT_TAB_INVESTMENT;
CREATE COLUMN TABLE PAL_DISTRRANDOM_RESULT_TAB_INVESTMENT (
    ID INTEGER,
    RANDOM DOUBLE
);
----------------------------
----------YEAR 1 -----------
-----Product revenue------
INSERT INTO PAL_DISTRRANDOM_DISTRPARAM_TAB VALUES ('DISTRIBUTIONNAME', 'NORMAL');
INSERT INTO PAL_DISTRRANDOM_DISTRPARAM_TAB VALUES ('MEAN', '0');
INSERT INTO PAL_DISTRRANDOM_DISTRPARAM_TAB VALUES ('VARIANCE', '0.0001');
INSERT INTO PAL_PARAMETER_TBL VALUES ('NUM_RANDOM',5000,null,null);
INSERT INTO PAL_PARAMETER_TBL VALUES ('SEED', 0, null, null);
CALL "_SYS_AFL"."PAL_DISTRIBUTION_RANDOM"(PAL_DISTRRANDOM_DISTRPARAM_TAB,
PAL_PARAMETER_TBL, PAL_DISTRRANDOM_RESULT_TAB_REVENUE) WITH OVERVIEW;
--SELECT * FROM PAL_DISTRRANDOM_RESULT_TAB_REVENUE;
---------Product Costs---------
DELETE FROM PAL_DISTRRANDOM_DISTRPARAM_TAB;
INSERT INTO PAL_DISTRRANDOM_DISTRPARAM_TAB VALUES ('DISTRIBUTIONNAME', 'NORMAL');
INSERT INTO PAL_DISTRRANDOM_DISTRPARAM_TAB VALUES ('MEAN', '1000');
INSERT INTO PAL_DISTRRANDOM_DISTRPARAM_TAB VALUES ('VARIANCE', '5625');
CALL "_SYS_AFL"."PAL_DISTRIBUTION_RANDOM"(PAL_DISTRRANDOM_DISTRPARAM_TAB,
PAL_PARAMETER_TBL, PAL_DISTRRANDOM_RESULT_TAB_COSTS) WITH OVERVIEW;
--SELECT * FROM PAL_DISTRRANDOM_RESULT_TAB_COSTS;
--------OVERHEADS---------
DELETE FROM PAL_DISTRRANDOM_DISTRPARAM_TAB;
INSERT INTO PAL_DISTRRANDOM_DISTRPARAM_TAB VALUES ('DISTRIBUTIONNAME',
'UNIFORM');
INSERT INTO PAL_DISTRRANDOM_DISTRPARAM_TAB VALUES ('MIN', '1400');
INSERT INTO PAL_DISTRRANDOM_DISTRPARAM_TAB VALUES ('MAX', '1500');
CALL "_SYS_AFL"."PAL_DISTRIBUTION_RANDOM"(PAL_DISTRRANDOM_DISTRPARAM_TAB,
PAL_PARAMETER_TBL, PAL_DISTRRANDOM_RESULT_TAB_OVERHEADS) WITH OVERVIEW;
```

```
--SELECT * FROM PAL_DISTRRANDOM_RESULT_TAB_OVERHEADS;
-------INVESTMENT---------
DELETE FROM PAL_DISTRRANDOM_DISTRPARAM_TAB;
INSERT INTO PAL_DISTRRANDOM_DISTRPARAM_TAB VALUES ('DISTRIBUTIONNAME', 'NORMAL');
INSERT INTO PAL_DISTRRANDOM_DISTRPARAM_TAB VALUES ('MEAN', '10000');
INSERT INTO PAL_DISTRRANDOM_DISTRPARAM_TAB VALUES ('VARIANCE', '250000');
CALL "_SYS_AFL"."PAL_DISTRIBUTION_RANDOM"(PAL_DISTRRANDOM_DISTRPARAM_TAB,
PAL_PARAMETER_TBL, PAL_DISTRRANDOM_RESULT_TAB_INVESTMENT) WITH OVERVIEW;
--SELECT * FROM PAL_DISTRRANDOM_RESULT_TAB_INVESTMENT;
--------calculate cash flow -------
DROP TABLE PAL_CASHFLOW_YEAR1;
CREATE COLUMN TABLE PAL_CASHFLOW_YEAR1(
    ID INTEGER,
    CASH DOUBLE
);
INSERT INTO PAL_CASHFLOW_YEAR1
    SELECT PAL_DISTRRANDOM_RESULT_TAB_REVENUE.ID,
           PAL_DISTRRANDOM_RESULT_TAB_REVENUE.RANDOM -
           PAL_DISTRRANDOM_RESULT_TAB_COSTS.RANDOM -
           PAL_DISTRRANDOM_RESULT_TAB_OVERHEADS.RANDOM -
           PAL_DISTRRANDOM_RESULT_TAB_INVESTMENT.RANDOM
    FROM PAL_DISTRRANDOM_RESULT_TAB_REVENUE
    LEFT JOIN PAL_DISTRRANDOM_RESULT_TAB_COSTS ON
PAL_DISTRRANDOM_RESULT_TAB_REVENUE.ID = PAL_DISTRRANDOM_RESULT_TAB_COSTS.ID
    LEFT JOIN PAL_DISTRRANDOM_RESULT_TAB_OVERHEADS ON
PAL_DISTRRANDOM_RESULT_TAB_REVENUE.ID=PAL_DISTRRANDOM_RESULT_TAB_OVERHEADS.ID
    LEFT JOIN PAL_DISTRRANDOM_RESULT_TAB_INVESTMENT ON
PAL_DISTRRANDOM_RESULT_TAB_REVENUE.ID=PAL_DISTRRANDOM_RESULT_TAB_INVESTMENT.ID;
--SELECT * FROM PAL_CASHFLOW_YEAR1;
----------------------------------
------------YEAR 2 --------------
--product revenue----
DELETE FROM PAL_DISTRRANDOM_DISTRPARAM_TAB;
INSERT INTO PAL_DISTRRANDOM_DISTRPARAM_TAB VALUES ('DISTRIBUTIONNAME', 'NORMAL');
INSERT INTO PAL_DISTRRANDOM_DISTRPARAM_TAB VALUES ('MEAN', '3000');
INSERT INTO PAL_DISTRRANDOM_DISTRPARAM_TAB VALUES ('VARIANCE', '90000');
DELETE FROM PAL_DISTRRANDOM_RESULT_TAB_REVENUE;
CALL "_SYS_AFL"."PAL_DISTRIBUTION_RANDOM"(PAL_DISTRRANDOM_DISTRPARAM_TAB,
PAL_PARAMETER_TBL, PAL_DISTRRANDOM_RESULT_TAB_REVENUE) WITH OVERVIEW;
--SELECT * FROM PAL_DISTRRANDOM_RESULT_TAB_REVENUE;
----Product Costs---------
DELETE FROM PAL_DISTRRANDOM_DISTRPARAM_TAB;
INSERT INTO PAL_DISTRRANDOM_DISTRPARAM_TAB VALUES ('DISTRIBUTIONNAME', 'NORMAL');
INSERT INTO PAL_DISTRRANDOM_DISTRPARAM_TAB VALUES ('MEAN', '1000');
INSERT INTO PAL_DISTRRANDOM_DISTRPARAM_TAB VALUES ('VARIANCE', '5625');
DELETE FROM PAL_DISTRRANDOM_RESULT_TAB_COSTS;
CALL "_SYS_AFL"."PAL_DISTRIBUTION_RANDOM"(PAL_DISTRRANDOM_DISTRPARAM_TAB,
PAL_PARAMETER_TBL, PAL_DISTRRANDOM_RESULT_TAB_COSTS) WITH OVERVIEW;
--SELECT * FROM PAL_DISTRRANDOM_RESULT_TAB_COSTS;
----OVERHEADS---------
DELETE FROM PAL_DISTRRANDOM_DISTRPARAM_TAB;
INSERT INTO PAL_DISTRRANDOM_DISTRPARAM_TAB VALUES ('DISTRIBUTIONNAME',
'UNIFORM');
INSERT INTO PAL_DISTRRANDOM_DISTRPARAM_TAB VALUES ('MIN', '1800');
INSERT INTO PAL_DISTRRANDOM_DISTRPARAM_TAB VALUES ('MAX', '2200');
DELETE FROM PAL_DISTRRANDOM_RESULT_TAB_OVERHEADS;
CALL "_SYS_AFL"."PAL_DISTRIBUTION_RANDOM"(PAL_DISTRRANDOM_DISTRPARAM_TAB,
PAL_PARAMETER_TBL, PAL_DISTRRANDOM_RESULT_TAB_OVERHEADS) WITH OVERVIEW;
--SELECT * FROM PAL_DISTRRANDOM_RESULT_TAB_OVERHEADS;
----INVESTMENT---------
DELETE FROM PAL_DISTRRANDOM_DISTRPARAM_TAB;
INSERT INTO PAL_DISTRRANDOM_DISTRPARAM_TAB VALUES ('DISTRIBUTIONNAME', 'NORMAL');
INSERT INTO PAL_DISTRRANDOM_DISTRPARAM_TAB VALUES ('MEAN', '2000');
INSERT INTO PAL_DISTRRANDOM_DISTRPARAM_TAB VALUES ('VARIANCE', '10000');
DELETE FROM PAL_DISTRRANDOM_RESULT_TAB_INVESTMENT;
CALL "_SYS_AFL"."PAL_DISTRIBUTION_RANDOM"(PAL_DISTRRANDOM_DISTRPARAM_TAB,
PAL_PARAMETER_TBL, PAL_DISTRRANDOM_RESULT_TAB_INVESTMENT) WITH OVERVIEW;
--SELECT * FROM PAL_DISTRRANDOM_RESULT_TAB_INVESTMENT;
```

