PUBLIC
SAP HANA Platform 2.0 SPS 03
Document Version: 1.1 – 2018-10-31

SAP HANA External Machine Learning Library (EML)

THE BEST RUN SAP

Content

1 SAP HANA External Machine Learning Library

This guide describes how to integrate an external machine learning framework, Google TensorFlow, with SAP HANA.

The integration of Google TensorFlow with SAP HANA is based on the SAP HANA Application Function Library (AFL) as well as Google's gRPC remote procedure call package. It also involves a separate server process that hosts the actual machine learning functionality.

2 Getting Started with EML

Familiarize yourself with the External Machine Learning Library (EML) and its specific requirements.

Related Information

2.1 Prerequisites

To use the SAP HANA EML features, you must:

- Install SAP HANA Platform 2.0 SPS 03.
- Install the Application Function Library (AFL), which includes the External Machine Learning Library (EML). For information about how to install or update the AFL, see the section on installing or updating SAP HANA components in the *SAP HANA Server Installation and Update Guide*.

  > **i** Note
  >
  > The revision number of the AFL must match the revision number of SAP HANA. For more information, see SAP Note 1898497.

- Enable the Script Server on the SAP HANA instance. For more information, see SAP Note 1650957.

Related Information

SAP Note 1898497
SAP Note 1650957

2.2 Components

The main components involved in the integration between SAP HANA and Google TensorFlow are described below.

TensorFlow

TensorFlow provides a set of open-source machine-learning libraries together with a supporting infrastructure that allows you to experiment with a wide range of machine learning algorithms. You can execute the algorithms efficiently across different types of hardware environments, ranging from mobile phones to distributed clusters of servers, with additional GPUs.

As a data scientist, you can use TensorFlow to create, train, and evaluate machine learning models. When you are satisfied with the performance of the finalized model, you can prepare additional metadata and export the model in the SavedModel format for consumption through TensorFlow Serving.

For more information, see the TensorFlow documentation at https://www.tensorflow.org/ .

The following TensorFlow versions are supported:

- Minimum version: 1.0
- Maximum and recommended version: 1.6

TensorFlow ModelServer (TMS)

TensorFlow Serving is a flexible, high-performance serving system for machine learning models and is designed for production environments. A TensorFlow ModelServer (TMS) makes TensorFlow models exported in SavedModel format accessible for execution through the gRPC remote procedure call mechanism. An effective SAP HANA EML AFL installation needs to have access to at least one running TMS in order to execute TensorFlow calculations.

For more information, see the general documentation about TensorFlow Serving at https://www.tensorflow.org/serving/ .

The following TensorFlow Serving versions are supported:

- Minimum version: 0.5.1
- Maximum and recommended version: 1.5

Prebuilt binaries (not officially supported) with the transport layer security enhancements can be downloaded from: https://drive.google.com/drive/folders/0B_HaefMdDVd8QXZkY2tIcFpuUVU

The source code for the modifications to the Google TMS (not officially supported) can be found at: https://github.com/bknl/serving

Official Google binaries (without security features) are available at: https://www.tensorflow.org/serving/setup#available_binaries

External Machine Learning (EML) AFL

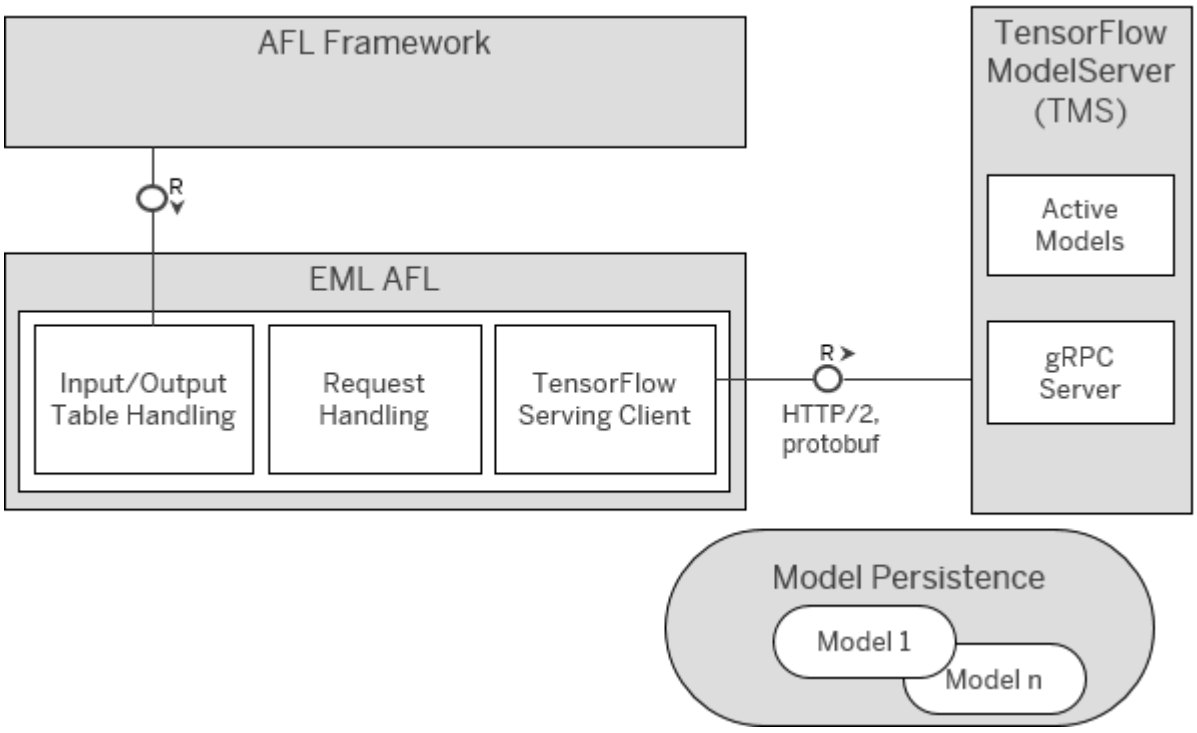
The EML AFL is a TensorFlow Serving client implementation for SAP HANA. It makes TensorFlow inferences on models deployed in a TensorFlow ModelServer available as local AFL procedure calls in SAP HANA.

