PUBLIC
2026-04-13

Personalized Recommendation

Content

1 What Is Personalized Recommendation?

Get recommendations based on browsing history, and item description.

Personalized Recommendation is a generic reusable service. It uses state-of-the-art machine learning techniques to give visitors to your website highly personalized recommendations based on their browsing history, and item description. Train and use machine learning models to deliver these recommendations across a wide range of business scenarios.

With Personalized Recommendation you can:

- Elevate user experience and engagement
- Enhance item discovery and conversion
- Retain business control, curate relevancy and meet KPIs (Key Performance Indicators)

> ⚠ Caution
>
> The Personalized Recommendation service was removed from the list of Eligible Cloud Services on March 31, 2026.
>
> Personalized Recommendation will be available until the end of the current subscription term. It won't be available for renewal terms that begin after the removal date. Service maintenance and continuous support are guaranteed until the end of the current subscription term.
>
> Please reach out to your account manager at SAP to discuss the way forward if you're using this service.
>
> As an alternative offering, consider using SAP-RPT-1 from SAP AI Core.

Features

Get next-item recommendations	Get personalized, or next-item recommendations that are more likely to be selected next, based on the user's current context or item interactions. Every user with a unique item interaction history receives a unique set of recommendations.
Get similar-item recommendations	Get alternative, or similar-item recommendations that match the item that is currently being viewed.
Get smart-search results	Get personalized search results based on the user's input and context. Receive traditional text queries, but also contextual information such as item interaction history, as well as nontext queries such as categorical features.
Get user-affinity recommendations	Get affinity recommendations or item-attribute preferences based on the user's past interactions. These recommendations enable you to tailor users' experience to their intrinsic affinities and preferences.

For more details on the service core features, see also Inference Options [page 93].

Environment

This service is available in the Cloud Foundry environment.

Prerequisites

See Initial Setup [page 15].

Technical Constraints

For information on technical limits, see Technical Constraints [page 159].

Regional Availability

Get an overview on the availability of Personalized Recommendation according to region, infrastructure provider, and release status in the SAP Discovery Center.

2 What's New for Personalized Recommendation

Technical Component	Environment	Title	Description	Action	Lifecycle	Type	Line of Business	Modular Business Process	Product	Latest Revision	Available as of
Personalized Recommendation	Cloud Foundry	Service Removal	The Personalized Recommendation service was removed from the list of Eligible Cloud Services on March 31, 2026. Personalized Recommendation will be available until the end of the current subscription term. It won't be available for renewal terms that begin after the removal date. Service maintenance and continuous support are guaranteed until the end of the current subscription term. Please reach out to your account manager at SAP to discuss the way forward if you're using this service. As an alternative offering, consider using SAP-RPT-1 from SAP AI Core.	Info only	Deprecated	Changed	Technology	Not applicable	SAP Business Technology Platform	2026-04-13	2026-03-31

Technical Component	Environment	Title	Description	Action	Lifecycle	Type	Line of Business	Modular Business Process	Product	Latest Revision	Available as of
Personalized Recommendation	Cloud Foundry	Scenario Parameters for Offline Next-Item Recommendations	The following scenario parameters are now available for offline next-item recommendations: • `cached_recommendations_per_page` • `cached_file_size_per_page` See Scenario Parameters for Offline Next-Item Recommendations [page 46].	Info only	General Availability	New	Technology	Not applicable	SAP Business Technology Platform	2025-09-30	2025-07-08
Personalized Recommendation	Cloud Foundry	Offline (Scenario) Recommendations	You can now use the scenario type `propensity` to get recommendations that include users with the highest affinity to a given item. See Offline (Scenario) Propensity Recommendations [page 151] and Scenario Parameters for Offline Propensity Recommendations [page 73].	Info only	General Availability	New	Technology	Not applicable	SAP Business Technology Platform	2025-09-30	2024-07-08
Personalized Recommendation	Cloud Foundry	Overall Improvements	There have been several code, stability, and performance improvements.	Info only	General Availability	Changed	Technology	Not applicable	SAP Business Technology Platform	2025-07-08	2025-07-08
Personalized Recommendation	Cloud Foundry	Overall Improvements	There have been several code, stability, and performance improvements.	Info only	General Availability	Changed	Technology	Not applicable	SAP Business Technology Platform	2025-03-31	2025-03-27

Technical Component	Environment	Title	Description	Action	Lifecycle	Type	Line of Business	Modular Business Process	Product	Latest Revision	Available as of
Personalized Recommendation	Cloud Foundry	Support for X.509 Authentication	The Personalized Recommendation APIs now support X.509 authentication. See Enable X.509 Authentication [page 17].	Info only	General Availability	New	Technology	Not applicable	SAP Business Technology Platform	2025-03-31	2025-03-27

2.1 2024 What's New for Personalized Recommendation (Archive)

Technical Component	Environment	Title	Description	Action	Lifecycle	Type	Line of Business	Modular Business Process	Product	Latest Revision	Available as of
Personalized Recommendation	Cloud Foundry	Overall Improvements	There have been several code and performance improvements.	Info only	General Availability	Changed	Technology	Not applicable	SAP Business Technology Platform	2024-10-01	2024-10-01
Personalized Recommendation	Cloud Foundry	Offline (Scenario) Recommendations	You can now use a model that has already been trained successfully to get offline personalized recommendations without the need to deploy an online serving instance. See Scenario Configurations and Data [page 42] and Offline (Scenario) Recommendations [page 148].	Info only	General Availability	New	Technology	Not applicable	SAP Business Technology Platform	2024-07-18	2024-07-18

2.2 2023 What's New for Personalized Recommendation (Archive)

Technical Component	Environment	Title	Description	Action	Lifecycle	Type	Line of Business	Modular Business Process	Product	Latest Revision	Available as of
Personalized Recommendation	Cloud Foundry	Overall Improvements	There have been several code and stability improvements.	Info only	General Availability	Changed	Technology	Not applicable	SAP Business Technology Platform	2023-12-20	2023-12-20
Personalized Recommendation	Cloud Foundry	Overall Improvements	There have been several code and stability improvements.	Info only	General Availability	Changed	Technology	Not applicable	SAP Business Technology Platform	2023-10-27	2023-10-27
Personalized Recommendation	Cloud Foundry	Complementary Next-Item Recommendations	The complementary next-item recommendation type is now available. It recommends items that are known to be viewed together in a single session. See Next-Item Recommendations [page 93] and Next-Item Recommendation Types [page 108].	Info only	General Availability	New	Technology	Not applicable	SAP Business Technology Platform	2023-07-17	2023-07-17

Technical Component	Environment	Title	Description	Action	Lifecycle	Type	Line of Business	Modular Business Process	Product	Latest Revision	Available as of
Personalized Recommendation	Cloud Foundry	Inference Options - Exclude Past Items and Ratings	The following inference parameters are now available for Next-Item Recommendations [page 93]: • `exclude_past_items` • `ratings` The parameter `ratings` is also available for Smart-Search Results [page 125] and User-Affinity Recommendations [page 136]. Optionally use `exclude_past_items` to exclude items found in the user's clickstream history (`items_ls`) from the recommendation results. Optionally use `ratings` to assign a rating to each item in the input sequence.	Info only	General Availability	New	Technology	Not applicable	SAP Business Technology Platform	2023-07-17	2023-07-17
Personalized Recommendation	Cloud Foundry	SAP Business Accelerator Hub	Personalized Recommendation is now available in the SAP Business Accelerator Hub. See Personalized Recommendation.	Info only	General Availability	New	Technology	Not applicable	SAP Business Technology Platform	2023-07-17	2023-07-17
Personalized Recommendation	Cloud Foundry	Overall Improvements	There have been several code and stability improvements.	Info only	General Availability	Changed	Technology	Not applicable	SAP Business Technology Platform	2023-07-17	2023-07-17

Technical Component	Environment	Title	Description	Action	Lifecycle	Type	Line of Business	Modular Business Process	Product	Latest Revision	Available as of
Personalized Recommendation	Cloud Foundry	Overall Improvements	• The quality of the recommendations for models trained without clickstream data has been improved. • There have been several code and stability improvements.	Info only	General Availability	Changed	Technology	Not applicable	SAP Business Technology Platform	2023-04-20	2023-04-20

2.3 2022 What's New for Personalized Recommendation (Archive)

Technical Component	Environment	Title	Description	Action	Lifecycle	Type	Line of Business	Modular Business Process	Product	Latest Revision	Available as of
Personalized Recommendation	Cloud Foundry	Overall Improvements	There have been several code and stability improvements.	Info only	General Availability	Changed	Technology	Not applicable	SAP Business Technology Platform	2022-11-18	2022-11-18
Personalized Recommendation	Cloud Foundry	Content-based Recommendations to Enhance Onboarding Experience	You can now consume the service when interaction data isn't available, the new content-based approach learns recommendations using item metadata (item catalogue, item attributes, and textual features).	Info only	General Availability	New	Technology	Not applicable	SAP Business Technology Platform	2022-11-18	2022-11-18

Technical Component	Environment	Title	Description	Action	Lifecycle	Type	Line of Business	Modular Business Process	Product	Latest Revision	Available as of
Personalized Recommendation	Cloud Foundry	Overall Improvements	There have been several code and stability improvements.	Info only	General Availability	Changed	Technology	Not applicable	SAP Business Technology Platform	2022-09-08	2022-09-08
Personalized Recommendation	Cloud Foundry	Tutorials	The tutorial mission Use Machine Learning to Get Recommendations Based on Users' Browsing History is now available for the Personalized Recommendation service. See Tutorials [page 19].	Info only	General Availability	New	Technology	Not applicable	SAP Business Technology Platform	2022-09-08	2022-09-08
Personalized Recommendation	Cloud Foundry	ML Explainability	Documentation on ML Explainability [page 152] is now available.	Info only	General Availability	New	Technology	Not applicable	SAP Business Technology Platform	2022-09-08	2022-09-08
Personalized Recommendation	Cloud Foundry	Accessibility Features	Documentation on Accessibility Features in Personalized Recommendation [page 168] is now available.	Info only	General Availability	New	Technology	Not applicable	SAP Business Technology Platform	2022-09-08	2022-09-08
Personalized Recommendation	Cloud Foundry	Overall Improvements	There have been several code and stability improvements.	Info only	General Availability	Changed	Technology	Not applicable	SAP Business Technology Platform	2022-07-07	2022-07-07

3 Concepts

Find out more about key concepts relating to the Personalized Recommendation service.