```
--------calculate cash flow -------
DROP TABLE PAL_CASHFLOW_YEAR2;
CREATE COLUMN TABLE PAL_CASHFLOW_YEAR2(
    ID INTEGER,
    CASH DOUBLE
);
INSERT INTO PAL_CASHFLOW_YEAR2
    SELECT PAL_DISTRRANDOM_RESULT_TAB_REVENUE.ID,
           PAL_DISTRRANDOM_RESULT_TAB_REVENUE.RANDOM -
           PAL_DISTRRANDOM_RESULT_TAB_COSTS.RANDOM -
           PAL_DISTRRANDOM_RESULT_TAB_OVERHEADS.RANDOM -
           PAL_DISTRRANDOM_RESULT_TAB_INVESTMENT.RANDOM
    FROM PAL_DISTRRANDOM_RESULT_TAB_REVENUE
    LEFT JOIN PAL_DISTRRANDOM_RESULT_TAB_COSTS ON
PAL_DISTRRANDOM_RESULT_TAB_REVENUE.ID = PAL_DISTRRANDOM_RESULT_TAB_COSTS.ID
    LEFT JOIN PAL_DISTRRANDOM_RESULT_TAB_OVERHEADS ON
PAL_DISTRRANDOM_RESULT_TAB_REVENUE.ID=PAL_DISTRRANDOM_RESULT_TAB_OVERHEADS.ID
    LEFT JOIN PAL_DISTRRANDOM_RESULT_TAB_INVESTMENT ON
PAL_DISTRRANDOM_RESULT_TAB_REVENUE.ID=PAL_DISTRRANDOM_RESULT_TAB_INVESTMENT.ID;
--SELECT * FROM PAL_CASHFLOW_YEAR2;
-----------------------------
----------YEAR 3 ------------
----product revenue----
DELETE FROM PAL_DISTRRANDOM_DISTRPARAM_TAB;
INSERT INTO PAL_DISTRRANDOM_DISTRPARAM_TAB VALUES ('DISTRIBUTIONNAME', 'NORMAL');
INSERT INTO PAL_DISTRRANDOM_DISTRPARAM_TAB VALUES ('MEAN', '8000');
INSERT INTO PAL_DISTRRANDOM_DISTRPARAM_TAB VALUES ('VARIANCE', '640000');
DELETE FROM PAL_DISTRRANDOM_RESULT_TAB_REVENUE;
CALL "_SYS_AFL"."PAL_DISTRIBUTION_RANDOM"(PAL_DISTRRANDOM_DISTRPARAM_TAB,
PAL_PARAMETER_TBL, PAL_DISTRRANDOM_RESULT_TAB_REVENUE) WITH OVERVIEW;
--SELECT * FROM PAL_DISTRRANDOM_RESULT_TAB_REVENUE;
----Product Costs---------
DELETE FROM PAL_DISTRRANDOM_DISTRPARAM_TAB;
INSERT INTO PAL_DISTRRANDOM_DISTRPARAM_TAB VALUES ('DISTRIBUTIONNAME', 'NORMAL');
INSERT INTO PAL_DISTRRANDOM_DISTRPARAM_TAB VALUES ('MEAN', '2500');
INSERT INTO PAL_DISTRRANDOM_DISTRPARAM_TAB VALUES ('VARIANCE', '35156.25');
DELETE FROM PAL_DISTRRANDOM_RESULT_TAB_COSTS;
CALL "_SYS_AFL"."PAL_DISTRIBUTION_RANDOM"(PAL_DISTRRANDOM_DISTRPARAM_TAB,
PAL_PARAMETER_TBL, PAL_DISTRRANDOM_RESULT_TAB_COSTS) WITH OVERVIEW;
--SELECT * FROM PAL_DISTRRANDOM_RESULT_TAB_COSTS;
----OVERHEADS---------
DELETE FROM PAL_DISTRRANDOM_DISTRPARAM_TAB;
INSERT INTO PAL_DISTRRANDOM_DISTRPARAM_TAB VALUES ('DISTRIBUTIONNAME',
'UNIFORM');
INSERT INTO PAL_DISTRRANDOM_DISTRPARAM_TAB VALUES ('MIN', '2200');
INSERT INTO PAL_DISTRRANDOM_DISTRPARAM_TAB VALUES ('MAX', '2800');
DELETE FROM PAL_DISTRRANDOM_RESULT_TAB_OVERHEADS;
CALL "_SYS_AFL"."PAL_DISTRIBUTION_RANDOM"(PAL_DISTRRANDOM_DISTRPARAM_TAB,
PAL_PARAMETER_TBL, PAL_DISTRRANDOM_RESULT_TAB_OVERHEADS) WITH OVERVIEW;
--SELECT * FROM PAL_DISTRRANDOM_RESULT_TAB_OVERHEADS;
--------calculate cash flow -------
DROP TABLE PAL_CASHFLOW_YEAR3;
CREATE COLUMN TABLE PAL_CASHFLOW_YEAR3(
    ID INTEGER,
    CASH DOUBLE
);
INSERT INTO PAL_CASHFLOW_YEAR3
    SELECT PAL_DISTRRANDOM_RESULT_TAB_REVENUE.ID,
           PAL_DISTRRANDOM_RESULT_TAB_REVENUE.RANDOM -
           PAL_DISTRRANDOM_RESULT_TAB_COSTS.RANDOM -
           PAL_DISTRRANDOM_RESULT_TAB_OVERHEADS.RANDOM
    FROM PAL_DISTRRANDOM_RESULT_TAB_REVENUE
    LEFT JOIN PAL_DISTRRANDOM_RESULT_TAB_COSTS ON
PAL_DISTRRANDOM_RESULT_TAB_REVENUE.ID = PAL_DISTRRANDOM_RESULT_TAB_COSTS.ID
    LEFT JOIN PAL_DISTRRANDOM_RESULT_TAB_OVERHEADS ON
PAL_DISTRRANDOM_RESULT_TAB_REVENUE.ID=PAL_DISTRRANDOM_RESULT_TAB_OVERHEADS.ID;
--SELECT * FROM PAL_CASHFLOW_YEAR3;
-----------------------------
```