Supported gRPC versions: 1.0.0 or higher

The gRPC version currently used in EML is 1.6.2.

2.3 Technical Architecture

The basic architecture underlying the integration of Google TensorFlow with SAP HANA is shown below.



Component	Description
AFL Framework	Allows predefined TensorFlow models to be remotely invoked through gRPC calls encapsulated inside AFL procedures
EML AFL	The TensorFlow Serving client implementation for SAP HANA
TensorFlow ModelServer (TMS)	Makes TensorFlow models exported in SavedModel format accessible for execution through gRPC remote procedure calls
Active Models	The models currently served and therefore available for execution
gRPC Server	The gRPC server interface for communication with the TMS client
Model Persistence	The models persisted in SavedModel format in a given TMS

2.4 Deployment Scenarios

The TMS can be co-deployed on the SAP HANA landscape or deployed on separate hardware resources.

TMS Co-deployed on the SAP HANA Landscape

This is the simplest setup since there is no additional hardware required. This type of setup is supported for test, development, and other non-production deployments. For additional security and robustness, the TMS instances should be configured to run in an appropriately configured container on the host machines.

TMS Deployed on Separate Hardware Resources

For production environments, TMS resources must be kept separate from SAP HANA.

2.5 Users and Roles

You require the following users in the SAP HANA and TMS systems.

SAP HANA EML Administrator

An EML administrator manages the system landscape by setting up and monitoring the connections to the TMS servers hosting the models.

An administrator user requires the following SAP HANA privileges:

- `MONITORING`
- `AFL__SYS_AFL_EML_EXECUTE`
- `CREATE REMOTE SOURCE`
- `SELECT ON _SYS_AFL.EML_MODEL_CONFIGURATION`
- `UPDATE ON _SYS_AFL.EML_MODEL_CONFIGURATION`
- `INSERT ON _SYS_AFL.EML_MODEL_CONFIGURATION`
- `DELETE ON _SYS_AFL.EML_MODEL_CONFIGURATION`

SAP HANA EML User

An EML user, typically the application developer, defines which models are available and how they can be invoked through appropriately typed wrapper procedures.

An EML user requires the following SAP HANA privileges:

- `AFL__SYS_AFL_EML_EXECUTE`
- `AFLPM_CREATOR_ERASER_EXECUTE`

Note that the `AFLPM_CREATOR_ERASER_EXECUTE` privilege is only needed to create wrappers. If wrappers are created by another user, an EML user only requires `AFL__SYS_AFL_EML_EXECUTE` to use the EML functionality.

TensorFlow ModelServer Owner

This is the operating system user that runs the TMS instances. There can be multiple operating system users on all target landscapes. For best isolation, this user should be distinct from the SAP HANA system administrator user.

2.6 Checking the EML Installation

To confirm that the EML functions were installed successfully, you can check the following 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 = 'EML';
SELECT * FROM "SYS"."AFL_PACKAGES" WHERE AREA_NAME = 'EML';
SELECT * FROM "SYS"."AFL_FUNCTIONS" WHERE AREA_NAME = 'EML';
```

The result will tell you whether the EML functions were successfully installed on your system. For example:

SQL | Result

```
SELECT FUNCTION_NAME, FUNCTION_TYPE, TECHNICAL_CATEGORY, SCHEMA_NAME, AREA_NAME, PACKAGE_NAME FROM
"SYS"."AFL_FUNCTIONS" WHERE AREA_NAME = 'EML' ORDER BY FUNCTION_NAME
```

	FUNCTION_NAME	FUNCTION_TYPE	TECHNICAL_CATEGORY	SCHEMA_NAME	AREA_NAME	PACKAGE_NAME
1	CHECKDESTINATION	LFunc	var_none	_SYS_AFL	EML	EMLLIB
2	CTL	LFunc	var_none	_SYS_AFL	EML	EMLLIB
3	PREDICT	LFunc	var_columns	_SYS_AFL	EML	EMLLIB
4	PREDICTM	LFunc	var_columns	_SYS_AFL	EML	EMLLIB
5	PREDICTM__OVERLOAD_3_1	LFunc	var_columns	_SYS_AFL	EML	EMLLIB
6	PREDICTM__OVERLOAD_4_1	LFunc	var_columns	_SYS_AFL	EML	EMLLIB
7	PREDICTM__OVERLOAD_5_1	LFunc	var_columns	_SYS_AFL	EML	EMLLIB
8	PREDICT__OVERLOAD_3_1	LFunc	var_columns	_SYS_AFL	EML	EMLLIB
9	PREDICT__OVERLOAD_4_1	LFunc	var_columns	_SYS_AFL	EML	EMLLIB
10	PREDICT__OVERLOAD_5_1	LFunc	var_columns	_SYS_AFL	EML	EMLLIB

2.7 Calling EML Functions

The CTL and CHECKDESTINATION functions can be called directly once the EML library has been installed and the script server has started.

The calling syntax is as follows:

```
CALL <schema_name>.AFL.EML_<function_name>_PROC(
    {inputTab1,…}, <output_tab>) WITH OVERVIEW;
```

Sample Code

```
CALL _SYS_AFL.EML_CHECKDESTINATION_PROC(CALL_PARAMS, RESULTS) WITH OVERVIEW;
```

The PREDICT and PREDICTM functions take a variable number of input tables and therefore require you to generate a procedure that wraps the function. The syntax used to generate and call procedures is described below.

Step 1: Generate an EML Procedure

Any user granted the `AFLPM_CREATOR_ERASER_EXECUTE` role can generate an AFLLANG procedure for a specific EML function. The syntax is shown below:

```
CALL SYS.AFLLANG_WRAPPER_PROCEDURE_CREATE ('<area_name>', '<function_name>',
'<schema_name>', '<procedure_name>', <signature_table>);
```

- `<area_name>`: Always set to `EML`.
- `<function_name>`: The EML built-in function name.
- `<schema_name>`: The name of the schema where the procedure should be located.
- `<procedure_name>`: A name for the EML procedure. This can be anything you want.
- `<signature_table>`: A user-defined table variable. The table contains records to describe the position, schema name, table type name, and parameter type, as defined below:

  ```
  (
  POSITION        int,
  SCHEMA_NAME     nvarchar(256),
  TYPE_NAME       nvarchar(256),
  PARAMETER_TYPE  varchar(7)
  )
  ```

  A typical table variable references a table with the following definition:

Position	Schema Name	Table Type Name	Parameter Type
1	<schema_name>	`<INPUT table type>`	IN
2	<schema_name>	…	IN
3	<schema_name>	`<INPUT table type>`	IN

Position	Schema Name	Table Type Name	Parameter Type
4	<schema_name>	<OUTPUT table type>	OUT

Note the following:

- The records in the signature table must be in the following order: first input table types, then output table types.
- The signature table must be created before generating the EML procedure. The table type names are user-defined.
- To generate a procedure again, drop the existing procedure first by calling `SYS.AFLLANG_WRAPPER_PROCEDURE_DROP`. The syntax is as follows:
  `CALL SYS.AFLLANG_WRAPPER_PROCEDURE_DROP('<schema_name>','<procedure_name>');`

Step 2: Call an EML Procedure

After generating an EML procedure, any user that has the `AFL__SYS_AFL_EML_EXECUTE` or `AFL__SYS_AFL_EML_EXECUTE_WITH_GRANT_OPTION` role can call the procedure, using the syntax below:

```
CALL <schema_name>.<procedure_name>(
{input_table1,…}, <output_table>) with overview;
```

- `<schema_name>`: The name of the schema where the procedure is located.
- `<procedure_name>`: The procedure name specified when generating the procedure in step 1.
- `<input_table1,…>`: The user-defined names of the procedure's input tables.
- `<output_table>`: The user-defined name of the procedure's output table.

Related Information

SAP Note 2046767

3 Models and Call Signatures

A TensorFlow machine learning model implements the actual machine learning functionality (typically inference), which involves mapping a set of input data to a set of output data (for example, images to textual descriptions, historical data to predictions). The actual artifact representing such a model in the TensorFlow Serving environment is a *SavedModel*.

TensorFlow models are created and trained by data scientists, typically in a Python-based programming and training environment. The inner workings of the finished model are hidden within a SavedModel, while the details of the inputs and outputs for using the model are exposed in the SavedModel *signature*.

All inputs and outputs of TensorFlow models are described in terms of *tensors*, which denote arrays of various data types and dimensionality, referred to as the tensor's *shape*. Every TensorFlow model takes a number of input tensors and transforms them into a number of output tensors. Since TensorFlow models are general dataflow graphs, there may be multiple different inputs and outputs possible that make available different parts of the overall transformation flow encapsulated in the model.

A SavedModel has a number of signatures, each describing one meaningful combination of input and output tensors. Each SavedModel should have exactly one `serving_default` signature and can have an arbitrary number of additional named signatures that provide alternative uses of the same graph.

The External Machinge Learning AFL supports only a subset of TensorFlow's tensor shapes and data types, namely those that can be relatively easily mapped between standard SQL tables and data types and tensors. Of the possible combinations defined in the TensorFlow documentation *Tensor Ranks, Shapes, and Types*, the following are supported in particular:

TensorFlow Data Types	HANA Data Types	Supported Shapes
DT_FLOAT, DT_DOUBLE	FLOAT, DOUBLE	Rank 2
DT_STRING	All string and BLOB data types	Rank 1 and Rank 2
DT_INT32, DT_INT64	INTEGER	Rank 2

Related Information

SavedModel README
Module: tf.saved_model
Module: tf.saved_model.signature_def_utils
Tensor Ranks, Shapes, and Types

3.1 TensorFlow Model Example

This example shows a simple linear regression TensorFlow model that tries to fit a vector of input values to a predict regressed output. The model has an input tensor named `features` and an output tensor named `results`.

The signature is as follows (based on the `saved_model_cli` utility from TensorFlow r1.2.0 and higher):

```
% saved_model_cli show --dir linear_regression --all
MetaGraphDef with tag-set: 'serve' contains the following SignatureDefs:
signature_def['serving_default']:
The given SavedModel SignatureDef contains the following input(s):
inputs['features'] tensor_info:
    dtype: DT_FLOAT
    shape: (-1, 1)
    name: x1:0
The given SavedModel SignatureDef contains the following output(s):
outputs['results'] tensor_info:
    dtype: DT_FLOAT
    shape: (-1, 1)
    name: y:0
Method name is: tensorflow/serving/predict
```

The shapes (`features` and `results` are tensors of width 1 and indefinite length, indicated by the -1 value) fit SAP HANA tables of the following shape:

```
CREATE TYPE FEATURES_INPUT_TT AS TABLE ("X" FLOAT);
CREATE TYPE RESULTS_OUTPUT_TT AS TABLE ("Y" FLOAT);
```

Related Information

SavedModel CLI (Command-Line Interface)

3.2 Preparing SavedModels

It is important to include complete signature information and, in particular, complete tensor shape information using the utilities provided by TensorFlow (for example, tf.saved_model.utils.build_tensor_info).

If you have multiple input and output tensors, they are mapped to tables and columns in SAP HANA in the order established by sorting their `key` values in the C locale. Although this might appear arbitrary, it needs to be done because TensorFlow represents the tensors in a position-independent manner in a map keyed by name, while in SAP HANA some of the relationships have to be represented by the relative position of the tables and columns in the argument order. Note that tensor names and SAP HANA column and table names are otherwise completely independent of each other.

The following example illustrates the mapping between tensor names and argument order. On the TensorFlow side, the tensors are listed as follows:

```
signature_def['serving_default']:
The given SavedModel SignatureDef contains the following input(s):
inputs['second'] tensor_info:
    dtype: DT_FLOAT
    shape: (-1, 1)
    name: x1:0
inputs['first'] tensor_info:
    dtype: DT_FLOAT
    shape: (-1, 1)
    name: x2:0
The given SavedModel SignatureDef contains the following output(s):
outputs['y'] tensor_info:
    dtype: DT_FLOAT
    shape: (-1, 1)
    name: y:0
```

On the SAP HANA side, the "x2" tensor maps to the first input table and the "x1" tensor to the second input table, despite being listed in the order shown above. This is because their `key` values sort in the order `first -> second`.

Unlike input tensors, which are mapped one by one to separate input tables in the sorted order of their keys, output tensors have a very limited set of types and shapes and are currently all combined into a single output table. This means that output tensors must all have the same shape and can differ only in their data type.

Related Information

tf.saved_model.utils.build_tensor_info
Serving a TensorFlow Model

3.3 Wrapping and Calling Models: PREDICT

Once you have the information about the number, type, and positions of the model inputs and outputs at hand, you can create a wrapper that exposes the model through the External Machine Learning AFL to the rest of SAP HANA. This is done by using SYS.AFLLANG_WRAPPER_PROCEDURE_CREATE to wrap the PREDICT call provided by the EML AFL.

The following example wraps the simple linear regression model that is specified as follows:

```
The given SavedModel SignatureDef contains the following input(s):
inputs['features'] tensor_info:
    dtype: DT_FLOAT
    shape: (-1, 1)
    name: x1:0
The given SavedModel SignatureDef contains the following output(s):
outputs['results'] tensor_info:
    dtype: DT_FLOAT
    shape: (-1, 1)
    name: y:0
```

1. Prepare the tables:

```
-- Define table types for linear regression
CREATE TYPE CALL_PARAMS_TT AS TABLE ("Parameter" VARCHAR(100), "Value"
VARCHAR(100));
CREATE TYPE FEATURES_TT AS TABLE ("X" FLOAT);
CREATE TYPE RESULTS_TT AS TABLE ("Y" FLOAT);
-- Create description table for procedure creation
CREATE COLUMN TABLE AFLLANG_PROC_PARAM_TABLE (
    POSITION INTEGER,
    SCHEMA_NAME NVARCHAR(256),
    TYPE_NAME NVARCHAR(256),
    PARAMETER_TYPE VARCHAR(7)
);
```

2. Create the wrapper procedure:

```
-- Create procedure for linear regression call
CALL SYS.AFLLANG_WRAPPER_PROCEDURE_DROP('THEUSER', 'LINEAR_REGRESSION');
TRUNCATE TABLE AFLLANG_PROC_PARAM_TABLE;
INSERT INTO AFLLANG_PROC_PARAM_TABLE VALUES (1, 'THEUSER', 'CALL_PARAMS_TT',
'in');
INSERT INTO AFLLANG_PROC_PARAM_TABLE VALUES (2, 'THEUSER', 'FEATURES_TT',
'in');
INSERT INTO AFLLANG_PROC_PARAM_TABLE VALUES (3, 'THEUSER', 'RESULTS_TT',
'out');
CALL SYS.AFLLANG_WRAPPER_PROCEDURE_CREATE('EML', 'PREDICT', 'THEUSER',
'LINEAR_REGRESSION', AFLLANG_PROC_PARAM_TABLE);
```

3. Invoke the TensorFlow model, in the same way as a normal AFL function:

```
-- The value for "Model" must match the name used by the TensorFlow
ModelServer.
CREATE TABLE CALL_PARAMS LIKE CALL_PARAMS_TT;
INSERT INTO CALL_PARAMS VALUES ('Model', 'linearregression');
CREATE TABLE FEATURES LIKE FEATURES_TT;
INSERT INTO FEATURES VALUES (1.1);
INSERT INTO FEATURES VALUES (2.5);
INSERT INTO FEATURES VALUES (2.9);
INSERT INTO FEATURES VALUES (8.9);
CREATE TABLE RESULTS LIKE RESULTS_TT;
-- Perform linear regression
CALL LINEAR_REGRESSION(CALL_PARAMS, FEATURES, RESULTS) WITH OVERVIEW;
SELECT * FROM RESULTS;
```

Related Information

PREDICT Parameters [page 14]

3.4 PREDICT Parameters

The general form of PREDICT is as follows:

```
PREDICT(CALL_PARAMS,INPUT1, INPUT2, … , INPUTY, OUTPUT)
```

The EML AFL currently supports up to four INPUT tables, each of which can have an arbitrary number of columns of FLOAT, DOUBLE, INTEGER, and various SAP HANA string and BLOB data types.

CALL_PARAMS

CALL_PARAMS is a table of the following type:

```
CREATE CALL_PARAMS AS TABLE ("Parameter" VARCHAR(100), "Value"
VARCHAR(100));
```

It can contain the following:

Parameter	Value	Mandatory	Comment
Model	Name of the model as deployed in a TMS, optionally qualified with a named signature	Yes	Must map to at least one running TMS to execute
Deadline	String representing an integer number	No	Maximum number of milliseconds to wait at most for a result

The syntax for accessing named signatures is `<model>%<signaturename>`, so to access a named signature `second_sig` on a model `samplemodel`, you would specify **`samplemodel%second_sig`** as the value. Specifying just **`samplemodel`** is equivalent to specifying **`samplemodel%serving_default`**.

> **i Note**
>
> The keywords for the `Parameter` column (that is, "Model" and "Deadline") are case sensitive.

INPUT

INPUT denotes the tables that match the input tensors of the specified signature of the model according to the following mapping rules, in the order given by sorting the input tensor `keys` in the C locale:

Tensor Shape	Tensor Type	Table
(-1) or (N)	DT_STRING	Single column of HANA string type
(-1,M) or (N,M)	DT_FLOAT DT_DOUBLE DT_STRING DT_INT32 DT_INT64	M columns of a matching HANA data type, at least N rows or an arbitrary number of rows when N == -1

For PREDICT and any definite value of N (that is, not "-1", which means arbitrary), PREDICT ignores rows beyond N. For PREDICTM, the number of rows under each tag has to match N exactly.

OUTPUT

The OUTPUT table has N rows of M columns matching M output tensors of shape (-1,N) and appropriate data types.

3.5 PREDICTM

The PREDICTM function enables multiple inferences to be processed within a single AFL call.

To distinguish the different sets of inputs and results, each input table has an INTEGER "tag" column as its first column. Spans of adjacent rows with the same INTEGER tag value are formatted into a separate inference call. The results are tagged with the corresponding INTEGER tag in the output table's "tag" column. The results in the output table may appear in a different order than the input ranges, but they are guaranteed to be in consecutive rows per partial result set. Where possible, calls are processed in parallel.

The general form of PREDICTM is as follows:

```
PREDICTM(CALL_PARAMS,INPUT1, INPUT2, … , INPUTY, OUTPUT)
```

The semantics of CALL_PARAMS and OUTPUT are identical to PREDICT, with the exception of the first additional "tag" column. So a parallel version involving multiple calls to the linear regression from the PREDICT section looks like this (reusing CALL_PARAMS from the PREDICT example):

```
 -- Create tagged feature table
CREATE TYPE MULTI_FEATURES_TT AS TABLE ("Tag" INTEGER, "X" DOUBLE);
CREATE TABLE MULTI_FEATURES LIKE MULTI_FEATURES_TT;
INSERT INTO MULTI_FEATURES VALUES (1,1.1);
INSERT INTO MULTI_FEATURES VALUES (1,2.5);
INSERT INTO MULTI_FEATURES VALUES (2,1.9);
INSERT INTO MULTI_FEATURES VALUES (2,3.3);
INSERT INTO MULTI_FEATURES VALUES (2,5.9);
INSERT INTO MULTI_FEATURES VALUES (2,8.9);
```

```
-- Create tagged result table
CREATE TYPE MULTI_RESULTS_TT AS TABLE ("Tag" INTEGER,"Y" DOUBLE);
CREATE TABLE MULTI_RESULTS LIKE MULTI_RESULTS_TT;
```

```
-- Create procedure for linear regression call
CALL SYS.AFLLANG_WRAPPER_PROCEDURE_DROP('EML_USER',
'MULTI_LINEAR_REGRESSION_PROC');
DELETE FROM AFLLANG_PROC_PARAM_TABLE;
INSERT INTO AFLLANG_PROC_PARAM_TABLE VALUES (1, 'EML_USER', 'CALL_PARAMS_TT',
'in');
INSERT INTO AFLLANG_PROC_PARAM_TABLE VALUES (2, 'EML_USER', 'MULTI_FEATURES_TT',
'in');
INSERT INTO AFLLANG_PROC_PARAM_TABLE VALUES (3, 'EML_USER', 'MULTI_RESULTS_TT',
'out');
CALL SYS.AFLLANG_WRAPPER_PROCEDURE_CREATE('EML', 'PREDICTM', 'EML_USER',
'MULTI_LINEAR_REGRESSION_PROC', AFLLANG_PROC_PARAM_TABLE);
```

```
-- Perform multiple linear regressions
CALL MULTI_LINEAR_REGRESSION_PROC(CALL_PARAMS, MULTI_FEATURES, MULTI_RESULTS)
WITH OVERVIEW;
SELECT * FROM MULTI_RESULTS;
```

> **i Note**
>
> In case of unrecoverable errors (either input formatting, remote processing, or result collection) in any given tag subset, PREDICTM may partially fill the output table with results prior to the unrecoverable error.

4 Deploying and Mapping Models on a Landscape

TensorFlow models you want to use need to be mapped to a system landscape that has at least one TMS hosting the models.

This task is performed by the system administrator and requires the appropriate roles and privileges. The networking-related part of this mapping makes use of the remote source concept from the SAP HANA federation framework, while the content part uses a system configuration table. All tasks are managed through normal SQL commands on the configuration table and the CTL and CHECKDESTINATION procedures.

Related Information

Deploying SavedModels in a TMS [page 18]
Creating Remote Sources [page 19]
Mapping Models onto Remote Sources [page 21]
Updating Model and Landscape Definitions [page 22]
Validating Model and Landscape Definitions [page 23]

4.1 Deploying SavedModels in a TMS

A SavedModel is a directory tree containing a defined structure of special files. For the purposes of deployment, the saved model directory tree has to be copied to a file system location that can be reached by the running TMS instance.

In order to use a model, you need to expose the model as a service using TensorFlow serving. You can load multiple models into a TMS at the same time by using the `model_config_file` method, which allows you to list the individual models that you want to add to the TMS in a configuration file.

Example: Serve the two example models "testmodel" and "anothermodel" on a single TMS

1. Ensure that you have the following model directory structure:
   - The SavedModel for `testmodel` is in the directory `/models/testmodel_1`.
   - The SavedModel for `anothermodel` is in the directory `/models/testmodel_2`.
2. Create a file called `mymodelconfig.conf` and add the models as follows:

```
model_config_list: {
    config: {
        name: "testmodel",
        base_path: "/models/testmodel_1",
        model_platform: "tensorflow"
    },
```

```
    config: {
        name: "anothermodel",
        base_path: "/models/testmodel_2",
        model_platform: "tensorflow"
    }
}
```

3. On the TMS host, invoke the `tensorflow_model_server` with the following command:

```
tensorflow_model_server --port=5000  --model_config_file=mymodelconfig.conf
```

Note that the file paths do not have to be absolute. Relative file paths are interpreted relative to the working directory of the TMS.
For the full set of arguments for the `tensorflow_model_server`, see the Google documentation.

4.2 Creating Remote Sources

To contact a remote TMS, SAP HANA needs network addresses, port numbers, and optionally security-related information about the network transport to be established. Because Tensorflow Serving is a protocol defined on top of gRPC, remote sources need to be defined on the `grpc` adapter.

> ### Example
>
> A TMS is deployed on a host reachable under the name `tmsserver.mydomain.com`, port number 5000. The model `testmodel` is deployed to that TMS. The connection is unauthenticated and unencrypted. The only transport setup currently understood by the TMS code as published by Google is as follows:
>
> ```
> CREATE REMOTE SOURCE TMS_1
>     ADAPTER "grpc"
>     CONFIGURATION 'server=tmsserver.mydomain.com;port=5000';
> ```

The full set of options for the `grpc` adapter is given below:

Property	Property Description	Values	Comment
server	Server	DNS name or IP address	
port	Port	Port number	
deadline	How long to wait at most for a response from the TMS	Number of milliseconds between 1 and 600000	Maximum timeout is 10 minutes
ssl_mode	SSL mode	"use SSL" or "no SSL"	
client_crt	Certificate	X509 client certificate	Only needed when the TMS is set to certificate authentication
client_key	Private key	X509 client private key	Only needed when the TMS is set to certificate authentication

Property	Property Description	Values	Comment
ca	Certification Authority	X509 Certification Authority certificate	Only needed if any of the keys on the client or server are not in the default CA trust list
proxyhost	HTTP CONNECT capable proxy server	DNS name and port	Example: `host.domain.com:8080`
proxyuser	Optional proxy credentials	Proxy user and optional proxy password	Example: proxyuser proxyuser:proxypassword

Note that it is not possible to enter the correct values for the *client_crt*, *client_key*, and *ca* fields in any of the graphical interfaces to SAP HANA, because the certificates are either binary or have a defined multiline encoding. The only way to create a remote source with these values is through a textual definition.

Example

Create a remote source to a TMS on the host `tmshost` behind the proxy `proxy.corp`:

```
CREATE REMOTE SOURCE PROXYTMS
    ADAPTER "grpc"
    CONFIGURATION 'server=tmshost;port=50052;proxyhost=proxy.corp:8888';
```

Example

Create a remote source with encryption enabled. The TMS server keys have been signed with a private Certification Authority that is not in the default trust list and therefore needs to be explicitly specified:

```
CREATE REMOTE SOURCE SSL_SOURCE ADAPTER "grpc" CONFIGURATION
'server=tmsserver.cloud.com;port=50055;use_ssl=use SSL;ca=-----BEGIN
CERTIFICATE-----
MIIFiTCCA3GgAwIBAgIJAO/+/ow1bMrnMA0GCSqGSIb3DQEBCwUAMFsxCzAJBgNV
BAYTAkRFMQswCQYDVQQIDAJCVzERMA8GA1UEBwwIV2FsbGRvcmYxDDAKBgNVBAoM
A1NBUDENMAsGA1UECwwESEFOQTEPMA0GA1UEAwwGRU1MIENBMB4XDTE3MDYxMjEw
MjUxNFoXDTE4MDYxMjEwMjUxNFowWzELMAkGA1UEBhMCREUxCzAJBgNVBAgMAkJX
MREwDwYDVQQHDAhXYWxsZG9yZjEMMAoGA1UECgwDU0FQMQ0wCwYDVQQLDARIQU5B
MQ8wDQYDVQQDDAZFTUwgQ0EwggIiMA0GCSqGSIb3DQEBAQUAA4ICDwAwggIKAoIC
AQChdgXRf+5TtYx5YGG6HGBFDMpwMAikaUqgK8K50mXxQhwR4aWX5C4BAdi0JZEZ
krqTD4HrMc98g7moY0gq0szaP90iVtTSF+7zm83rRPjKBm+KXu0vGXaaOlKRlWiY
xcAlO2ZRCcEpCYt9cC7megh+8SCzLHtO5q1mBRyny9l7dIg4+Ia4J8ZPjdwacllG
zm3aGbUqABpvDjcUm0TSOtkA1mpza5wj1w8bE7iT+tfxDp8UQg7eQodEPyVVEe/B
6sG8J5mppC/dyrAbgHKKe+zRtgdegTu/a4gpYOYn3Z3zgRDLbZTZjurIF/DQW44+
MahjHeQ7rBUlX7npSkjzBK4VuQ0avYYyJALeiCv6oQoRkZgzUEaEJCt9iEl878BB
2sLtS2eTuT7kUFTuhHs3vxOd10h0KCbr3HDTSoszo+UtTuBwoehHzP21yny2NEpe
65W4gxfI1us1OQh2Li02vhEdiiLdr1yVvkUTlmyYCPf3dAm4QFm2CLUiBUbLU4oZ
cmzuk53FHTaZLHbJYG1gE82sIsxqwZQY8zkQkMXoeNNtkKQYe1vWueZ5RjfLXD+t
0iva+ROgFowM9H8DX4bm/mXfBM5iAc/WVm1Wl/+oKYzlGXWrGTYSu29nduxEjiN9
BSEiI7y0KdoExBzxFE9PB2bPYOrL/Jw7vqBrtI76WnfBkQIDAQABo1AwTjAdBgNV
HQ4EFgQUd1+EF+e7TNZ2Ws2uMpbwwXMDDe4wHwYDVR0jBBgwFoAUd1+EF+e7TNZ2
Ws2uMpbwwXMDDe4wDAYDVR0TBAUwAwEB/zANBgkqhkiG9w0BAQsFAAOCAgEAZ+SU
tJLZsI/K1u94VuVawOqE2hXOw/Jh4G9A/hKTQLFuZjz9Vpgb6zNJSRp+6qimGA6X
5lbh5nBvDJfUEXNAbtrtkqCqWc4Wz3I2DfFoExcsTkrGh1rYx1I259Q7OfpmEjFK
wKr/jCfo4fWPUJAmWMGZrIjIdNFxpynoIYtrL1reo6HS/ewKXi8YVrA2W5Qbcm82
3pP0bXOlkcDBDaJMLWtNW3YoTRNlbTZF4ZAu6rsTuJg3sNq7GZVHZZJVRVV2YllM
INV7X4Zgevy26qQI6/CsvbRfGLMW1KTBef1Rz5poYbhRDZD9M35xy9FdnZ6lNqUN
sDFgJWhwpMPUVhLXuIyzsoHpnC6DLPf5hAJB+pqy0xMZsG0zq5aetK9nDgK4nERr
a8s4meKqIYgBrdkJdMG/BaUFAMpQ+xYchABjUvzHzX4rbEmmkMFIUaThJyVTkgNf
```