Term	Definition
attribute boosting	Reorders recommendation results based on boosting configurations.
attribute filtering	Controls recommendation results based on specific filter configurations.
clickstream	Logged interaction data between user and items, for example, clickstream that enables the machine learning model to learn the relations between the items that appear in the same context.
cold start items	New item metadata that weren't part of the item catalogue training data but are freshly provided as part of the inference request payload. The Personalized Recommendation service uses these new items to generate recommendation results in a meaningful way.
cold start users	New user profiles that weren't part of the user metadata training data but are freshly provided as part of the inference request payload. The Personalized Recommendation service uses these new users to generate recommendation results in a meaningful way.
item catalogue	Additional item information in the form of categorical and text features, for example, product catalogue to improve the quality of predictions and enable additional recommendation features.
ML explainability	Machine learning explainability. The ability of a machine learning solution to provide meaningful insight into the inference results, such as scores and weight distributions.
offline recommendations	Offline recommendations are generated during training job executions. With offline recommendations, you can get downloadable recommendations without the need to deploy an online serving instance. In addition, offline recommendations also enable customization of inference parameters (similar to online recommendations), which you can download or receive after the training job is complete.
online recommendations	Online recommendations are generated on demand with a serving instance after a successful training job and model deployment.
user affinity	A list of preferred items based on the user's past interactions.
user metadata	Additional user information in the form of categorical and text features, for example, user profile information to improve the quality of predictions and allow better personalized recommendations.

4 Service Plans

Learn more about the different types of service plans for Personalized Recommendation.

Personalized Recommendation provides different types of service plans. The type you choose determines pricing, conditions of use, resources, available services, and hosts.

It depends on your use case whether you choose a free or a paid service plan. If you plan to use your global account in productive mode, you must purchase a paid enterprise account. It's important that you're aware of the differences when you're planning and setting up your account model. See Initial Setup [page 15].

There are two service plans available:

- Free
- Standard

For more details about the service plans, see the following table:

Service Plan	Details	Account Type
Free	• Service plan intended for development and try-out purposes on your enterprise account. • You can send up to 1000 inference requests per month. See Free Tier Option Technical Constraints [page 160] and the tutorial Get an Account on SAP BTP to Try Out Free Tier Service Plans.	Enterprise
Standard	• Personalized Recommendation default service plan. • Service plan intended for productive usage. See Technical Constraints [page 159].	Enterprise

→ Remember

- If you first activated the Free service plan, you can update the same service instance to switch to Standard.
- Both metadata and transaction data, including trained models, are transferred to the Standard service plan when you switch from Free to Standard.
- If you don't want Free and Standard data to be combined together, you can split them by subscribing to the service plans in separate subaccounts.

5 Initial Setup

Get started with Personalized Recommendation using the standard procedures for SAP BTP Cloud Foundry environment.

> → Tip
>
> See Tutorials [page 19] to find out how to use the free tier option for Personalized Recommendation to try out the service.

Prerequisites

You have set up your global account and at least one subaccount on SAP BTP. For an overview of the required steps, see Getting Started in the Cloud Foundry Environment.

> ⓘ Note
>
> Personalized Recommendation allows you to move subaccounts between your global accounts. For more information, see Relationship Between Global Accounts, Subaccounts, and Directories.

Enabling the Service in the Cloud Foundry Environment

Enable Personalized Recommendation using the standard procedures for SAP BTP Cloud Foundry environment.

Context

> → Tip
>
> You can also use the booster *Set Up Account for Personalized Recommendation* to automate the steps described below on the SAP BTP cockpit. See Boosters and the tutorial Use Free Tier to Set Up Account for Personalized Recommendation and Get Service Key.

Procedure

1. Create a service instance in the Cloud Foundry environment. See Creating Service Instances.

   > ⓘ Note
   >
   > In the *New Instance or Subscription* wizard, enter only the *Basic Info* details, and leave the *Parameters* details empty. The configuration of instance parameters isn't required or supported for this service.

2. You can then bind the service instance to your application, or you can create a service key to communicate directly with the service instance. See Binding Service Instances to Applications and Creating Service Keys.

6 Enable X.509 Authentication

Find out how to enable your service instance for authentication with an X.509 client certificate.

The Personalized Recommendation service supports X.509 authentication with the certificates managed either by the SAP Authorization and Trust Management service or self-managed. The authentication with an X.509 client certificate is enabled for every service instance by default.

Create Service Key or Service Binding Additional Parameters

To use X.509 secrets, you need to set additional parameters when you create your service key or service binding. We support the following two scenarios:

- The SAP Authorization and Trust Management service generates certificates for you. In this case, the parameters you need to set when you create your service key or service binding are the following in `JSON` format:

  ```
  {
     "xsuaa":{
        "credential-type":"x509",
        "x509":{
           "key-length":2048,
           "validity":8,
           "validity-type":"DAYS"
        }
     }
  }
  ```

  For a detailed description of the parameters, see Parameters for X.509 Certificates Managed by SAP Authorization and Trust Management Service.

- You already have your own public key infrastructure (PKI), with certificates issued from one of the trusted Certificate Authorities (CAs). In this case, the parameters you need to set when you create your service key or service binding are the following in `JSON` format::

  ```
  {
     "xsuaa":{
        "credential-type":"x509",
        "x509":{
           "certificate":"-----BEGIN CERTIFICATE-----...-----END
  CERTIFICATE-----",
           "ensure-uniqueness":false,
           "certificate-pinning":true,
           "hide-certificate":true
        }
     }
  }
  ```

  For a detailed description of the parameters, see Parameters for Self-Managed X.509 Certificates. See also Trusted Certificate Authentication.

Get an Authorization Token with X.509 Certificate

To get an authorization token using an X.509 certificate, use “certurl”. In the scenario of already generated certificates, also use “key” and “certificate” from the service key.

Example of a request using `curl`:

```
curl --cert <path to certificate.pem> --key <path to
key.pem> --request POST <value of "uaa.certurl">/oauth/token -d
'grant_type=client_credentials&client_id=<Value of "uaa.clientid">'
```

See also the blog post: X.509 certificate-based authentication(mTLS) – Generating X.509 certificates of BTP managed services.

7 Tutorials

Follow our tutorials to get familiar with the Personalized Recommendation APIs and functionalities.

Tutorial Mission	Description
Use Machine Learning to Get Recommendations Based on Users' Browsing History	Give visitors to your website highly personalized recommendations based on their browsing history and/or item description. Train and use machine learning models to deliver these recommendations across a wide range of business scenarios.

Related Information

Tutorial Navigator

8 API Reference

Explore the Personalized Recommendation APIs.

Before using the Personalized Recommendation APIs listed below, you need to retrieve your OAuth access token as described in Get Access Token [page 20].

- Resource [page 21]
- Training [page 22]
- Inference [page 91]
- Serving [page 156]

To display the comprehensive specification of the Personalized Recommendation API in Swagger UI, add the URL path extension `/doc` to the Personalized Recommendation base URL (that is, the url value from outside the uaa section of your service key).

Related Information

Initial Setup [page 15]
Common Status and Error Codes [page 157]
Technical Constraints [page 159]

8.1 Get Access Token

Retrieve your OAuth access token to access the Personalized Recommendation APIs.

> ⓘ Note
>
> The token is valid for 12 hours. After that, you need to generate a new one.

Request

Base URL: url value from inside the uaa section of the service key

URL Path: `/oauth/token`

HTTP Method: *POST*

Request Headers

Header	Required	Values
`Content-Type`	Yes	<application/x-www-form-urlencoded>

Request Parameters

Parameter	Required	Data Type	Description
`client_id`	Yes	*String*	The clientid value from the service key.
`client_secret`	Yes	*String*	The clientsecret value from the service key.
`grant_type`	Yes	*String*	Token grant type. Set it to `client_credentials`.
`response_type`	Yes	*String*	Token response type. Set it to `token`.

Response

The response is given as a status (200 or 401). See Common Status and Error Codes [page 157].

Response Example

200 “Success”

```
{
    "access_token": "<< your access token >>",
    "token_type": "bearer",
    "expires_in": 43199,
    "scope": "uaa.resource",
    "jti": "8d00c157058949daab714a44c04c416b"
}
```

8.2 Resource

Use the Resource APIs to view and manage subaccount and tenant resources, including the deletion of tenant and site data.

Request

Base URL: url value from outside the uaa section of the service key

URL Path Extension: `/doc`

Resource APIs

HTTP Method	URL Endpoint Path	Description
GET	`/standard/rs/v1/subaccount`	Get the status of all online serving instances that belong to a subaccount.
GET	`/standard/rs/v1/tenants/{tenant}`	Get the file count of any sites that belong to a subaccount and tenant. Use this endpoint to verify tenant and/or site deletion. Observe the following guidelines: • If `site` is provided, the endpoint returns the files for `site` only. • If `site` is not provided, and `all_sites` is set to *true*, the endpoint returns the files for all sites. • If `site` is not provided, and `all_sites` is set to *false*, the endpoint returns the files for `default` site only.
DELETE	`/standard/rs/v1/tenants/{tenant}`	Delete all files for any sites that belong to a subaccount and tenant. Use this endpoint to offboard the Personalized Recommendation service. Observe the following guidelines: • If `site` is provided, the endpoint deletes data for `site` only. • If `site` is not provided, and `all_sites` is set to *true*, the endpoint deletes data for all sites. • If `site` is not provided, and `all_sites` is set to *false*, the endpoint deletes data for `default` site only.

8.3 Training

Use the Training APIs to submit training jobs via file upload or object store integration, and to get status updates for submitted training jobs. Each training job and subsequent trained model belongs to a tenant with

site-support enabled (each site has its own model, allowing a tenant to have multiple models under multiple sites).

Request

Base URL: url value from outside the uaa section of the service key

URL Path Extension: `/doc`

Training APIs

HTTP Method	URL Endpoint Path	Description
POST	`/standard/rs/v1/tenants/{tenant}/jobs`	Submit a training job to the Personalized Recommendation service via object store integration. **Note** The serving of a model instance can be triggered by setting `serve_model` to *true* in the payload.
POST	`/standard/rs/v1/tenants/{tenant}/jobs/file-upload`	Submit a training job to the Personalized Recommendation service via file upload. See Technical Constraints [page 159]. **Note** The serving of a model instance can be triggered by setting `serve_model` to *true* in the payload.