```
--------YEAR 4 --------------
--Product revenue---------
DELETE FROM PAL_DISTRRANDOM_DISTRPARAM_TAB;
INSERT INTO PAL_DISTRRANDOM_DISTRPARAM_TAB VALUES ('DISTRIBUTIONNAME', 'NORMAL');
INSERT INTO PAL_DISTRRANDOM_DISTRPARAM_TAB VALUES ('MEAN', '18000');
INSERT INTO PAL_DISTRRANDOM_DISTRPARAM_TAB VALUES ('VARIANCE', '3240000');
DELETE FROM PAL_DISTRRANDOM_RESULT_TAB_REVENUE;
CALL "_SYS_AFL"."PAL_DISTRIBUTION_RANDOM"(PAL_DISTRRANDOM_DISTRPARAM_TAB,
PAL_PARAMETER_TBL, PAL_DISTRRANDOM_RESULT_TAB_REVENUE) WITH OVERVIEW;
--SELECT * FROM PAL_DISTRRANDOM_RESULT_TAB_REVENUE;
----Product Costs---------
DELETE FROM PAL_DISTRRANDOM_DISTRPARAM_TAB;
INSERT INTO PAL_DISTRRANDOM_DISTRPARAM_TAB VALUES ('DISTRIBUTIONNAME', 'NORMAL');
INSERT INTO PAL_DISTRRANDOM_DISTRPARAM_TAB VALUES ('MEAN', '7000');
INSERT INTO PAL_DISTRRANDOM_DISTRPARAM_TAB VALUES ('VARIANCE', '275625');
DELETE FROM PAL_DISTRRANDOM_RESULT_TAB_COSTS;
CALL "_SYS_AFL"."PAL_DISTRIBUTION_RANDOM"(PAL_DISTRRANDOM_DISTRPARAM_TAB,
PAL_PARAMETER_TBL, PAL_DISTRRANDOM_RESULT_TAB_COSTS) WITH OVERVIEW;
--SELECT * FROM PAL_DISTRRANDOM_RESULT_TAB_COSTS;
----OVERHEADS---------
DELETE FROM PAL_DISTRRANDOM_DISTRPARAM_TAB;
INSERT INTO PAL_DISTRRANDOM_DISTRPARAM_TAB VALUES ('DISTRIBUTIONNAME',
'UNIFORM');
INSERT INTO PAL_DISTRRANDOM_DISTRPARAM_TAB VALUES ('MIN', '2600');
INSERT INTO PAL_DISTRRANDOM_DISTRPARAM_TAB VALUES ('MAX', '3400');
DELETE FROM PAL_DISTRRANDOM_RESULT_TAB_OVERHEADS;
CALL "_SYS_AFL"."PAL_DISTRIBUTION_RANDOM"(PAL_DISTRRANDOM_DISTRPARAM_TAB,
PAL_PARAMETER_TBL, PAL_DISTRRANDOM_RESULT_TAB_OVERHEADS) WITH OVERVIEW;
--SELECT * FROM PAL_DISTRRANDOM_RESULT_TAB_OVERHEADS;
--------calculate cash flow -------
DROP TABLE PAL_CASHFLOW_YEAR4;
CREATE COLUMN TABLE PAL_CASHFLOW_YEAR4(
    ID INTEGER,
    CASH DOUBLE
);
INSERT INTO PAL_CASHFLOW_YEAR4
    SELECT PAL_DISTRRANDOM_RESULT_TAB_REVENUE.ID,
           PAL_DISTRRANDOM_RESULT_TAB_REVENUE.RANDOM -
           PAL_DISTRRANDOM_RESULT_TAB_COSTS.RANDOM -
           PAL_DISTRRANDOM_RESULT_TAB_OVERHEADS.RANDOM
    FROM PAL_DISTRRANDOM_RESULT_TAB_REVENUE
    LEFT JOIN PAL_DISTRRANDOM_RESULT_TAB_COSTS ON
PAL_DISTRRANDOM_RESULT_TAB_REVENUE.ID = PAL_DISTRRANDOM_RESULT_TAB_COSTS.ID
    LEFT JOIN PAL_DISTRRANDOM_RESULT_TAB_OVERHEADS ON
PAL_DISTRRANDOM_RESULT_TAB_REVENUE.ID=PAL_DISTRRANDOM_RESULT_TAB_OVERHEADS.ID;
--SELECT * FROM PAL_CASHFLOW_YEAR4;
----------------------------
-------YEAR 5 --------------
--Product revenue----
DELETE FROM PAL_DISTRRANDOM_DISTRPARAM_TAB;
INSERT INTO PAL_DISTRRANDOM_DISTRPARAM_TAB VALUES ('DISTRIBUTIONNAME', 'NORMAL');
INSERT INTO PAL_DISTRRANDOM_DISTRPARAM_TAB VALUES ('MEAN', '30000');
INSERT INTO PAL_DISTRRANDOM_DISTRPARAM_TAB VALUES ('VARIANCE', '9000000');
DELETE FROM PAL_DISTRRANDOM_RESULT_TAB_REVENUE;
CALL "_SYS_AFL"."PAL_DISTRIBUTION_RANDOM"(PAL_DISTRRANDOM_DISTRPARAM_TAB,
PAL_PARAMETER_TBL, PAL_DISTRRANDOM_RESULT_TAB_REVENUE) WITH OVERVIEW;
--SELECT * FROM PAL_DISTRRANDOM_RESULT_TAB_REVENUE;
----Product Costs---------
DELETE FROM PAL_DISTRRANDOM_DISTRPARAM_TAB;
INSERT INTO PAL_DISTRRANDOM_DISTRPARAM_TAB VALUES ('DISTRIBUTIONNAME', 'NORMAL');
INSERT INTO PAL_DISTRRANDOM_DISTRPARAM_TAB VALUES ('MEAN', '10000');
INSERT INTO PAL_DISTRRANDOM_DISTRPARAM_TAB VALUES ('VARIANCE', '562500');
DELETE FROM PAL_DISTRRANDOM_RESULT_TAB_COSTS;
CALL "_SYS_AFL"."PAL_DISTRIBUTION_RANDOM"(PAL_DISTRRANDOM_DISTRPARAM_TAB,
PAL_PARAMETER_TBL, PAL_DISTRRANDOM_RESULT_TAB_COSTS) WITH OVERVIEW;
--SELECT * FROM PAL_DISTRRANDOM_RESULT_TAB_COSTS;
----OVERHEADS---------
DELETE FROM PAL_DISTRRANDOM_DISTRPARAM_TAB;
```

```
INSERT INTO PAL_DISTRRANDOM_DISTRPARAM_TAB VALUES ('DISTRIBUTIONNAME',
'UNIFORM');
INSERT INTO PAL_DISTRRANDOM_DISTRPARAM_TAB VALUES ('MIN', '3000');
INSERT INTO PAL_DISTRRANDOM_DISTRPARAM_TAB VALUES ('MAX', '4000');
DELETE FROM PAL_DISTRRANDOM_RESULT_TAB_OVERHEADS;
CALL "_SYS_AFL"."PAL_DISTRIBUTION_RANDOM"(PAL_DISTRRANDOM_DISTRPARAM_TAB,
PAL_PARAMETER_TBL, PAL_DISTRRANDOM_RESULT_TAB_OVERHEADS) WITH OVERVIEW;
--SELECT * FROM PAL_DISTRRANDOM_RESULT_TAB_OVERHEADS;
--------calculate cash flow -------
DROP TABLE PAL_CASHFLOW_YEAR5;
CREATE COLUMN TABLE PAL_CASHFLOW_YEAR5(
    ID INTEGER,
    CASH DOUBLE
);
INSERT INTO PAL_CASHFLOW_YEAR5
    SELECT PAL_DISTRRANDOM_RESULT_TAB_REVENUE.ID,
           PAL_DISTRRANDOM_RESULT_TAB_REVENUE.RANDOM -
           PAL_DISTRRANDOM_RESULT_TAB_COSTS.RANDOM -
           PAL_DISTRRANDOM_RESULT_TAB_OVERHEADS.RANDOM
    FROM PAL_DISTRRANDOM_RESULT_TAB_REVENUE
    LEFT JOIN PAL_DISTRRANDOM_RESULT_TAB_COSTS ON
PAL_DISTRRANDOM_RESULT_TAB_REVENUE.ID = PAL_DISTRRANDOM_RESULT_TAB_COSTS.ID
    LEFT JOIN PAL_DISTRRANDOM_RESULT_TAB_OVERHEADS ON
PAL_DISTRRANDOM_RESULT_TAB_REVENUE.ID = PAL_DISTRRANDOM_RESULT_TAB_OVERHEADS.ID;
--SELECT * FROM PAL_CASHFLOW_YEAR5;
```

### Step 2

Calculate the net present value of the investment by the following equation for each sampling.

$$\text{NPV of investment} = \sum_{i=0}^{k} \frac{\text{Cash flow}_i}{(1+r)^i} = \sum_{i=0}^{k} \frac{\text{product revenue}_i\text{-product cost}_i\text{-overhead}_i\text{-capital flow}_i}{(1+r)^i}$$