```
NEzKCZNRk2wdNArGg1XQOlj1TDddC9dNbo9iRHgtz22Zvli5B40h+Cve3XkqV2ID
9Xc1Yhxh0Jni79lhJmMpxCGMbI+83sXxNWKR79tllEVaFhTAWgc3fbOj+RIG7pow
oedqgR62A4FohogcU/uYPg+ERj7Qb5KHBp80NhlnC4oCUJ4rezy9ubm8ZwzqdmQJ
Cu6G5boq1winTtX3gxJVr7HduIdSzIfVqf8reQ0=
-----END CERTIFICATE-----';
```

4.3 Mapping Models onto Remote Sources

With the TMS contact points defined, you can enter the mapping specifying which models are deployed where in the _SYS_AFL.EML_MODEL_CONFIGURATION table.

The table has the following definition:

```
TABLE _SYS_AFL.EML_MODEL_CONFIGURATION (
    "Model" VARCHAR(256),
    "Parameter" VARCHAR(256),
    "Value" VARCHAR(256)
);
```

Model

The name of the model as deployed in a given TMS. Model names must be unique across the entire landscape and the underlying TensorFlow model must be the same model on all TMS instances where it is deployed, otherwise the behavior is undefined.

Parameter and Value

An open extensible set of parameters that can be assigned to a given model. Currently the only valid "Parameter" is **`RemoteSource`** , with a "Value" specifying the name of a `grpc` remote source, as defined by the SAP HANA federation framework.

> **i Note**
>
> The keywords for the `Parameter` column (that is, `RemoteSource`) are case sensitive.

> **Example**
>
> The TMS is deployed on a host that can be reached through the `TMS_1` remote source already defined in the previous example. The model `testmodel` is deployed to that TMS. Insert the required values into the EML model configuration table:
>
> ```
> INSERT INTO _SYS_AFL.EML_MODEL_CONFIGURATION VALUES('testmodel',
> 'RemoteSource', 'TMS_1');
> ```

A single TMS can host multiple models and a single model can be hosted on multiple TMS instances. Typically, it is better to deploy multiple models on a single TMS on a given host rather than deploying multiple TMS instances on a single host with a single model in each TMS. In the latter case, this occupies multiple ports and needlessly replicates the overhead of the underlying TensorFlow runtime in each TMS.

4.4 Updating Model and Landscape Definitions

In typical steady-state operation, SAP HANA reads the contents of the _SYS_AFL.EML_MODEL_CONFIGURATION table during system startup and routes all inference calls to the deployed models according to their state, as defined in SYS_AFL.EML_MODEL_CONFIGURATION, at that point in time.

Any changes applied to SYS_AFL.EML_MODEL_CONFIGURATION at runtime (either through changes to the table or changes to the underlying remote source definitions) are not immediately honored by the runtime. As an administrator, you can prepare changes to the definitions with multiple independent statements and then apply the entire set of changes to the EML AFL runtime through an explicit update call as follows:

```
SYS_AFL.EML_CTL_PROC(Request,CALL_PARAMS,RESULTS)
```

Request

Currently the only valid value for `Request` is `UpdateModelConfiguration`.

CALL_PARAMS

`CALL_PARAMS` is a table with the following format:

```
TABLE CALL_PARAMS ("Parameter" VARCHAR(100), "Value"
VARCHAR(100));
```

The contents are ignored for `UpdateModelConfiguration`.

RESULTS

`RESULTS` is a table with the following format:

```
TABLE CTL_RESULTS ("Key" VARCHAR(100), "Value" INTEGER, "Text"
VARCHAR(100));
```

It is used to report the result of whatever action CTL has taken, by giving both the error codes and long texts for the various items of a control request.

The only key for the `UpdateModelConfiguration` request is `Status`, with the error code and error text shown in the *Value* and *Text* columns.

Example

Add a second model to the TMS created in the previous section:

```
INSERT INTO _SYS_AFL.EML_MODEL_CONFIGURATION VALUES('anothermodel',
'RemoteSource', 'TMS_1');
```

Update the runtime in an SAP HANA system with a single index server/script server:

```
CREATE TABLE CALL_PARAMS ("Parameter" VARCHAR(100), "Value" VARCHAR(100));
CREATE TABLE CTL_RESULTS ("Key" VARCHAR(100), "Value" INTEGER, "Text"
VARCHAR(100));
CALL _SYS_AFL.EML_CTL_PROC(''UpdateModelConfiguration'', CALL_PARAMS,
CTL_RESULTS);
```

Update the runtime in an SAP HANA system with multiple index servers/script servers:

```
CREATE TABLE CALL_PARAMS ("Parameter" VARCHAR(100), "Value" VARCHAR(100));
```

```
CREATE TABLE CTL_RESULTS ("Key" VARCHAR(100), "Value" INTEGER, "Text"
VARCHAR(100));
CREATE PROCEDURE UPDATE_MODEL_CONFIGURATION() AS
BEGIN
    DECLARE CURSOR CUR FOR
        SELECT VOLUME_ID FROM SYS.M_VOLUMES WHERE SERVICE_NAME =
'indexserver';
    FOR CUR_ROW AS CUR DO
        EXEC 'CALL _SYS_AFL.EML_CTL_PROC(''UpdateModelConfiguration'',
CALL_PARAMS, CTL_RESULTS)'
            || ' WITH OVERVIEW WITH HINT(ROUTE_TO(' || :CUR_ROW.VOLUME_ID ||
'))';
    END FOR;
END;
CALL UPDATE_MODEL_CONFIGURATION();
```

4.5 Validating Model and Landscape Definitions

The CHECKDESTINATION call allows you to validate that a given model can be actively called under a specified set of landscape definitions. It can be used to verify that the model is both reachable and has a valid serving state.

You can use the CHECKDESTINATION call as follows:

```
CREATE TABLE CALL_PARAMS_TT ("Parameter" VARCHAR(100), "Value" VARCHAR(100));
TABLE RESULTS ("Code" VARCHAR(10), "Longtext" VARCHAR(100));
_SYS_AFL.EML_CHECKDESTINATION_PROC(CALL_PARAMS, RESULTS);
```

CALL_PARAMS

The following is valid for `CALL_PARAMS`:

Parameter	Value	Mandatory	Comment
Model	Name of the model as deployed in a TMS	No	Must map to at least one running TMS to execute. It does not have to be in SYS_AFL.EML_MODEL_CONFIGURATION.
RemoteSource	Name of a remote source	No	Does not have to be in SYS_AFL.EML_MODEL_CONFIGURATION
Deadline	String representing an integer number	No	Maximum number of milliseconds to wait at most for a result, as specified by Remote-Source or 5000 ms as default

The "RemoteSource" parameter allows TMS hosts to be checked that have not yet been deployed into the active runtime service. Actual model invocations through PREDICT-style calls need to have a remote source defined in SYS_AFL.EML_MODEL_CONFIGURATION.

> **Note**
>
> The keywords for the `Parameter` column (that is, "Model", "RemoteSource", and "Deadline") are case sensitive.

> **Example**
>
> Check whether "testmodel" can be reached and is actively deployed on a given remote source:

```
CREATE TABLE CALL_PARAMS ("Parameter" VARCHAR(100), "Value"
VARCHAR(100));
CREATE TABLE RESULTS ("Code" VARCHAR(10), "Longtext"
VARCHAR(100));
INSERT INTO CALL_PARAMS VALUES ('Model', 'testmodel');
CALL _SYS_AFL.EML_CHECKDESTINATION_PROC(CALL_PARAMS,
RESULTS);
```

- If the "Model" parameter is omitted but a "RemoteSource" is explicitly specified, the CHECKDESTINATION function tries to check basic connectivity and credentials on a TMS by trying to invoke a dummy model request. Depending on the settings of the TMS, this may lead to (informational) log messages for non-existent "dummy-model-check" messages by the TMS servers.
- If a "Model" parameter is specified but not a "RemoteSource", the CHECKDESTINATION function uses the currently active mapping to remote sources and checks the status of every TMS the model is listed on.
- If both "Model" and "RemoteSource" are omitted, the CHECKDESTINATION function tries to do a complete check of all models and remote sources specified in SYS_AFL.EML_MODEL_CONFIGURATION. That is, determine what the status of all models and target destinations would be if a CTL call were made to push the contents of SYS_AFL.EML_MODEL_CONFIGURATION into the runtime service.

RESULTS

The return values in the RESULTS table after successful invocation (that is, the call to CHECKDESTINATION was technically correct, but not necessarily the underlying destination being checked) are as follows:

Code	Long Text	Comment
0	Ok	The model was reached at the given destination.
1	Cannot determine destination for model <model>	Either the host is missing or there is no matching model. Check the parameters and SYS_AFL.EML_MODEL_CONFIGURATION.
2	Cannot create channel credentials for model	Check the authorization settings of the remote sources registered for the model
3	Messages describing problems with the remote source definition in the HANA federation framework	The remote source named in _SYS_AFL.EML_MODEL_CONFIGURATION either doesn't exist in SAP HANA or the configuration string is ill-formed

Code	Long Text	Comment
4	Destination <destination> exceeded deadline	The host could not be resolved or reached within the configured timeout (default: 5 seconds). Check the name resolution and IP reachability.
5	Server <destination> ok but no model <model>	A TMS was found but did not have a matching model. Check the TMS configuration and the model name.
14	Server <destination> is alive but refused connection	The host was reached but no TMS or proxy was found at the given port number. Check the proxy or TMS (depending on whether a proxy is used).

5 Important Disclaimer for Features in SAP HANA Platform

For information about the capabilities available for your license and installation scenario, refer to the Feature Scope Description (FSD) for your specific SAP HANA version on the SAP HANA Platform webpage.

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 a 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.

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.

Gender-Related Language

We try not to use gender-specific word forms and formulations. As appropriate for context and readability, SAP may use masculine word forms to refer to all genders.

www.sap.com/contactsap

THE BEST RUN SAP