HTTP Method	URL Endpoint Path	Description
GET	`/standard/rs/v1/tenants/{tenant}/jobs/latest`	Get the status of the latest training job that belongs to the specified tenant and site of your subaccount. If training is successful, training details are provided in a metrics payload. If training fails, an appropriate exception message is provided for further assistance in an exception_info payload.
GET	`/standard/rs/v1/tenants/{tenant}/jobs/{jobid}`	Provide the job ID of the training job to get the status of a particular training job that belongs to the specified tenant and site of your subaccount. **Note** A job ID is provided for every training job submitted. If training is successful, training details are provided in a metrics payload. If training fails, an appropriate exception message is provided for further assistance in an exception_info payload.

Related Information

Training Data Requirements [page 24]
Training Job Options [page 79]
Training Job Statuses [page 84]
Trained Model Metrics [page 89]

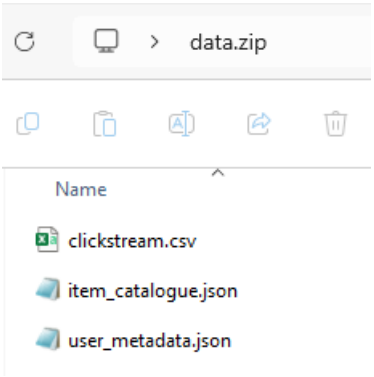
8.3.1 Training Data Requirements

> **Note**
>
> The Personalized Recommendation service has been developed primarily with datasets in English.
>
> The service has also been validated to achieve similar accuracy results with datasets in other languages that use ISO-8859-1 - Latin 1 character-encoding scripts.

The training data requirements have been simplified and standardized to fit any recommendation scenario. The following training data types are supported:

Training Data Type	Required	File Format	Allowed File Names
Clickstream	Yes (if item catalogue data isn't provided)	A CSV file containing the following columns: • `userId` (String) • `itemId` (String) • `timestamp` (Float) See also Clickstream [page 26].	• **`clickstream.csv`**
Item catalogue	Yes (if clickstream data isn't provided)	A JSON file containing item information and metadata, referenced by a unique `itemId`. See also Item Catalogue [page 30].	• **`item_catalogue.json`** • **`catalogue.json`**
User metadata	No (optional, use **only** if clickstream data is provided)	A JSON file containing user profile information and metadata, referenced by a unique `userId`. See also User Metadata [page 36].	• **`user_metadata.json`**
Scenario configurations and data	No (optional, use **only** to get Offline (Scenario) Recommendations [page 148])	A JSON file containing the configurations and data required for the scenario, allowing each scenario to be customizable to your business needs, referenced by the `scenario_type` you want to use (either `next-items` or `similar-items`). See also Scenario Configurations and Data [page 42].	• **`<scenario_name>.scenario.json`**

To train a model, you need to collect all the available training data types applicable to your recommendation scenario or use case, and add them in a compressed file with the following structure:



> → Remember
>
> If you don't have a particular type of training data, **do not** include the respective file in the compressed **`data.zip`** file.
>
> The compressed file naming convention and the file types used for each training data type are name-sensitive. **Only** the following names are allowed:
>
> - **`data.zip`**
> - **`clickstream.csv`**
> - **`item_catalogue.json`**
> - **`catalogue.json`**
> - **`user_metadata.json`**
> - **`<scenario_name>.scenario.json`**

For your reference, see the following samples of training data compressed files:

- data.zip (containing the data types: clickstream, item catalogue, and user metadata)
- data.zip (containing the data types: clickstream and item catalogue)
- data.zip (containing the data types: scenario configurations and data, clickstream, item catalogue, and user metadata)

8.3.1.1 Clickstream

Clickstream data is essential to understand the customer behavior, given by the sequences of historical item interactions for each user. It enables the machine learning model to:

- Learn relations between items that appear in the same context, for example, user session.
- Identify browsing patterns for each user, leading to personalized recommendations, for example, for users with different history interacting with the same item.
- Learn relations between items and their attributes, for example, from consecutive items in the same session that belong to the same category or having a similar name.

> → Remember
>
> Item catalogue must be provided.

- Learn relations between users' attributes and item interaction sequences, for example, users with the same attribute values have different histories of item interactions.

> → Remember
>
> User metadata must be provided.

Data Schema

The following table shows the data schema of clickstream data used for training the model.

CSV File Column	Data Type	Description
`userId`	*String*	*Validity check* Raw user IDs are considered valid if they are nonempty string or float. Float values are converted to string and invalid values are removed during clickstream parsing. *General usage* The raw user ID is primarily used to group the item interactions for each user. If the user metadata is available, the raw user ID is used to link the users' item interactions to their metadata information. Therefore, it is important that exactly the same ID is used in both data sources. *Handling edge cases* If the user metadata is unavailable, the model does not train embeddings for user attributes and users with only one item interaction are discarded (together with their associated event).

CSV File Column	Data Type	Description
`itemId`	*String*	*Validity check* Raw item IDs are considered valid if they are nonempty string or float. Float values (including 0) are converted to string and invalid values are removed during clickstream parsing. *General usage* The raw item ID is used to link the catalogue information with the clickstream occurrences of items. Therefore, it is important that exactly the same ID is used in both data sources. *Handling edge cases* Item IDs that occur only once in the entire clickstream are ignored by the model. This means that the model does not train an embedding for these IDs, but rather treat them as an array of zeros. Note that the item may still have a nonzero embedding if it has a catalogue entry with valid attributes (only the ID contribution to this embedding is null). It is equivalent to the cold start scenario, where an item ID is unknown, but its attribute values are (at least partially) known from the item catalogue.

CSV File Column	Data Type	Description
`timestamp`	*Float*	*Validity check* Timestamp values are considered valid if they are float or numeric string. String values are converted to float and invalid values are removed during clickstream parsing. *General usage* The timestamp is primarily used to sort the item interactions for each user. Therefore, when preparing or converting the data to fit the dataset schema, the most important aspect to retain is the order of the user events. *Handling edge cases* If the timestamp values are identical for multiple items for the same user, the order of item interactions cannot be determined. In this case, a default sorting by raw item ID is used.

Training Data Sample (`clickstream.csv`):

```
userId,       itemId,      timestamp
user-id-enc1, item-id-1,   978300019
user-id-enc1, item-id-2,   978300055
user-id-enc1, item-id-3,   978300055
user-id-enc2, item-id-2,   978300055
user-id-enc2, item-id-4,   978300103
user-id-enc2, item-id-5,   978300172
user-id-enc3, item-id-1,   978300275
user-id-enc3, item-id-5,   978300719
```

Synthetic Clickstream Enhancement

Use the synthetic clickstream enhancement feature to create or supplement clickstream data with synthetic clickstream data to enhance training data when it is insufficient. For that, set the `enhance_clickstream` flag in the training job options. The `enhance_clickstream` flag is set to *true* by default.

Scenario	If `enhance_clickstream` is set to *true*	If `enhance_clickstream` is set to *false*
Clickstream file with enough events (more or equal to 1000 rows)	Training passes.	Training passes.
Clickstream file is not provided	The clickstream enhancement feature generates clickstream data. Training passes.	Training fails.
Clickstream file with not enough events (less than1000 rows)	The clickstream enhancement feature generates clickstream data to supplement submitted clickstream file. Training passes.	Training fails.

8.3.1.2 Item Catalogue

The item catalogue provides additional information on the items, in the form of categorical and text features. It enables the machine learning model to:

- Improve the quality of the predictions: more item information leads to more patterns in the clickstream data to be identified and learned by the model
- Support cold start items: approximate item representations, for example embeddings, based on attribute representations
- Give flexibility during inference: item attributes may be customized (modified attribute values and/or weights), influencing recommendation results
- Enable smart-search: as the model learns to represent words and item attributes, it can take them as inputs to recommend relevant items

During training, the model learns embeddings for all valid item IDs, categorical attribute values, numerical values, and text tokens. Combined according to the catalogue, the embeddings of ID, categorical, and text attributes compose the final item embeddings. As long as some of the values are valid, a nonzero item embedding can be constructed.

> → Tip
>
> Providing the item catalogue almost always improves the recommendation model's performance. Therefore, while not mandatory, it is highly recommended to provide the item catalogue together with the clickstream data to unlock the full potential of the recommendation engine.

If item catalogue is not provided, the following inference options and service features are unavailable:

- Smart-search results
- Custom attribute recommendations
- Support to cold start items
- Feature importance for item attributes (ML Explainability)
- Item attribute contribution (ML Explainability)

Data Schema

The item catalogue is a JSON file with the following structure (see the following table for details on each parameter):

```
{
    <item_id:String>: {
        'unavailable': <unavailable_flag:Boolean>,
        'categoricalFeatures': {
            <categorical_attribute_1:String>: {
                'values': [<cat_value_1:String>, ..., <cat_value_n:String>]
(list of String),
                'weights': [<cat_weight_1:Float>, ..., <cat_weight_n:Float>]
(list of Float, Optional)
            },
            <categorical_attribute_2:String>: { ... }
        },
        'textFeatures': {
            <text_attribute_1:String>: <text_value_1:String>,
            <text_attribute_2:String>: <text_value_2:String>,
            ...
        },
        'numericalFeatures': {
            <numerical_attribute_1:String>: <numerical_value_1:Float>,
            <numerical_attribute_2:String>: <numerical_value_2:Float>,
            ...
        }
    }
}
```

Parameter	Data Type	Required	Description
`item_id`	*String*	Yes	The item ID of an item. This ID is used to map the items in the following areas: • Clickstream data • Next-item recommendations: inference clickstream input • Similar-item recommendations: inference clickstream input • Smart-search results: inference clickstream input • User metadata: excludes unwanted items feature It is important that exactly the same ID is used across data sources. The model trains ID embeddings for valid item IDs. They are used as components in the overall item embedding.
`unavailable`	*Boolean*	No	A flag that specifies if the item can be recommended as part of the recommendation results. The default value is set to *false* (item is available and visible in recommendations).