```
SET SCHEMA DM_PAL;
---- calculate net present value of investment ----
DROP TABLE NPV;
CREATE COLUMN TABLE NPV (
    NPVALUE DOUBLE
);
INSERT INTO NPV SELECT PAL_CASHFLOW_YEAR1.CASH +
                       PAL_CASHFLOW_YEAR2.CASH/1.05 +
                       PAL_CASHFLOW_YEAR3.CASH/POWER(1.05,2) +
                       PAL_CASHFLOW_YEAR4.CASH/POWER(1.05,3) +
                       PAL_CASHFLOW_YEAR5.CASH/POWER(1.05,4)
                       FROM PAL_CASHFLOW_YEAR1 LEFT JOIN PAL_CASHFLOW_YEAR2 ON
PAL_CASHFLOW_YEAR1.ID = PAL_CASHFLOW_YEAR2.ID
                                               LEFT JOIN PAL_CASHFLOW_YEAR3 ON
PAL_CASHFLOW_YEAR1.ID = PAL_CASHFLOW_YEAR3.ID
                                               LEFT JOIN PAL_CASHFLOW_YEAR4 ON
PAL_CASHFLOW_YEAR1.ID = PAL_CASHFLOW_YEAR4.ID
                                               LEFT JOIN PAL_CASHFLOW_YEAR5 ON
PAL_CASHFLOW_YEAR1.ID = PAL_CASHFLOW_YEAR5.ID;
SELECT * FROM NPV;
```

The expected result is as follows:

| | NPVALUE |
|---|---|
| 1 | 9,871.556537429577 |
| 2 | 11,281.454962711654 |
| 3 | 12,945.113820765706 |
| 4 | 8,209.398301637899 |
| 5 | 7,218.670123793434 |
| 6 | 7,546.377503215723 |
| 7 | 8,225.713408075022 |
| 8 | 6,733.5291134230565 |
| 9 | 4,408.209364592299 |
| 10 | 5,713.46865453767 |
| 11 | 4,387.7890943143875 |

**Step 3**

Plot the distribution of the net present value of the investment and run Distribution Fitting to fit a normal distribution to the NPV of the investment as. (The Central Limit theorem states that the output distribution will be a normal distribution.)

```
SET SCHEMA DM_PAL;
----------------------------------------
------- distribution fit process ---
----------------------------------------
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO PAL_PARAMETER_TBL VALUES ('DISTRIBUTIONNAME', null, null, 'NORMAL');
INSERT INTO PAL_PARAMETER_TBL VALUES ('OPTIMAL_METHOD', 0, null, null);
DROP TABLE PAL_DISTRFIT_ESTIMATION_TBL;
CREATE COLUMN TABLE PAL_DISTRFIT_ESTIMATION_TBL (
    NAME VARCHAR(50),
    VAL VARCHAR(50)
);
DROP TABLE PAL_DISTRFIT_STATISTICS_TBL;
CREATE COLUMN TABLE PAL_DISTRFIT_STATISTICS_TBL (
    NAME VARCHAR(50),
    VAL DOUBLE
);
CALL "_SYS_AFL"."PAL_DISTRIBUTION_FIT"(NPV, PAL_PARAMETER_TBL,
PAL_DISTRFIT_ESTIMATION_TBL, PAL_DISTRFIT_STATISTICS_TBL) WITH OVERVIEW;
SELECT * FROM PAL_DISTRFIT_ESTIMATION_TBL;
```

The expected result is as follows:

| | NAME | VAL |
|---|---|---|
| 1 | DISTRIBUTIONNAME | NORMAL |
| 2 | MEAN | 8876 |
| 3 | SD | 3219.56 |

**Step 4**

According to the fitted model, run the Cumulative Distribution function to obtain the probability of having an NPV of investment smaller than or equal to a given NPV of the investment.