Parameter	Data Type	Required	Description
`categorical_attribute`	*String*	Yes, only if `categoricalFeatures` key is defined	The name of a categorical attribute. **Note** The schema of the catalogue should be consistent throughout, and all items in the catalogue should contain the same categorical attribute entry. Categorical attributes are used in the following features: • Next-item recommendations: Attribute boosting, Attribute filtering, Cold start items. • Similar-item recommendations: Attribute boosting, Attribute filtering, Cold start items. • Smart-search results: Attribute queries. For example, in the sample data, all items contain the categories and tags categorical attributes.
`categorical_attribute` (values)	List of *String*	Yes, it is part of the definition of a `categorical_attribute`	The categorical values that belong to the particular categorical attribute. The model calculates an embedding for each valid categorical attribute for each item, depending on the associated values and weights. They are used as components in the overall item embedding.

Parameter	Data Type	Required	Description
`categorical_attribute` (weights)	List of *Float*	No	If the weights key exists, the list of valid weights should match the list of valid values (same length). Otherwise, the weights list is removed and precalculated weights (based on the relative frequency of attribute values in the train catalogue) are used instead. The list of weights is used by the model to weigh the embeddings of respective attribute values when calculating a categorical attribute embedding.
`text_attribute`	*String*	Yes, only if `textFeatures` key is defined	The name of a text attribute. **Note** The schema of the catalogue should be consistent throughout, and all items in the catalogue should contain the same text attribute entry. Text attributes are used in the following features: • Next-item recommendations • Similar-item recommendations • Smart-search results: String queries For example, in the sample data, all items contain the `title` text attribute.

Parameter	Data Type	Required	Description
`text_attribute` (`text_value`)	*String*	Yes, it is part of the definition of a `text_attribute`	The text value that belongs to the particular text attribute. The model calculates an embedding for each valid text attribute for each item, depending on its text value. They are used as components in the overall item embedding.
`numerical_attribute`	*String*	Yes, only if `numericalFeatures` key is defined	The name of a numerical attribute. **Note** The schema of the catalogue should be consistent throughout, and all items in the catalogue should contain the same numerical attribute entry. Numerical attributes are used in the following features: • Next-item recommendations: Attribute filtering, Cold start items. • Similar-item recommendations: Attribute filtering, Cold start items. • Smart-search results: String Queries, Attribute Queries.
`numerical_attribute` (`numerical_value`)	*Float*	Yes, it is part of the definition of a `numerical_attribute`	The numerical value that belongs to the particular numerical attribute.

Training Data Sample (`item_catalogue.json`):

```
{
   "item-id-1":{
      "textFeatures":{
         "title":"Black T-shirt",
         "description":"This is a black T-shirt."
      },
      "categoricalFeatures":{
```

```
         "tags":{
            "values":[
               "Unisex",
               "Shirt",
               "Casual"
            ]
         }
      }
   },
   "item-id-2":{
      "textFeatures":{
         "title":"Pencil box",
         "description":"This is a children’s pencil box."
      },
      "categoricalFeatures":{
         "tags":{
            "values":[
               "Stationery",
               "Children"
            ]
         }
      }
   }
}
```

8.3.1.3 User Metadata

The user metadata provides additional information on the users, in the form of categorical and text features. It enables the machine learning model to:

- Improve the quality of the predictions: the model can learn relations between user attributes and their browsing patterns (attributes and sequence of item interactions)
- Further personalize the recommendations: in addition to relying on past item interactions to predict future ones, the model can relate the user profile to certain item preferences
- Support cold start users: approximate user representations, for example embeddings, based on attribute representations
- Personalize smart-search: the user ID can be passed to the smart-search query to find the most relevant items to the query as well as the user profile

> ⚠ Caution
>
> The Personalized Recommendation service doesn't require (sensitive) personal data for training.
>
> Personal data is subject to the data protection laws applicable in specific countries/regions. We therefore recommend that you don't include personal data in your user metadata training data file.
>
> For more information, see Data Protection and Privacy [page 162].

During training, the model learns embeddings for all valid user IDs, categorical attribute values, and text tokens. Combined according to the metadata, the embeddings of ID, categorical, and text attributes compose the final user embeddings. As long as some of the values are valid, a nonzero user embedding can be constructed.

> → Tip
>
> Providing user metadata can improve the recommendation model's performance. Therefore, while not mandatory, it's recommended to provide user metadata together with the clickstream data to unlock the full potential of the recommendation engine.

If user metadata isn't provided, the following inference options and service features are unavailable:

- Support to cold start users
- Feature importance for user attributes (ML Explainability)
- User attribute contribution (ML Explainability)

Data Schema

The format of the user metadata closely follows the item catalogue for simplicity and standardization.

The user metadata is a JSON file with the following structure (see the following table for details on each parameter):

```
{
    "<user_id_1>" (string): {
        "categoricalFeatures": {
            "<categorical_feature_1>" (string): {
                "values": <categorical attribute values> (list of string),
                "weights": <attribute boosting weights> (list of numeric
[Integer | Float], Optional)
            },
            "<categorical_feature_n>": { ... }
        },
        "textFeatures": {
            "<text_feature_1>" (string): <text_feature_value_1> (string),
            "<text_feature_n>": ...
        },
        "numericalFeatures": {
            "<numerical_feature_1>" (string): <numerical_feature_value_1>
(numeric [Integer | Float]),
            "<numerical_feature_n>": ...
        }
        "unwantedItems": <categorical attribute values> (list of string,
Optional)
        "unavailable": <unavailable> (boolean, Optional)
        }
}
```

Parameter	Data Type	Required	Description
`userId`	*String*	Yes	The user ID of a user. This ID is used to map the users found in the following areas: • Clickstream data • Next-item recommendations: inference users' input • Similar-item recommendations: inference users' input • Smart-search results: inference users' input It's important that exactly the same ID is used across data sources. The model trains ID embeddings for valid user IDs. They're used as components in the overall user embedding.
`unavailable`	*Boolean*	No	Specifies that the user isn't available and, as a consequence, offline recommendations aren't generated for this user. The default value is set to *false*, which means the user is available and included in the recommendations. → Remember Use the user metadata `unavailable` parameter for offline recommendations **only**.
`unwantedItems`	List of *String*	No	Use this feature to define a list of items (by item ID) that won't be recommended for the user if the `Exclude Unwanted Items` setting is enabled.

Parameter	Data Type	Required	Description
`categorical_attribute`	*String*	Yes, only if `categoricalFeatures` key is defined	The name of a categorical attribute. **Note** The schema of the user metadata should be consistent throughout, and all users in the user metadata should contain the same categorical attribute entry. Categorical attributes are used in the following features: • Next-item recommendations: enhance results • Similar-item recommendations: enhance results • Smart-search results: enhance results
`categorical_attribute` (values)	List of *String*	Yes, it's part of the definition of a `categorical_attribute`	The categorical values that belong to the particular categorical attribute. The model calculates an embedding for each valid categorical attribute for each user, depending on the associated values and weights. They're used as components in the overall user embedding.

Parameter	Data Type	Required	Description
`categorical_attribute` (weights)	List of *Float*	No	If the weights key exists, the list of valid weights should match the list of valid values (same length). Otherwise, the weights list is removed and precalculated weights (based on the relative frequency of attribute values in the train user metadata) are used instead. The list of weights is used by the model to weigh the embeddings of respective attribute values when calculating a categorical attribute embedding.
`text_attribute`	*String*	Yes, only if `textFeatures` key is defined	The name of a text attribute. **Note** The schema of the user metadata should be consistent throughout, and all users in the user metadata should contain the same text attribute entry. Text attributes are used in the following features: • Next-item recommendations: enhance results • Similar-item recommendations: enhance results • Smart-search results: enhance results
`text_attribute` (`text_value`)	*String*	Yes, it's part of the definition of a `text_attribute`	The text value that belongs to the particular text attribute. The model calculates an embedding for each valid text attribute for each user, depending on its text value. They're used as components in the overall user embedding.

Parameter	Data Type	Required	Description
`numericalFeatures`	*String*	No	Used to denote numerical features within the user-level metadata. The numerical features are encoded within the model and are used to provide additional user-level context for model training and inference.
`numericalFeatures` (`numerical_feature_key`)	*Numeric*	No	The name or identifier of a numerical feature. Each numerical feature has its own corresponding numerical feature value (numeric). **Note** The schema of the catalogue should be consistent throughout and that all users in the user metadata should contain the same numerical feature entries.

Training Data Sample (`user_metadata.json`):

```
{
   "user-id-enc1":{
      "categoricalFeatures":{
         "tags":{
            "values":[
               "VIP Member",
               "Member"
            ]
         }
      }
   },
   "user-id-enc2":{
      "categoricalFeatures":{
         "tags":{
            "values":[
               "Member"
            ]
         }
      }
   }
}
```

8.3.1.4 Scenario Configurations and Data

To submit one or more scenarios during the training job, you need to provide the scenario configurations and data file (or files) along with the training data in the compressed data file during training job submission.

Each scenario is defined by its scenario configurations and data file containing the configurations and data required for the scenario, allowing each scenario to be customizable to your business needs.

Each Personalized Recommendation training job is limited to 4 scenarios.

The scenario configurations and data files are JSON (.json) files with the secondary `.scenario` extension.

Scenario file naming convention: **`<scenario_name>.scenario.json`**, where scenario_name will be used as the identifier for the scenario for this tenant.