```
SET SCHEMA DM_PAL;
---------------------------------------
----distribution probability process --
---------------------------------------
DROP TABLE PAL_DISTRPROB_DATA_TBL;
CREATE TABLE PAL_DISTRPROB_DATA_TBL (
    DATACOL DOUBLE
);
INSERT INTO PAL_DISTRPROB_DATA_TBL VALUES (7000);
INSERT INTO PAL_DISTRPROB_DATA_TBL VALUES (8000);
INSERT INTO PAL_DISTRPROB_DATA_TBL VALUES (9000);
INSERT INTO PAL_DISTRPROB_DATA_TBL VALUES (10000);
INSERT INTO PAL_DISTRPROB_DATA_TBL VALUES (11000);
DROP TABLE PAL_DISTRPROB_DISTRPARAM_TBL;
CREATE TABLE PAL_DISTRPROB_DISTRPARAM_TBL (
    NAME VARCHAR(50),
    VALUEE VARCHAR(50)
);
INSERT INTO PAL_DISTRPROB_DISTRPARAM_TBL VALUES ('DISTRIBUTIONNAME', 'Normal');
INSERT INTO PAL_DISTRPROB_DISTRPARAM_TBL VALUES ('MEAN', '100');
INSERT INTO PAL_DISTRPROB_DISTRPARAM_TBL VALUES ('VARIANCE', '1');
UPDATE PAL_DISTRPROB_DISTRPARAM_TBL SET VALUEE = (SELECT VAL FROM
PAL_DISTRFIT_ESTIMATION_TBL WHERE NAME = 'MEAN') WHERE NAME = 'MEAN';
UPDATE PAL_DISTRPROB_DISTRPARAM_TBL SET VALUEE = (SELECT VAL*VAL FROM
PAL_DISTRFIT_ESTIMATION_TBL WHERE NAME = 'SD') WHERE NAME = 'VARIANCE';
SELECT * FROM PAL_DISTRPROB_DISTRPARAM_TBL;
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('LOWER_UPPER',0,null,null);
CALL "_SYS_AFL"."PAL_DISTRIBUTION_FUNCTION"(PAL_DISTRPROB_DATA_TBL,
PAL_DISTRPROB_DISTRPARAM_TBL, #PAL_PARAMETER_TBL, ?);
```

The expected result is as follows:

| | DATACOL | PROBABILITY |
|---|---|---|
| 1 | 7,000 | 0.28005152809569717 |
| 2 | 8,000 | 0.39277767592010426 |
| 3 | 9,000 | 0.5153613572627084 |
| 4 | 10,000 | 0.6364988326970519 |
| 5 | 11,000 | 0.7452824896521799 |

## 8.3 Scenario: Survival Analysis

In clinical trials or community trials, the effect of an intervention is assessed by measuring the number of subjects who have survived or are saved after that intervention over a period of time. We wish to measure the survival probability of Dukes'C colorectal cancer patients after treatment and evaluate statistically whether the patients who accept treatment can survive longer than those who are only controlled conservatively.

## Option 1: Kaplan-Meier Estimate

### Technology Background

Kaplan-Meier estimate is one of the simplest way to measure the fraction of subjects living for a certain amount of time after treatment. The time starting from a defined point to the occurrence of a given event, for example death, is called as survival time.

This scenarios describes a clinical trial of 49 patients for the treatment of Dukes'C colorectal cancer. The following data shows the survival time in 49 patients with Dukes'C colorectal cancer who are randomly assigned to either linoleic acid or control treatment.

| Treatment | Survival Time (months) |
|---|---|
| Linoleic acid (n = 25) | 1+, 5+, 6, 6, 9+, 10, 10, 10+, 12, 12, 12, 12, 12+, 13+, 15+, 16+, 20+, 24, 24+, 27+, 32, 34+, 36+, 36+, 44+ |
| Control (n = 24) | 3+, 6, 6, 6, 6, 8, 8, 12, 12, 12+, 15+, 16+, 18+, 18+, 20, 22+, 24, 28+, 28+, 28+, 30, 30+, 33+, 42 |

The + sign indicates censored data. Until 6 months after treatment, there are no deaths. The effect of the censoring is to remove from the alive group those that are censored. At time 6 months two subjects have been censored so the number alive just before 6 months is 23. There are two deaths at 6 months. Thus,

$\hat{S}(6)$ = `1*(23-2) / 23 = 0.9130`

We now reduce the number alive ("at risk") by two. The censored event at 9 months reduces the "at risk" set to 20. At 10 months there are two deaths. So the proportion surviving is `18/20 = 0.9`, and the cumulative proportion surviving is `0.913*0.90 = 0.8217`.

The data can then be loaded into table as follows:

| TIME | STATUS | OCCURENCES | GROUP |
|---|---|---|---|
| 1 | 0 | 1 | linoleic acid |
| 5 | 0 | 1 | linoleic acid |
| 6 | 1 | 1 | linoleic acid |
| 6 | 1 | 1 | linoleic acid |
| 9 | 0 | 1 | linoleic acid |
| 10 | 1 | 1 | linoleic acid |
| 10 | 1 | 1 | linoleic acid |
| 10 | 0 | 1 | linoleic acid |
| 12 | 1 | 4 | linoleic acid |
| 12 | 0 | 1 | linoleic acid |
| 13 | 0 | 1 | linoleic acid |
| 15 | 0 | 1 | linoleic acid |
| 16 | 0 | 1 | linoleic acid |
| 20 | 0 | 1 | linoleic acid |
| 24 | 1 | 1 | linoleic acid |
| 24 | 0 | 1 | linoleic acid |
| 27 | 0 | 1 | linoleic acid |
| 32 | 1 | 1 | linoleic acid |
| 34 | 0 | 1 | linoleic acid |
| 36 | 0 | 2 | linoleic acid |
| 44 | 0 | 1 | linoleic acid |
| 3 | 0 | 1 | control |
| 6 | 1 | 4 | control |
| 8 | 1 | 2 | control |
| 12 | 1 | 2 | control |
| 12 | 0 | 1 | control |

Then we get the survival estimates as follows:

| | GROUP | TIME | RISKNUM | EVENTSNUM | PROB | STDERR | LOWERLIMIT | UPPERLIMIT |
|---|---|---|---|---|---|---|---|---|
| 1 | control | 6 | 23 | 4 | 0.8260869565217391 | 0.0790341964475118 | 0.6006104112371988 | 0.9309035583871156 |
| 2 | control | 8 | 19 | 2 | 0.7391304347826088 | 0.09156053715170008 | 0.5092094387542188 | 0.8733757816976766 |
| 3 | control | 12 | 17 | 2 | 0.6521739130434783 | 0.09931134621220186 | 0.42347858443917... | 0.8084499771514064 |
| 4 | control | 20 | 10 | 1 | 0.5869565217391305 | 0.10870510355884236 | 0.3488770820094191 | 0.7636977693383366 |
| 5 | control | 24 | 8 | 1 | 0.5135869565217391 | 0.11729212936202411 | 0.2713180524944534 | 0.7115052089560592 |
| 6 | control | 30 | 4 | 1 | 0.3851902173913043 | 0.14178453369559724 | 0.1310407651073283 | 0.6389978236289816 |
| 7 | control | 42 | 1 | 1 | 0 | ? | ? | ? |
| 8 | linole... | 6 | 23 | 2 | 0.9130434782608696 | 0.058753384755841... | 0.6949475375320585 | 0.9775156699007136 |
| 9 | linole... | 10 | 20 | 2 | 0.8217391304347826 | 0.08091665803648351 | 0.5917336879636097 | 0.9291697513009319 |
| 10 | linole... | 12 | 17 | 4 | 0.628388746803069 | 0.10476559644019677 | 0.3911328385442529 | 0.794578585914622 |
| 11 | linole... | 24 | 8 | 1 | 0.5498401534526853 | 0.11748198297014116 | 0.29979455261707... | 0.7430604641784113 |
| 12 | linole... | 32 | 5 | 1 | 0.43987212276214... | 0.13604287804080634 | 0.17941541573926... | 0.6753068933112216 |

To compare survival estimates produced from two groups, we use log-rank test. It is a hypothesis test to compare the survival distribution of two groups (some of the observations may be censored) and is used to test the null hypothesis that there is no difference between the populations (treatment group and control group) in the probability of an event (here a death) at any time point. The methods are nonparametric in that

they do not make assumptions about the distributions of survival estimates. The analysis is based on the times of events (here deaths). For each such time we calculate the observed number of deaths in each group and the number expected if there were in reality no difference between the groups. It is widely used in clinical trials to establish the efficacy of a new treatment in comparison with a control treatment when the measurement is the time to event (such as the time from initial treatment to death).

Because the log-rank test is purely a test of significance, it cannot provide an estimate of the size of the difference between the groups.

| | GROUP | TOTALRISK | OBSERVED | EXPECTED | LOGRANKSTAT |
|---|---|---|---|---|---|
| 1 | control | 24 | 12 | 10.714679152837048 | 0.328949386383633 |
| 2 | linoleic acid | 25 | 10 | 11.285320847162952 | 0.328949386383633 |

| | STATISTICS | VALUE |
|---|---|---|
| 1 | chiSqr | 0.3289493863836331 |
| 2 | df | 1 |
| 3 | p-value | 0.5662783832720855 |

**Implementation Step**

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

Input customer data and use the Kaplan-Meier function to get the survival estimates and log-rank test statistics.

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_TRIAL_DATA_TBL;
CREATE COLUMN TABLE PAL_TRIAL_DATA_TBL (
    "TIME" INTEGER,
    "STATUS" INTEGER,
    "OCCURRENCES" INTEGER,
    "GROUP" VARCHAR(50)
);
INSERT INTO PAL_TRIAL_DATA_TBL VALUES(1, 0, 1, 'linoleic acid');
INSERT INTO PAL_TRIAL_DATA_TBL VALUES(5, 0, 1, 'linoleic acid');
INSERT INTO PAL_TRIAL_DATA_TBL VALUES(6, 1, 1, 'linoleic acid');
INSERT INTO PAL_TRIAL_DATA_TBL VALUES(6, 1, 1, 'linoleic acid');
INSERT INTO PAL_TRIAL_DATA_TBL VALUES(9, 0, 1, 'linoleic acid');
INSERT INTO PAL_TRIAL_DATA_TBL VALUES(10, 1, 1, 'linoleic acid');
INSERT INTO PAL_TRIAL_DATA_TBL VALUES(10, 1, 1, 'linoleic acid');
INSERT INTO PAL_TRIAL_DATA_TBL VALUES(10, 0, 1, 'linoleic acid');
INSERT INTO PAL_TRIAL_DATA_TBL VALUES(12, 1, 4, 'linoleic acid');
INSERT INTO PAL_TRIAL_DATA_TBL VALUES(12, 0, 1, 'linoleic acid');
INSERT INTO PAL_TRIAL_DATA_TBL VALUES(13, 0, 1, 'linoleic acid');
INSERT INTO PAL_TRIAL_DATA_TBL VALUES(15, 0, 1, 'linoleic acid');
INSERT INTO PAL_TRIAL_DATA_TBL VALUES(16, 0, 1, 'linoleic acid');
INSERT INTO PAL_TRIAL_DATA_TBL VALUES(20, 0, 1, 'linoleic acid');
INSERT INTO PAL_TRIAL_DATA_TBL VALUES(24, 1, 1, 'linoleic acid');
INSERT INTO PAL_TRIAL_DATA_TBL VALUES(24, 0, 1, 'linoleic acid');
INSERT INTO PAL_TRIAL_DATA_TBL VALUES(27, 0, 1, 'linoleic acid');
INSERT INTO PAL_TRIAL_DATA_TBL VALUES(32, 1, 1, 'linoleic acid');
INSERT INTO PAL_TRIAL_DATA_TBL VALUES(34, 0, 1, 'linoleic acid');
INSERT INTO PAL_TRIAL_DATA_TBL VALUES(36, 0, 2, 'linoleic acid');
INSERT INTO PAL_TRIAL_DATA_TBL VALUES(44, 0, 1, 'linoleic acid');
```

```
INSERT INTO PAL_TRIAL_DATA_TBL VALUES(3, 0, 1, 'control');
INSERT INTO PAL_TRIAL_DATA_TBL VALUES(6, 1, 4, 'control');
INSERT INTO PAL_TRIAL_DATA_TBL VALUES(8, 1, 2, 'control');
INSERT INTO PAL_TRIAL_DATA_TBL VALUES(12, 1, 2, 'control');
INSERT INTO PAL_TRIAL_DATA_TBL VALUES(12, 0, 1, 'control');
INSERT INTO PAL_TRIAL_DATA_TBL VALUES(15, 0, 1, 'control');
INSERT INTO PAL_TRIAL_DATA_TBL VALUES(16, 0, 1, 'control');
INSERT INTO PAL_TRIAL_DATA_TBL VALUES(18, 0, 2, 'control');
INSERT INTO PAL_TRIAL_DATA_TBL VALUES(20, 1, 1, 'control');
INSERT INTO PAL_TRIAL_DATA_TBL VALUES(22, 0, 1, 'control');
INSERT INTO PAL_TRIAL_DATA_TBL VALUES(24, 1, 1, 'control');
INSERT INTO PAL_TRIAL_DATA_TBL VALUES(28, 0, 3, 'control');
INSERT INTO PAL_TRIAL_DATA_TBL VALUES(30, 0, 1, 'control');
INSERT INTO PAL_TRIAL_DATA_TBL VALUES(30, 1, 1, 'control');
INSERT INTO PAL_TRIAL_DATA_TBL VALUES(33, 0, 1, 'control');
INSERT INTO PAL_TRIAL_DATA_TBL VALUES(42, 1, 1, 'control');
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
DROP TABLE PAL_TRIAL_ESTIMATES_TBL;
CREATE COLUMN TABLE PAL_TRIAL_ESTIMATES_TBL (
    "GROUP" VARCHAR (50),
    "TIME" INTEGER,
    "RISKNUM" INTEGER,
    "EVENTSNUM" INTEGER,
    "PROB" DOUBLE,
    "STDERR" DOUBLE,
    "LOWERLIMIT" DOUBLE,
    "UPPERLIMIT" DOUBLE
);
DROP TABLE PAL_TRIAL_LOGRANK_STAT1_TBL;
CREATE COLUMN TABLE PAL_TRIAL_LOGRANK_STAT1_TBL (
    "GROUP" VARCHAR (50),
    "TOTALRISK" INTEGER,
    "OBSERVED" INTEGER,
    "EXPECTED" DOUBLE,
    "LOGRANKSTAT" DOUBLE
);
DROP TABLE PAL_TRIAL_LOGRANK_STAT2_TBL;
CREATE COLUMN TABLE PAL_TRIAL_LOGRANK_STAT2_TBL (
    "STATISTICS" VARCHAR(20),
    "VALUE" DOUBLE
);
CALL "_SYS_AFL"."PAL_KMSURV"(PAL_TRIAL_DATA_TBL, #PAL_PARAMETER_TBL,
PAL_TRIAL_ESTIMATES_TBL, PAL_TRIAL_LOGRANK_STAT1_TBL,
PAL_TRIAL_LOGRANK_STAT2_TBL) WITH OVERVIEW;
SELECT * FROM PAL_TRIAL_ESTIMATES_TBL;
SELECT * FROM PAL_TRIAL_LOGRANK_STAT1_TBL;
SELECT * FROM PAL_TRIAL_LOGRANK_STAT2_TBL;
```

## Option 2: Weibull Distribution

### Technology Background

Weibull distribution is often used for reliability and survival analysis. It is defined by 3 parameters: shape, scale, and location. Scale works as key to magnify or shrink the curve. Shape is the crucial factor to define how the curve looks like, as described below:

- Shape = 1: The failure rate is constant over time, indicating random failure.
- Shape < 1: The failure rate decreases over time.
- Shape > 1: The failure rate increases over time.

For the same raw data as in the above Kaplan-Meier option, also shown below:

| Treatment | Survival Time (months) |
|---|---|
| Linoleic acid (n = 25) | 1+, 5+, 6, 6, 9+, 10, 10, 10+, 12, 12, 12, 12, 12+, 13+, 15+, 16+, 20+, 24, 24+, 27+, 32, 34+, 36+, 36+, 44+ |
| Control (n = 24) | 3+, 6, 6, 6, 6, 8, 8, 12, 12, 12+, 15+, 16+, 18+, 18+, 20, 22+, 24, 28+, 28+, 28+, 30, 30+, 33+, 42 |

The DISTRFITCENSORED function is used to fit the Weibull distribution on the censored data. For the two types of treatment, linoleic acid and control, two separate calls of DISTRFITCENSORED are performed to get two Weibull distributions.

### Implementation Steps

Assume that:

- `DM_PAL` is a schema belonging to USER1;
- USER1 has been assigned the `AFL__SYS_AFL_AFLPAL_EXECUTE` or `AFL__SYS_AFL_AFLPAL_EXECUTE_WITH_GRANT_OPTION` role.

### Step 1

Get Weibull distribution and statistics from the linoleic acid treatment data:

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_DISTRFITCENSORED_DATA_TBL;
CREATE COLUMN TABLE PAL_DISTRFITCENSORED_DATA_TBL (
    "LEFT" DOUBLE,
    "RIGHT" DOUBLE
);
INSERT INTO PAL_DISTRFITCENSORED_DATA_TBL VALUES (1,NULL);
INSERT INTO PAL_DISTRFITCENSORED_DATA_TBL VALUES (5,NULL);
INSERT INTO PAL_DISTRFITCENSORED_DATA_TBL VALUES (6,6);
INSERT INTO PAL_DISTRFITCENSORED_DATA_TBL VALUES (6,6);
INSERT INTO PAL_DISTRFITCENSORED_DATA_TBL VALUES (9,NULL);
INSERT INTO PAL_DISTRFITCENSORED_DATA_TBL VALUES (10,10);
INSERT INTO PAL_DISTRFITCENSORED_DATA_TBL VALUES (10,10);
INSERT INTO PAL_DISTRFITCENSORED_DATA_TBL VALUES (10,NULL);
INSERT INTO PAL_DISTRFITCENSORED_DATA_TBL VALUES (12,12);
INSERT INTO PAL_DISTRFITCENSORED_DATA_TBL VALUES (12,12);
INSERT INTO PAL_DISTRFITCENSORED_DATA_TBL VALUES (12,12);
INSERT INTO PAL_DISTRFITCENSORED_DATA_TBL VALUES (12,12);
INSERT INTO PAL_DISTRFITCENSORED_DATA_TBL VALUES (12,NULL);
INSERT INTO PAL_DISTRFITCENSORED_DATA_TBL VALUES (13,NULL);
INSERT INTO PAL_DISTRFITCENSORED_DATA_TBL VALUES (15,NULL);
INSERT INTO PAL_DISTRFITCENSORED_DATA_TBL VALUES (16,NULL);
INSERT INTO PAL_DISTRFITCENSORED_DATA_TBL VALUES (20,NULL);
INSERT INTO PAL_DISTRFITCENSORED_DATA_TBL VALUES (24,24);
INSERT INTO PAL_DISTRFITCENSORED_DATA_TBL VALUES (24,NULL);
INSERT INTO PAL_DISTRFITCENSORED_DATA_TBL VALUES (27,NULL);
INSERT INTO PAL_DISTRFITCENSORED_DATA_TBL VALUES (32,32);
INSERT INTO PAL_DISTRFITCENSORED_DATA_TBL VALUES (34,NULL);
INSERT INTO PAL_DISTRFITCENSORED_DATA_TBL VALUES (36,NULL);
INSERT INTO PAL_DISTRFITCENSORED_DATA_TBL VALUES (36,NULL);
```