<scenario_name> may only contain:

- Alpha-numeric characters
- The only special characters allowed are: underscore "_" and hyphen "-"

Data Schema

The scenario configurations and data is a JSON file with the following structure (see the following table for details on each root key):

```
{
   "scenario_type":"String",
   "config":"Dict",
   "data":"Dict"
}
```

Root Key	Data Type	Required	Description
`scenario_type`	*String*	Yes	Tells the service which scenario to execute. Personalized Recommendation supports the following scenario types: • `next-items` for Offline (Scenario) Next-Item Recommendations [page 149] • `similar-items` for Offline (Scenario) Similar-Item Recommendations [page 150] • `propensity` for Offline (Scenario) Propensity Recommendations [page 151]
`config`	*Dictionary*	No	Contains the configurations for the scenario. For more information, see: • Scenario Parameters for Offline Next-Item Recommendations [page 46] • Scenario Parameters for Offline Similar-Item Recommendations [page 60] • Scenario Parameters for Offline Propensity Recommendations [page 73]

Root Key	Data Type	Required	Description
`data`	*Dictionary*	No	Contains the dataset to be used for the scenario. **→ Remember** The `data` parameter is required if "config": {"only_provided_ids": true}, as this tells the offline recommendations scenario to only use scenario-provided data to generate offline recommendations. For more information, see: • Scenario Parameters for Offline Next-Item Recommendations [page 46] • Scenario Parameters for Offline Similar-Item Recommendations [page 60] • Scenario Parameters for Offline Propensity Recommendations [page 73]

Training Data Samples (`<scenario_name>.scenario.json`):

```
{
   "scenario_type":"next-items",
   "config":{
      "only_provided_ids":true
   },
   "data":{
      "test_user1":null,
      "test_user2":null
   }
}
```

```
{
   "scenario_type":"similar-items",
   "config":{
      "only_provided_ids":true
   },
   "data":{
```

```
      "test_user1":null,
      "test_user2":null
   }
}
```

```
{
   "scenario_type":"propensity",
   "config":{
      "only_provided_ids":true
   },
   "data":{
      "test_item1":null,
      "test_item2":null
   }
}
```

8.3.1.4.1 Scenario Parameters for Offline Next-Item Recommendations

See the details for each `config` and `data` dictionary parameter in the scenario configurations and data JSON file for the `scenario_type` `next-items`.

Configuration Dictionary Parameters

`config` Dictionary Parameter (scenario declarations)	Data Type	Required	Default Value	Description
`only_provided_ids`	*Boolean*	No	**`true`**	Specifies the ID set to be used for generating offline inferences. If set to **`true`** (default), the scenario only uses IDs provided in the `data` root key. This is the default value to prevent accidental generation of recommendations for non-intended IDs. If set to **`false`**, the scenario uses both the IDs provided in the `data` root key, and the IDs that were used for training the inference model. For more information about the IDs you can use for the scenario type `next-items`, see Offline (Scenario) Next-Item Recommendations [page 149].

`config` Dictionary Parameter (scenario declarations)	Data Type	Required	Default Value	Description
`modify_mode`	*String*	No	**`append`**	Specifies the action taken to merge the values you provide for scenario and training data. For offline next-item recommendations, the scenario parameters affected by `modify_mode` include: • `items_ls`: Each users' item-interaction history that is provided as training data, and as scenario data. > → Remember > `modify_mode` is only effective if the `keep_user_history` training job option is set to **`true`**, which allows the model to retain user-specific interaction history to be used in recommendations. • `excluded_items`: Each users' excluded items list that is provided as training data, and as scenario data.

`config` Dictionary Parameter (scenario declarations)	Data Type	Required	Default Value	Description
				→ Remember Excluded items per user are retained regardless of the value of the `keep_user_history` training job option. The Personalized Recommendation service supports the following `modify_mode` values: • **`append`**: To append scenario-provided ID lists to existing model-trained data (for example, Training Data: ['a', 'b'], Scenario Data: ['c', 'd'] = Final: ['a', 'b', 'c', 'd']). • **`replace`**: To replace existing model-trained data with scenario-provided ID lists (for example. Training Data: ['a', 'b'], Scenario Data: ['c', 'd'] = Final: ['c', 'd']).

`config` Dictionary Parameter (scenario declarations)	Data Type	Required	Default Value	Description
`cached_recommendations_per_page`	*Integer*	No	`-1`	Requirement: `cached_recommendations_per_page` > 10,000 (50 MB before compression, 15 MB after gzip compression). The `cached_recommendations_per_page` parameter determines the number of item rows per page of the output files. Printed in the following format, for example: {scenario-name}-pageX.txt.gzip, where X is the number of the page starting from 1. Calculation for `approximate_file_size` = `cached_recommendations_per_page` x `num_bytes_per_item` / compression ratio. Where `average_bytes_per_item` is 560 bytes, compression ratio is 4x for gzip. Single page pagination is used if `cached_recommendations_per_page` and `cached_file_size_per_page` aren't specified.

`config` Dictionary Parameter (scenario declarations)	Data Type	Required	Default Value	Description
`cached_file_size_per_page`	*Integer*	No	`-1`	Requirement: `cached_file_size_per_page` > 50,000,000 (50 MB before compression, 15 MB after gzip compression). The `cached_file_size_per_page` parameter determines the page size corresponding to the set file size per page. Printed in the following format, for example: {scenario-name}-pageX.txt.gzip, where X is the number of the page starting from 1. Calculation for `approximate_file_size` = `cached_file_size_per_page` / compression ratio. Where compression ratio is 4x for gzip. If `cached_file_size_per_page` is > 50,000,000, it takes precedence over `cached_recommendations_per_page`.

`config` Dictionary Parameter (scenario declarations)	Data Type	Required	Default Value	Description
k	*Integer*	No	`10`	Sets the k parameter for the generation of recommendation results. The k parameter specifies the number of recommendations to be returned.
`explain`	*Boolean*	No	`false`	Sets the `explain` parameter for the generation of recommendation results. The `explain` parameter enables ML Explainability [page 152] for all users in the generated recommendation results.
`explain_k`	*Integer*	No	`10`	Sets the `explain_k` parameter for the generation of recommendation results. The `explain_k` parameter specifies the number of explained recommendations to be returned.

`config` Dictionary Parameter (scenario declarations)	Data Type	Required	Default Value	Description
`exclude_past_items`	*Boolean*	No	`false`	Sets the `exclude_past_items` parameter for the generation of recommendation results. The `exclude_past_items` parameter is a boolean flag to exclude items found in the user's clickstream history (`items_ls`) from the recommendation results.
`diverse`	*Boolean*	No	`false`	Returns a diverse range of recommendations.
`only_allow_clickstream_transitions`	*Boolean*	No	`false`	Means that only item-to-item transitions that are found in the clickstream history are allowed. **⚠ Restriction** This parameter is **only** supported by the sequential next-item recommendation type.

`config` Dictionary Parameter (scenario declarations)	Data Type	Required	Default Value	Description
`items_ls`	*List of String*	No	None	Broadcasts the specified value to all users during inference. The final `items_ls` value for the user is used to either replace, or append to, the existing `items_ls` values of users obtained from the training data (if the training job option `keep_user_history` is set to **`true`**) depending on the `modify_mode` flag.
`excluded_items`	*List of String*	No	None	Broadcasts the specified value to all users during inference. The final `excluded_items` value for the user is used to either replace, or append to, the existing `excluded_items` values of users obtained from the training data (if the training job option `keep_user_history` is set to **`true`**) depending on the `modify_mode` flag.

`config` Dictionary Parameter (scenario declarations)	Data Type	Required	Default Value	Description
`boosting_attributes`	*Dictionary*	No	None	Broadcasts the specified value to all users during inference. The final `boosting_attributes` value for the user is then applied to the users' recommendation results. The `boosting_attributes` parameter enables user-level attribute boosting in the generated recommendation results.
`filtering_attributes`	*Dictionary*	No	None	Broadcasts the specified value to all users during inference. The final `filtering_attributes` value for the user is then applied to the users' recommendation results. The `filtering_attributes` parameter enables user-level attribute filtering in the generated recommendation results.

Dataset Dictionary Parameters

`data` Dictionary Parameter (user declarations)	Data Type	Required	Default Value	Description
`items_ls`	*List of String*	No	None	Specifies the value of `items_ls` for the particular user (identified by `user_id`). The final `items_ls` value for the user is used to either replace, or append to, the existing `items_ls` values of users obtained from the training data (if the training job option `keep_user_history` is set to **`true`**) depending on the `modify_mode` flag.
`excluded_items`	*List of String*	No	None	Specifies the value of `excluded_items` for the particular user (identified by `user_id`). The final excluded_items value for the user is used to either replace, or append to, the existing excluded_items values of users obtained from the training data (if the training job option `keep_user_history` is set to **`true`**) depending on the `modify_mode` flag.

`data` Dictionary Parameter (user declarations)	Data Type	Required	Default Value	Description
`boosting_attributes`	*Dictionary*	No	None	Specifies the value of `boosting_attributes` for the particular user (identified by `user_id`). The final `boosting_attributes` value for the user is then applied to the users' recommendation results. The `boosting_attributes` parameter enables user-level attribute boosting in the generated recommendation results.
`filtering_attributes`	*Dictionary*	No	None	Specifies the value of `filtering_attributes` for the particular user (identified by `user_id`). The final `filtering_attributes` value for the user is then applied to the users' recommendation results. The `filtering_attributes` parameter enables user-level attribute filtering in the generated recommendation results.

Only Model Training Data Sample (`<scenario_name>.scenario.json`):

```
{
```

```
   "scenario_type":"next-items",
   "config":{
      "only_provided_ids":false
   }
}
```

Both Model Training Data and Scenario Sample (`<scenario_name>.scenario.json`):

```
{
   "scenario_type":"next-items",
   "config":{
      "only_provided_ids":false
   },
   "data":{
      "test_user1":null,
      "test_user2":null
   }
}
```

Only Scenario Training Data Sample (`<scenario_name>.scenario.json`):

```
{
   "scenario_type":"next-items",
   "config":{
      "only_provided_ids":true
   },
   "data":{
      "test_user1":null,
      "test_user2":null
   }
}
```

Only Broadcast Parameters Training Data Sample (`<scenario_name>.scenario.json`):

```
{
   "scenario_type":"next-items",
   "config":{
      "only_provided_ids":false,
      "k":5,
      "explain":true,
      "excluded_items":[
         "trained_item1"
      ],
      "filtering_attributes":{
         "categoricalFeatures":{
            "genre":{
               "values":[
                  "comedy"
               ]
```

```
            }
         }
      },
      "boosting_attributes":{
         "categoricalFeatures":{
            "tag":{
               "values":[
                  "christmas"
               ],
               "weights":[
                  3.5
               ]
            }
         }
      }
   }
}
```

Both Broadcast Parameters and Scenario Training Data Sample (`<scenario_name>.scenario.json`):

```
{
   "scenario_type":"next-items",
   "config":{
      "only_provided_ids":false,
      "k":5,
      "explain":true,
      "excluded_items":[
         "trained_item1"
      ],
      "filtering_attributes":{
         "categoricalFeatures":{
            "genre":{
               "values":[
                  "comedy"
               ]
            }
         }
      },
      "boosting_attributes":{
         "categoricalFeatures":{
            "tag":{
               "values":[
                  "christmas"
               ],
               "weights":[
                  3.5
               ]
            }
         }
      }
   },
   "data":{
      "trained_user1":{
         "items_ls":[
            "trained_item2"
         ],
         "filtering_attributes":{
            "categoricalFeatures":{
               "genre":{
                  "values":[
                     "comedy",
```

```
                        "children"
                     ]
                  }
               }
            }
         },
         "trained_user2":{
            "excluded_items":[
               "trained_item1",
               "trained_item2"
            ],
            "filtering_attributes":{
               "categoricalFeatures":{
                  "genre":{
                     "values":[
                        "comedy",
                        "children"
                     ]
                  }
               }
            },
            "boosting_attributes":{
               "categoricalFeatures":{
                  "tag":{
                     "values":[
                        "christmas",
                        "sci-fi"
                     ],
                     "weights":[
                        3.5,
                        3.5
                     ]
                  }
               }
            }
         }
      }
}
```

8.3.1.4.2 Scenario Parameters for Offline Similar-Item Recommendations

See the details for each `config` and `data` dictionary parameter in the scenario configurations and data JSON file for the `scenario_type` `similar-items`.

Configuration Dictionary Parameters

`config` Dictionary Parameter (scenario declarations)	Data Type	Required	Default Value	Description
`only_provided_ids`	*Boolean*	No	**`true`**	Specifies the ID set to be used for generating offline inferences. If set to **`true`** (default), the scenario only uses IDs provided in the `data` root key. This is the default value to prevent accidental generation of recommendations for non-intended IDs. If set to **`false`**, the scenario uses both the IDs provided in the `data` root key, and the IDs that were used for training the inference model. For more information about the IDs you can use for the scenario type `similar-items`, see Offline (Scenario) Similar-Item Recommendations [page 150].

`config` Dictionary Parameter (scenario declarations)	Data Type	Required	Default Value	Description
`modify_mode`	*String*	No	**`append`**	Specifies the action taken to merge the values you provide for scenario and training data. For offline next-item recommendations, the scenario parameters affected by `modify_mode` include: • `items_ls`: Each users' item-interaction history that is provided as training data, and as scenario data. > → Remember > `modify_mode` is only effective if the `keep_user_history` training job option is set to **`true`**, which allows the model to retain user-specific interaction history to be used in recommendations. • `excluded_items`: Each users' excluded items list that is provided as training data, and as scenario data.

`config` Dictionary Parameter (scenario declarations)	Data Type	Required	Default Value	Description
				→ Remember Excluded items per user are retained regardless of the value of the `keep_user_history` training job option. The Personalized Recommendation service supports the following `modify_mode` values: • **`append`**: To append scenario-provided ID lists to existing model-trained data (for example, Training Data: ['a', 'b'], Scenario Data: ['c', 'd'] = Final: ['a', 'b', 'c', 'd']). • **`replace`**: To replace existing model-trained data with scenario-provided ID lists (for example. Training Data: ['a', 'b'], Scenario Data: ['c', 'd'] = Final: ['c', 'd']).

`config` Dictionary Parameter (scenario declarations)	Data Type	Required	Default Value	Description
k	*Integer*	No	`10`	Sets the k parameter for the generation of recommendation results. The k parameter specifies the number of recommendations to be returned.
`explain`	*Boolean*	No	`false`	Sets the `explain` parameter for the generation of recommendation results. The `explain` parameter enables ML Explainability [page 152] for all users in the generated recommendation results.
`explain_k`	*Integer*	No	`10`	Sets the `explain_k` parameter for the generation of recommendation results. The `explain_k` parameter specifies the number of explained recommendations to be returned.

`config` Dictionary Parameter (scenario declarations)	Data Type	Required	Default Value	Description
`exclude_past_items`	*Boolean*	No	**`false`**	Sets the `exclude_past_items` parameter for the generation of recommendation results. The `exclude_past_items` parameter is a boolean flag to exclude items found in the user's clickstream history (`items_ls`) from the recommendation results.
`diverse`	*Boolean*	No	**`false`**	Returns a diverse range of recommendations.
`excluded_items`	*List of String*	No	None	Broadcasts the specified value to all users during inference. The final `excluded_items` value for the user is used to either replace, or append to, the existing `excluded_items` values of users obtained from the training data (if the training job option `keep_user_history` is set to **`true`**) depending on the `modify_mode` flag.

`config` Dictionary Parameter (scenario declarations)	Data Type	Required	Default Value	Description
`boosting_attributes`	*Dictionary*	No	None	Broadcasts the specified value to all users during inference. The final `boosting_attributes` value for the user is then applied to the users' recommendation results. The `boosting_attributes` parameter enables user-level attribute boosting in the generated recommendation results.
`filtering_attributes`	*Dictionary*	No	None	Broadcasts the specified value to all users during inference. The final `filtering_attributes` value for the user is then applied to the users' recommendation results. The `filtering_attributes` parameter enables user-level attribute filtering in the generated recommendation results.

Dataset Dictionary Parameters

`data` Dictionary Parameter (user declarations)	Data Type	Required	Default Value	Description
`items_ls`	*List of String*	No	None	Specifies the value of `items_ls` for the particular user (identified by `user_id`). The final `items_ls` value for the user is used to either replace, or append to, the existing `items_ls` values of users obtained from the training data (if the training job option `keep_user_history` is set to **`true`**) depending on the `modify_mode` flag.

`data` Dictionary Parameter (user declarations)	Data Type	Required	Default Value	Description
`modify_mode`	*String*	No	**`append`**	Specifies the action taken to merge the values you provide for scenario and training data. For offline next-item recommendations, the scenario parameters affected by `modify_mode` include: - `items_ls`: Each users' item-interaction history that is provided as training data, and as scenario data. - `excluded_items`: Each users' excluded items list that is provided as training data, and as scenario data. The Personalized Recommendation service supports the following `modify_mode` values: - **`append`**: To append scenario-provided ID lists to existing model-trained data (for example, Training Data: ['a', 'b'], Scenario Data: ['c', 'd'] = Final: ['a', 'b', 'c', 'd']). - **`replace`**: To replace existing model-trained

`data` Dictionary Parameter (user declarations)	Data Type	Required	Default Value	Description
				data with scenario-provided ID lists (for example. Training Data: ['a', 'b'], Scenario Data: ['c', 'd'] = Final: ['c', 'd']).
`excluded_items`	*List of String*	No	None	Specifies the value of `excluded_items` for the particular user (identified by `user_id`). The final excluded_items value for the user is used to either replace, or append to, the existing excluded_items values of users obtained from the training data (if the training job option `keep_user_history` is set to **`true`**) depending on the `modify_mode` flag.

`data` Dictionary Parameter (user declarations)	Data Type	Required	Default Value	Description
`boosting_attributes`	*Dictionary*	No	None	Specifies the value of `boosting_attributes` for the particular user (identified by `user_id`). The final `boosting_attributes` value for the user is then applied to the users' recommendation results. The `boosting_attributes` parameter enables user-level attribute boosting in the generated recommendation results.
`filtering_attributes`	*Dictionary*	No	None	Specifies the value of `filtering_attributes` for the particular user (identified by `user_id`). The final `filtering_attributes` value for the user is then applied to the users' recommendation results. The `filtering_attributes` parameter enables user-level attribute filtering in the generated recommendation results.

Only Model Training Data Sample (`<scenario_name>.scenario.json`):

```
{
```

```
   "scenario_type":"similar-items",
   "config":{
      "only_provided_ids":false
   }
}
```

Both Model Training Data and Scenario Sample (`<scenario_name>.scenario.json`):

```
{
   "scenario_type":"similar-items",
   "config":{
      "only_provided_ids":false
   },
   "data":{
      "test_user1":null,
      "test_user2":null
   }
}
```

Only Scenario Training Data Sample (`<scenario_name>.scenario.json`):

```
{
   "scenario_type":"similar-items",
   "config":{
      "only_provided_ids":true
   },
   "data":{
      "test_user1":null,
      "test_user2":null
   }
}
```

Only Broadcast Parameters Training Data Sample (`<scenario_name>.scenario.json`):

```
{
   "scenario_type":"similar-items",
   "config":{
      "only_provided_ids":false,
      "k":5,
      "explain":true,
      "excluded_items":[
         "trained_item1"
      ],
      "filtering_attributes":{
         "categoricalFeatures":{
            "genre":{
               "values":[
                  "comedy"
               ]
```

```
            }
         }
      },
      "boosting_attributes":{
         "categoricalFeatures":{
            "tag":{
               "values":[
                  "christmas"
               ],
               "weights":[
                  3.5
               ]
            }
         }
      }
   }
}
```

Both Broadcast Parameters and Scenario Training Data Sample (`<scenario_name>.scenario.json`):

```
{
   "scenario_type":"similar-items",
   "config":{
      "only_provided_ids":false,
      "k":5,
      "explain":true,
      "excluded_items":[
         "trained_item1"
      ],
      "filtering_attributes":{
         "categoricalFeatures":{
            "genre":{
               "values":[
                  "comedy"
               ]
            }
         }
      },
      "boosting_attributes":{
         "categoricalFeatures":{
            "tag":{
               "values":[
                  "christmas"
               ],
               "weights":[
                  3.5
               ]
            }
         }
      }
   },
   "data":{
      "trained_user1":{
         "items_ls":[
            "trained_item2"
         ],
         "filtering_attributes":{
            "categoricalFeatures":{
               "genre":{
                  "values":[
                     "comedy",
```

```
                     "children"
                  ]
               }
            }
         }
      },
      "trained_user2":{
         "excluded_items":[
            "trained_item1",
            "trained_item2"
         ],
         "filtering_attributes":{
            "categoricalFeatures":{
               "genre":{
                  "values":[
                     "comedy",
                     "children"
                  ]
               }
            }
         },
         "boosting_attributes":{
            "categoricalFeatures":{
               "tag":{
                  "values":[
                     "christmas",
                     "sci-fi"
                  ],
                  "weights":[
                     3.5,
                     3.5
                  ]
               }
            }
         }
      }
   }
}
```

8.3.1.4.3 Scenario Parameters for Offline Propensity Recommendations

See the details for each `config` dictionary parameter in the scenario configurations for the `scenario_type` `propensity`.