```
INSERT INTO PAL_DISTRFITCENSORED_DATA_TBL VALUES (44,NULL);
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO PAL_PARAMETER_TBL VALUES ('DISTRIBUTIONNAME', NULL, NULL,'WEIBULL');
INSERT INTO PAL_PARAMETER_TBL VALUES ('OPTIMAL_METHOD',0, NULL, NULL);
DROP TABLE PAL_DISTRFITCENSORED_ESTIMATION_TBL;
CREATE COLUMN TABLE PAL_DISTRFITCENSORED_ESTIMATION_TBL (
    "NAME" VARCHAR(50),
    "VALUE" VARCHAR(50)
);
DROP TABLE PAL_DISTRFITCENSORED_STATISTICS_TBL;
CREATE COLUMN TABLE PAL_DISTRFITCENSORED_STATISTICS_TBL (
    "NAME" VARCHAR(50),
    "VALUE" DOUBLE
);
CALL "_SYS_AFL"."PAL_DISTRIBUTION_FIT_CENSORED"(
    PAL_DISTRFITCENSORED_DATA_TBL, PAL_PARAMETER_TBL,
PAL_DISTRFITCENSORED_ESTIMATION_TBL, PAL_DISTRFITCENSORED_STATISTICS_TBL) WITH
OVERVIEW;
SELECT * FROM PAL_DISTRFITCENSORED_ESTIMATION_TBL;
SELECT * FROM PAL_DISTRFITCENSORED_STATISTICS_TBL;
```

The expected results are as follows:

| | NAME | VALUE |
|---|---|---|
| 1 | DISTRIBUTIONNAME | WEIBULL |
| 2 | SCALE | 36.3069 |
| 3 | SHAPE | 1.40528 |

| | NAME | VALUE |
|---|---|---|
| 1 | LOGLIKELIHOOD | -47.035827236974946 |

### Step 2

Get Weibull distribution and statistics from the control treatment data:

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_DISTRFITCENSORED_DATA_TBL;
CREATE COLUMN TABLE PAL_DISTRFITCENSORED_DATA_TBL (
    "LEFT" DOUBLE,
    "RIGHT" DOUBLE
);
INSERT INTO PAL_DISTRFITCENSORED_DATA_TBL VALUES (3,NULL);
INSERT INTO PAL_DISTRFITCENSORED_DATA_TBL VALUES (6,6);
INSERT INTO PAL_DISTRFITCENSORED_DATA_TBL VALUES (6,6);
INSERT INTO PAL_DISTRFITCENSORED_DATA_TBL VALUES (6,6);
INSERT INTO PAL_DISTRFITCENSORED_DATA_TBL VALUES (6,6);
INSERT INTO PAL_DISTRFITCENSORED_DATA_TBL VALUES (8,8);
INSERT INTO PAL_DISTRFITCENSORED_DATA_TBL VALUES (8,8);
INSERT INTO PAL_DISTRFITCENSORED_DATA_TBL VALUES (12,12);
INSERT INTO PAL_DISTRFITCENSORED_DATA_TBL VALUES (12,12);
INSERT INTO PAL_DISTRFITCENSORED_DATA_TBL VALUES (12,NULL);
INSERT INTO PAL_DISTRFITCENSORED_DATA_TBL VALUES (15,NULL);
INSERT INTO PAL_DISTRFITCENSORED_DATA_TBL VALUES (16,NULL);
INSERT INTO PAL_DISTRFITCENSORED_DATA_TBL VALUES (18,NULL);
INSERT INTO PAL_DISTRFITCENSORED_DATA_TBL VALUES (18,NULL);
INSERT INTO PAL_DISTRFITCENSORED_DATA_TBL VALUES (20,20);
INSERT INTO PAL_DISTRFITCENSORED_DATA_TBL VALUES (22,NULL);
```

```
INSERT INTO PAL_DISTRFITCENSORED_DATA_TBL VALUES (24,24);
INSERT INTO PAL_DISTRFITCENSORED_DATA_TBL VALUES (28,NULL);
INSERT INTO PAL_DISTRFITCENSORED_DATA_TBL VALUES (28,NULL);
INSERT INTO PAL_DISTRFITCENSORED_DATA_TBL VALUES (28,NULL);
INSERT INTO PAL_DISTRFITCENSORED_DATA_TBL VALUES (30,30);
INSERT INTO PAL_DISTRFITCENSORED_DATA_TBL VALUES (30,NULL);
INSERT INTO PAL_DISTRFITCENSORED_DATA_TBL VALUES (33,NULL);
INSERT INTO PAL_DISTRFITCENSORED_DATA_TBL VALUES (42,42);
DROP TABLE PAL_PARAMETER_TBL;
CREATE COLUMN TABLE PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO PAL_PARAMETER_TBL VALUES ('DISTRIBUTIONNAME', NULL, NULL,'WEIBULL');
INSERT INTO PAL_PARAMETER_TBL VALUES ('OPTIMAL_METHOD',0, NULL, NULL);
DROP TABLE PAL_DISTRFITCENSORED_ESTIMATION_TBL;
CREATE COLUMN TABLE PAL_DISTRFITCENSORED_ESTIMATION_TBL (
    "NAME" VARCHAR(50),
    "VALUE" VARCHAR(50)
);
DROP TABLE PAL_DISTRFITCENSORED_STATISTICS_TBL;
CREATE COLUMN TABLE PAL_DISTRFITCENSORED_STATISTICS_TBL (
    "NAME" VARCHAR(50),
    "VALUE" DOUBLE
);
CALL "_SYS_AFL"."PAL_DISTRIBUTION_FIT_CENSORED"(
    PAL_DISTRFITCENSORED_DATA_TBL, PAL_PARAMETER_TBL,
PAL_DISTRFITCENSORED_ESTIMATION_TBL, PAL_DISTRFITCENSORED_STATISTICS_TBL) WITH
OVERVIEW;
SELECT * FROM PAL_DISTRFITCENSORED_ESTIMATION_TBL;
SELECT * FROM PAL_DISTRFITCENSORED_STATISTICS_TBL;
```

The expected results are as follows:

| | NAME | VALUE |
|---|---|---|
| 1 | DISTRIBUTIONNAME | WEIBULL |
| 2 | SCALE | 31.2021 |
| 3 | SHAPE | 1.43217 |

| | NAME | VALUE |
|---|---|---|
| 1 | LOGLIKELIHOOD | -54.02222287154294 |

The results show that the shape values for both treatments are greater than 1, indicating the failure rate increases over time.

### Step 3

Get the CDF (cumulative distribution function) of Weibull distribution for the linoleic acid treatment data:

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_DISTRPROB_DATA_TBL;
CREATE COLUMN TABLE PAL_DISTRPROB_DATA_TBL (
    "DATACOL" DOUBLE
);
INSERT INTO PAL_DISTRPROB_DATA_TBL VALUES (6);
INSERT INTO PAL_DISTRPROB_DATA_TBL VALUES (8);
INSERT INTO PAL_DISTRPROB_DATA_TBL VALUES (12);
INSERT INTO PAL_DISTRPROB_DATA_TBL VALUES (20);
INSERT INTO PAL_DISTRPROB_DATA_TBL VALUES (24);
```