Configuration Dictionary Parameters

`config` Dictionary Parameter (scenario declarations)	Data Type	Required	Default Value	Description
`only_provided_ids`	*Boolean*	No	**`true`**	Specifies the ID set to be used for generating offline inferences. If set to **`true`** (default), the scenario only uses IDs provided in the `data` root key. This is the default value to prevent accidental generation of recommendations for non-intended IDs. If set to **`false`**, the scenario uses both the IDs provided in the `data` root key, and the IDs that were used for training the inference model. For more information about the IDs you can use for the scenario type `similar-items`, see Offline (Scenario) Similar-Item Recommendations [page 150].

`config` Dictionary Parameter (scenario declarations)	Data Type	Required	Default Value	Description
`cached_recommendations_per_page`	*Integer*	No	`-1`	Requirement: `cached_recommendations_per_page` > 10,000 (50 MB before compression, 15 MB after gzip compression). The `cached_recommendations_per_page` parameter determines the number of item rows per page of the output files. Printed in the following format, for example: {scenario-name}-pageX.txt.gzip, where X is the number of the page starting from 1. Calculation for `approximate_file_size` = `cached_recommendations_per_page` x `num_bytes_per_item` / compression ratio. Where `average_bytes_per_item` is 560 bytes, compression ratio is 4x for gzip. Single page pagination is used if `cached_recommendations_per_page` and `cached_file_size_per_page` aren't specified.

`config` Dictionary Parameter (scenario declarations)	Data Type	Required	Default Value	Description
`cached_file_size_per_page`	*Integer*	No	`-1`	Requirement: `cached_file_size_per_page` > 50,000,000 (50 MB before compression, 15 MB after gzip compression). The `cached_file_size_per_page` parameter determines the page size corresponding to the set file size per page. Printed in the following format, for example: {scenario-name}-pageX.txt.gzip, where X is the number of the page starting from 1. Calculation for `approximate_file_size` = `cached_file_size_per_page` / compression ratio. Where compression ratio is 4x for gzip. If `cached_file_size_per_page` is > 50,000,000, it takes precedence over `cached_recommendations_per_page`.

`config` Dictionary Parameter (scenario declarations)	Data Type	Required	Default Value	Description
k	*Integer*	No	`10`	Sets the k parameter for the generation of recommendation results. The k parameter specifies the number of recommendations to be returned.
`explain`	*Boolean*	No	`false`	Sets the `explain` parameter for the generation of recommendation results. The `explain` parameter enables ML Explainability [page 152] for all users in the generated recommendation results.
`explain_k`	*Integer*	No	`10`	Sets the `explain_k` parameter for the generation of recommendation results. The `explain_k` parameter specifies the number of explained recommendations to be returned.

`config` Dictionary Parameter (scenario declarations)	Data Type	Required	Default Value	Description
`item_catalogue`	*Dictionary*	No	None	Broadcasts the specified value to all items during inference. The `item_catalogue` parameter enables cold start items for the generated recommendation results, by either declaring new items with metadata, or updating existing items' metadata so that the model can generate meaningful recommendations for this scenario.
`user_metadata`	*Dictionary*	No	None	Broadcasts the specified value to all users during inference. The `user_metadata` parameter enables cold start users for the generated recommendation results, by either declaring new users with metadata, or updating existing users' metadata so that the model can generate meaningful recommendations for this scenario.

Only Model Training Data Sample (`<scenario_name>.scenario.json`):

```
{
   "scenario_type":"propensity",
   "config":{
      "only_provided_ids":false
   }
```

```
}
```

Only Scenario Training Data Sample (`<scenario_name>.scenario.json`):

```
{
   "scenario_type":"propensity",
   "config":{
      "only_provided_ids":true
   },
   "data":{
      "test_item1":null,
      "test_item2":null
   }
}
```

Both Model Training Data and Scenario Data Sample (`<scenario_name>.scenario.json`):

```
{
   "scenario_type":"propensity",
   "config":{
      "only_provided_ids":false
   },
   "data":{
      "test_item1":null,
      "test_item2":null
   }
}
```

Only Broadcast Parameters Training Data Sample (`<scenario_name>.scenario.json`):

```
{
   "scenario_type":"propensity",
   "config":{
      "only_provided_ids":false,
      "k":5,
      "item_catalogue":{
         "test_item1":{
            "categoricalFeatures":{
               "preferences":{
                  "values":[
                     "catA"
                  ]
               }
            }
         },
         "test_item2":{
            "categoricalFeatures":{
```

```
                "preferences":{
                   "values":[
                      "catC"
                   ]
                }
             }
          }
       }
    }
}
```

8.3.2 Training Job Options

See more details about the required and optional parameters for training job submission.

Required Parameters for Training Job Submission

Category	Parameter	Data Type	Default Value	Description
General	`tenant`	*String*	None	The name of the tenant that you want to use to train the model. Tenant names are unique for each subaccount. **Note** For the `tenant` name, use **only** alphanumeric characters (for example, 0–9, a-z, A-Z), hyphens "-" and underscores "_". In the first and last characters, use **only** alphanumeric characters (hyphens and underscores are **not** supported). No spaces are allowed. The service supports UUID4 string values.
Training	`training_data`	*File*	None	Data file that you want to use to train the model.

Optional Parameters for Training Job Submission

Category	Parameter	Data Type	Default Value	Description
General	`jobid`	*String*	Random UUID is generated.	The job ID of the submitted training job, in UUID format. If you leave `jobid` empty, a random UUID is generated and used as the job ID instead.
	`site`	*String*	*default*	The name of the website that you want to use to train the model. Site names are unique for each tenant of the subaccount. If you leave `site` empty, the *default* name is used. **Note** For the `site` name, use **only** alphanumeric characters (for example, 0–9, a-z, A-Z), hyphens "-" and underscores "_". In the first and last characters, use **only** alphanumeric characters (hyphens and underscores are **not** supported). No spaces are allowed. The service supports UUID4 string values.

Category	Parameter	Data Type	Default Value	Description
Inference	`exclude_past_items`	*Boolean*	*true*	If set to *true*, past items in the user history aren't considered for recommendation (for example, watched movies aren't recommended again). If set to *false*, past items are considered (for example, recurring groceries that the user has bought before should also be considered for recommendation again).
	`explain`	*Boolean*	*false*	If set to *true*, provides explainability of inference results in the form of recommendation scores and score breakdown.
	`infer_cold_start_items`	*Boolean*	*false*	If set to *true*, the model recommends items that aren't part of the training clickstream data, but are found in the item catalogue data during training. This option is only applicable if clickstream data is provided during training. Precomputed recommendation results are also affected.
	`top_k`	*Integer*	10	Returns top-k number of recommendations during batch inference.

Category	Parameter	Data Type	Default Value	Description
Training	`keep_user_history`	*Boolean*	*true*	If set to *true*, the model caches user clickstream history, so that only `user_id` is required in inference and item history is automatically looked up.
	`protected_item_features`	*String*	{}	List of features that are kept even after automatic feature pruning.
	`recommendations_type`	*String*	sequential	Use to select the type of recommendations that suits your use case scenarios. Available recommendations types: ["sequential", "complementary"], where: • sequential: get personalized, or sequential-type recommendations • complementary: get complementary, or basket-type recommendations
	`serve_model`	*Boolean*	*false*	If set to *true*, then automatically triggers the deployment of the model for online serving with inference endpoints.
Batch Inference	`cached_file_size_per_page`	*Integer*	-1	Sets the maximum file size per page in cached results. If not changed, results are paged according to `num_items_per_page`.

Category	Parameter	Data Type	Default Value	Description
	`cached_next_item_recommendations`	*Boolean*	*false*	If set to *true*, provides batch item to user (next-item) batch recommendations. **Note** The batch recommendations are counted towards the inference record metering whereby the number of records is equal to the number of key-value pairs in the precomputed batch results.
	`cached_recommendations_per_page`	*Integer*	10,000	Sets the number of records per page in cached results.
	`cached_similar_item_recommendations`	*Boolean*	*false*	If set to *true*, provides batch item to item (similar-items) batch recommendations. **Note** The batch recommendations are counted towards the inference record metering whereby the number of records is equal to the number of key-value pairs in the precomputed batch results.

8.3.3 Training Job Statuses

Discover all the possible training job statuses, and the next steps required for each status.

Training Job Statuses

Status	Description	Next Step	Response Body Example

SUBMITTED	Training job has been received, allocating resources for training	None. The training job status automatically changes to PENDING once resources have been allocated for a training job. The status change time depends on SAP AI Core response times. It shouldn't take more than 15 minutes.	{
			"id":"9b6ef7ea-faab-11ed-be56-0242ac120002",
			"message":"Model training submitted.",
			"name":"sample_tenantsample_sitesubaccount_id",
			"created":"2023-05-25T02:52:27.528123",
			"updated":"2023-05-25T02:52:27.528132",
			"status":"SUBMITTED",
			"metrics":{
			},
			"exception_info":null,
			"site":"sample_site"
			}

Status	Description	Next Step	Response Body Example

PENDING	Training job is currently in progress.	None. The training job status automatically changes to SUCCEEDED or FAILED upon completion. The status change time can take up to 5 hours.	{
			"id":"9b6ef7ea-faab-11ed-be56-0242ac120002",
			"message":"Model training submitted.",
			"name":"sample_tenantsample_sitesubaccount_id",
			"created":"2023-05-25T02:52:27.528123",
			"updated":"2023-05-25T02:52:27.528132",
			"status":"PENDING",
			"metrics":{
			},
			"exception_info":null,
			"site":"sample_site"
			}

Status	Description	Next Step	Response Body Example

SUCCEEDED	Training job has been completed successfully.	If the training job option `serve_model` is set to *true* during the job submission, the model deployment workflow is triggered automatically.
		