```
INSERT INTO PAL_DISTRPROB_DATA_TBL VALUES (30);
INSERT INTO PAL_DISTRPROB_DATA_TBL VALUES (42);
DROP TABLE PAL_DISTRPROB_DISTRPARAM_TBL;
CREATE COLUMN TABLE PAL_DISTRPROB_DISTRPARAM_TBL (
    "NAME" VARCHAR(50),
    "VALUE" VARCHAR(50)
);
INSERT INTO PAL_DISTRPROB_DISTRPARAM_TBL VALUES ('DistributionName', 'Weibull');
INSERT INTO PAL_DISTRPROB_DISTRPARAM_TBL VALUES ('Shape', '1.40528');
INSERT INTO PAL_DISTRPROB_DISTRPARAM_TBL VALUES ('Scale', '36.3069');
DROP TABLE #PAL_PARAMETER_TBL;
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('LOWER_UPPER',0,null,null);
DROP TABLE PAL_DISTRPROB_RESULT_TBL;
CREATE COLUMN TABLE PAL_DISTRPROB_RESULT_TBL (
    "INPUTDATA" DOUBLE,
    "PROBABILITY" DOUBLE
);
CALL "_SYS_AFL"."PAL_DISTRIBUTION_FUNCTION"(PAL_DISTRPROB_DATA_TBL,
PAL_DISTRPROB_DISTRPARAM_TBL, #PAL_PARAMETER_TBL, PAL_DISTRPROB_RESULT_TBL) WITH
OVERVIEW;
SELECT * FROM PAL_DISTRPROB_RESULT_TBL;
```

The expected result is as follows:

| | INPUTDATA | PROBABILITY |
|---|---|---|
| 1 | 6 | 0.07657965127936783 |
| 2 | 8 | 0.1125151447665862 |
| 3 | 12 | 0.19024473441441703 |
| 4 | 20 | 0.35118218925555456 |
| 5 | 24 | 0.42818254577905457 |
| 6 | 30 | 0.5345725815649545 |
| 7 | 42 | 0.7068736406026395 |

### Step 4

Get the CDF (cumulative distribution function) of Weibull distribution for the control treatment data:

```
SET SCHEMA DM_PAL;
DROP TABLE PAL_DISTRPROB_DATA_TBL;
CREATE COLUMN TABLE PAL_DISTRPROB_DATA_TBL (
    "DATACOL" DOUBLE
);
INSERT INTO PAL_DISTRPROB_DATA_TBL VALUES (6);
INSERT INTO PAL_DISTRPROB_DATA_TBL VALUES (10);
INSERT INTO PAL_DISTRPROB_DATA_TBL VALUES (12);
INSERT INTO PAL_DISTRPROB_DATA_TBL VALUES (24);
INSERT INTO PAL_DISTRPROB_DATA_TBL VALUES (32);
DROP TABLE PAL_DISTRPROB_DISTRPARAM_TBL;
CREATE COLUMN TABLE PAL_DISTRPROB_DISTRPARAM_TBL (
    "NAME" VARCHAR(50),
    "VALUE" VARCHAR(50)
);
INSERT INTO PAL_DISTRPROB_DISTRPARAM_TBL VALUES ('DistributionName', 'Weibull');
INSERT INTO PAL_DISTRPROB_DISTRPARAM_TBL VALUES ('Shape', '1.71902');
INSERT INTO PAL_DISTRPROB_DISTRPARAM_TBL VALUES ('Scale', '20.444');
DROP TABLE #PAL_PARAMETER_TBL;
```

```
CREATE LOCAL TEMPORARY COLUMN TABLE #PAL_PARAMETER_TBL (
    "PARAM_NAME" VARCHAR (256),
    "INT_VALUE" INTEGER,
    "DOUBLE_VALUE" DOUBLE,
    "STRING_VALUE" VARCHAR (1000)
);
INSERT INTO #PAL_PARAMETER_TBL VALUES ('LOWER_UPPER',0,null,null);
DROP TABLE PAL_DISTRPROB_RESULT_TBL;
CREATE COLUMN TABLE PAL_DISTRPROB_RESULT_TBL (
    "INPUTDATA" DOUBLE,
    "PROBABILITY" DOUBLE
);
CALL "_SYS_AFL"."PAL_DISTRIBUTION_FUNCTION"(PAL_DISTRPROB_DATA_TBL,
PAL_DISTRPROB_DISTRPARAM_TBL, #PAL_PARAMETER_TBL, PAL_DISTRPROB_RESULT_TBL) WITH
OVERVIEW;
SELECT * FROM PAL_DISTRPROB_RESULT_TBL;
```

The expected result is as follows:

| | INPUTDATA | PROBABILITY |
|---|---|---|
| 1 | 6 | 0.11445660910656985 |
| 2 | 10 | 0.25360736088540314 |
| 3 | 12 | 0.3297943140085646 |
| 4 | 24 | 0.7321725929430624 |
| 5 | 32 | 0.8846979463928679 |

## Related Information

Kaplan-Meier Survival Analysis [page 1262]
Distribution Fitting [page 1233]

# 9 Best Practices

- Create an SQL view for the input table if the table structure does not meet what is specified in this guide.
- Avoid NULL values in the input data. You can replace the NULL values with the default values via an SQL statement (SQL view or SQL update) because PAL functions cannot infer the default values.
- Create the parameter table as a local temporary table to avoid table name conflicts.
- If you do not use PMML export, you do not need to create a PMML output table to store the result. Just set the `PMML_EXPORT` parameter to 0 and pass ? or `NULL` to the function.
- When using the `KMEANS` function, different `INIT_TYPE` and `NORMALIZATION` settings may produce different results. You may need to try a few combinations of these two parameters to get the best result.
- When using the `APRIORIRULE` function, in some circumstances the rules set can be huge. To avoid an extra long runtime, you can set the `MAXITEMLENGTH` parameter to a smaller number, such as 2 or 3.

# 10 Important Disclaimer for Features in SAP HANA

For information about the capabilities available for your license and installation scenario, refer to the Feature Scope Description for SAP HANA.

# Important Disclaimers and Legal Information

## Hyperlinks

Some links are classified by an icon and/or a mouseover text. These links provide additional information.
About the icons:

- Links with the icon : You are entering a Web site that is not hosted by SAP. By using such links, you agree (unless expressly stated otherwise in your agreements with SAP) to this:
  - The content of the linked-to site is not SAP documentation. You may not infer any product claims against SAP based on this information.
  - SAP does not agree or disagree with the content on the linked-to site, nor does SAP warrant the availability and correctness. SAP shall not be liable for any damages caused by the use of such content unless damages have been caused by SAP's gross negligence or willful misconduct.
- Links with the icon : You are leaving the documentation for that particular SAP product or service and are entering an SAP-hosted Web site. By using such links, you agree that (unless expressly stated otherwise in your agreements with SAP) you may not infer any product claims against SAP based on this information.

## Videos Hosted on External Platforms

Some videos may point to third-party video hosting platforms. SAP cannot guarantee the future availability of videos stored on these platforms. Furthermore, any advertisements or other content hosted on these platforms (for example, suggested videos or by navigating to other videos hosted on the same site), are not within the control or responsibility of SAP.

## Beta and Other Experimental Features

Experimental features are not part of the officially delivered scope that SAP guarantees for future releases. This means that experimental features may be changed by SAP at any time for any reason without notice. Experimental features are not for productive use. You may not demonstrate, test, examine, evaluate or otherwise use the experimental features in a live operating environment or with data that has not been sufficiently backed up.
The purpose of experimental features is to get feedback early on, allowing customers and partners to influence the future product accordingly. By providing your feedback (e.g. in the SAP Community), you accept that intellectual property rights of the contributions or derivative works shall remain the exclusive property of SAP.

## Example Code

Any software coding and/or code snippets are examples. They are not for productive use. The example code is only intended to better explain and visualize the syntax and phrasing rules. SAP does not warrant the correctness and completeness of the example code. SAP shall not be liable for errors or damages caused by the use of example code unless damages have been caused by SAP's gross negligence or willful misconduct.

## Bias-Free Language

SAP supports a culture of diversity and inclusion. Whenever possible, we use unbiased language in our documentation to refer to people of all cultures, ethnicities, genders, and abilities.

www.sap.com/contactsap

© 2026 SAP SE or an SAP affiliate company. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP SE or an SAP affiliate company. The information contained herein may be changed without prior notice.

Some software products marketed by SAP SE and its distributors contain proprietary software components of other software vendors. National product specifications may vary.

These materials are provided by SAP SE or an SAP affiliate company for informational purposes only, without representation or warranty of any kind, and SAP or its affiliated companies shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP or SAP affiliate company products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

SAP and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP SE (or an SAP affiliate company) in Germany and other countries. All other product and service names mentioned are the trademarks of their respective companies.

Please see https://www.sap.com/about/legal/trademark.html for additional trademark information and notices.