		If the training job option `serve_model` is set to *false*, you have to deploy the model manually.
			If the training job status is SUCCEEDED, post-training reports and metrics are provided under "metrics" as shown in the following example:
			
			`{`
			`"id":"9b6ef7ea-faab-11ed-be56-0242ac120002",`
			`"message":"Model training succeeded.",`
			`"name":"sample_tenantsample_sitesubaccount_id",`
			`"created":"2023-07-07T03:38:19.629787",`
			`"updated":"2023-07-07T04:00:45.872739",`
			`"status":"SUCCEEDED",`
			`"metrics":{`
			`"job_secs":36,`
			`"run_stats":{`
			`"job_id":"9b6ef7ea-faab-11ed-be56-0242ac120002",`
			`"model_parameters":{`
			`"model_type":"sequential"`
			`},`
			`"app_version":"1.4.2",`
			`"serving_version":"1.4.2",`
			`"job_options":"...",`
			`"model_evaluation":"...",`

Status	Description	Next Step	Response Body Example

			"data_quality_report":"...",
			"training_data_stats":{
			"clickstream.csv":"255KB",
			"user_metadata.json":"4KB",
			"item_catalogue.json":"195KB"
			}
			}
			},
			"exception_info":null,
			"site":"sample_site"
			}

Status	Description	Next Step	Response Body Example

FAILED	Training job has failed.	Take actions based on the error message details provided in the response under "exception_info".	If the training job status is FAILED, an error message is provided under "exception_info" as shown in the following example:
			
			{
			"id":"9b6ef7ea-faab-11ed-be56-0242ac120002",
			"message":"Model training failed.",
			"name":"sample_tenantsample_sitesubaccount_id",
			"created":"2023-07-07T03:38:19.629787",
			"updated":"2023-07-07T04:00:45.872739",
			"status":"FAILED",
			"metrics":{
			},
			"exception_info":{
			"description":"Data requirements not met; Personalized Recommendation service requires either Clickstream and/or Item Catalogue training data to provide recommendations",
			"status_code":"400-04-02",
			"exception_class":"InvalidTrainingFilesException"
			},

Status	Description	Next Step	Response Body Example

			<pre>"site":"sample_site"
			}</pre> In this case, the training data (**data.zip**) file provided didn't meet the requirements to train a machine learning model for the Personalized Recommendation service.

8.3.4 Trained Model Metrics

See more details about the metrics used to evaluate trained models. Find these metrics in the `run_stats` report for every model trained successfully.

Metric	Payload Key	Description
Average Hit Rate (measured in %)	`Avg_hr`	Average percentage of users with correct recommendations. For example, the item selected by a user is inside the top-k recommendation results. The hit rate is a measure of how accurate each recommendation was for a user. A higher average hit rate indicates that the recommender engine is effectively matching user preferences with appropriate recommendations.

Metric	Payload Key	Description
Mean Reciprocal Rank (ranges from 0 to 1)	`MRR`	Evaluates how quickly a ranking system can show the first relevant item in the top-k recommendation results. Scores closer to 1.0 are better. For example, if "recommendations":["item1", "item2", "item3"]: • If "item1" is the selected item, the score is 1/1 (1.0) • If "item2" is the selected item, the score is 1/2 (0.5) • If "item3" is the selected item, the score is 1/3 (0.33) `MRR` reflects how fast, on average, we can find a relevant item for each user as it emphasizes the importance of being correct early in the list. A `MRR` score of 0.5 indicates that, on average, the selected items are ranked around second in the recommendations.
Average Normalized Discounted Cumulative Gain (ranges from 0 to 1)	`Avg_NDCG`	Measures the relevance of recommendations based on the position of the consumed item within the recommended items. It's similar to `MRR`, except that it uses a different scale for calculation. `Avg_NDCG` is measured based on an ideal ranking of all the recommended items sorted in decreasing order of relevance. And then it's measured how close the algorithm output comes to this perfect order. A higher `Avg_NDCG` value indicates that the service is doing a better job of presenting relevant items at the top of the list.

Metric	Payload Key	Description
Coverage (measured in %)	`Coverage`	The percentage of items that the model is able to recommend. For example, among all 1000 items on a website: • 1% coverage for a pure popularity recommender that always recommends the top 10 popular items. • 100% coverage for a pure random recommender that randomly recommends all the items. `Coverage` depends on the number of interactions in the clickstream or rather how popular the items are. A sequential-based recommender generally suggests the items that many users have interacted with, and if there are many rare items in the clickstream, the coverage score isn't expected to be high. A random recommender can easily have coverage of 100% because it recommends products randomly.

8.4 Inference

Use the Inference APIs to get online recommendations from the Personalized Recommendation service after a successful training job and model deployment. Each trained model belongs to a tenant with site-support enabled (each site has its own model, allowing a tenant to have multiple models under multiple sites). Ensure that the model has been trained and is serving on the respective tenant and site.

Request

Base URL: url value from outside the uaa section of the service key

URL Path Extension: `/doc`

Inference APIs

HTTP Method	URL Endpoint Path	Description
GET	`/standard/rs/v1/tenants/{tenant}/recommendations/batch-results`	Get batch inference results. You can use this endpoint to get both offline and online recommendations.
POST	`/standard/rs/v1/tenants/{tenant}/recommendations/next-items`	Get next-item recommendations based on the type of recommendations specified during training job submission. There are two types of next-item recommendations: sequential (default) and complementary, that are specified via Training Job Options [page 79] during training job submission.
POST	`/standard/rs/v1/tenants/{tenant}/recommendations/similar-items`	Get similar-item recommendations that have the highest similarity to a given item.
POST	`/standard/rs/v1/tenants/{tenant}/recommendations/smart-search`	Get smart-search recommendations based on item or text queries.
POST	`/standard/rs/v1/tenants/{tenant}/recommendations/user-affinity`	Get user-affinity recommendations of item attribute affinities for the user based on past-item interactions and user profile.

> ⓘ Note
>
> The results from the POST requests previously listed are computed online via a live model instance served on SAP AI Core.

> → Remember
>
> To get online inference results, a model must be successfully trained, and an online model instance must be successfully served.

Related Information

Inference Options [page 93]
ML Explainability [page 152]

8.4.1 Inference Options

The Personalized Recommendation service supports the following inference options:

Inference Option	Input
Next-item recommendations	List of multiple item IDs. See also Next-Item Recommendations [page 93].
Similar-item recommendations	List of one item ID. See also Similar-Item Recommendations [page 112].
Smart-search results	Free form text and/or dictionary of attributes. See also Smart-Search Results [page 125].
User-affinity recommendations	List of multiple item IDs, or a user ID. See also User-Affinity Recommendations [page 136].
Offline (Scenario) Recommendations	List of multiple item IDs for the scenario type `next-items`, and list of one item ID for the scenario type `similar-items`. See also Offline (Scenario) Recommendations [page 148].

8.4.1.1 Next-Item Recommendations

Get top-k recommended items for users based on their user ID, and/or a list of past item interactions (`items_ls`).

Optionally assign ratings to each interaction to adjust its influence on the recommendation results, and apply boosting (`boosting_attributes`) and filtering (`filtering_attributes`) attributes to customize the inference call. You can exclude recommendations from the item catalogue (`excluded_items` and `exclude_past_items`).

You can also add optional metadata to the inference results to either define new users, and/or items unknown by the model (not found in the training dataset). Or to redefine existing users and/or items by overwriting them with a different set of attributes and/or weights. This way, newly defined items address the cold start scenario, while redefined items can be used to temporarily alter the user and/or item metadata to skew recommendations towards some ongoing promotions (custom attribute recommendations).

Additionally, you can use the ML Explainability feature (`explain`) to get insights into recommendation results by breaking down the contribution of each item towards that recommendation.

For example, in a learning recommendation scenario, if you have the items “Data Science Essential Training” and “Data Science Intermediate Training” in your clickstream, the next-item recommendations inference option recommends to you the item “Data Science Advanced Training”.

There are two types of next-item recommendations: sequential (default) and complementary. Select the type you want to use via Training Job Options [page 79] (parameter `recommendations_type`) during training job submission. See Next-Item Recommendation Types [page 108].

Inference Concept	Inference Parameter	Required	Data Type	Default Value	Description	Sample Payload

Attribute boosting	`boosting_attributes`	No	*Dictionary*	None	A list of objects that represents feature-types of boosting attributes to be applied in the recommendation results.
					
					Attribute boosting aims to increase the likelihood of recommending items containing certain categorical attributes in a controlled manner. While being similar to filtering, this feature aims to promote certain categories of items, rather than to remove certain items from recommendations altogether that don't contain certain attribute values.
					
					Each feature-type can contain multiple boosting attributes, represented by their features, values, and respective weight. Multiple boosting attributes can be applied simultaneously.
						<pre>{
						"boosting_attributes":[
						{
						"categoricalFeatures":{
						"categories":{
						"values":[
						"Drama",
						"Comedy"
						]
						,
						"weights":[
						0.2,
						0.4
						]
						},
						{
						"..."
						}
						}
						}
						]
						}</pre>
						Where "categories" is a categorical-type

Inference Concept	Inference Parameter	Required	Data Type	Default Value	Description	Sample Payload
					For this parameter to be valid, the input has to: • Contain valid boosting feature-types (for example, categorical features). • Contain valid features in each feature-type that are identical to the features in the item catalogue training data. • Contain valid booster values means they exactly match at least one known item. Invalid values are ignored.	feature in the item catalogue, and "Drama" and "Comedy" are its values to be boosted. Each float value in "weights" corresponds to its respective value (for example, "Drama" has a boosting value of 0.2).

Inference Concept	Inference Parameter	Required	Data Type	Default Value	Description	Sample Payload

Attribute filtering	`filtering_attributes`	No	*Dictionary*	None	A list of objects that represents feature-types of filters to be applied in the recommendation results.
					
					Attribute filtering changes the order of the recommendation results based on the number of applied filters, where items matching more filters are ranked ahead of items matching fewer or no filters. This feature aims to remove certain items from recommendations that don't contain certain attribute values.
					
					Each feature-type can contain multiple filters, represented by their features and values. Multiple feature-types and multiple filters (within each feature-type) can be applied simultaneously. A cut-off index is provided to show the index of
						<pre>{
						"filtering_attributes":[
						{
						"categoricalFeatures":{
						"categories":{
						"values":[
						"Drama",
						"Comedy"
						]
						},
						"tags":{
						"values":[
						"Disney"
						]
						}
						}
						}
						]
						}</pre>
						Where "categories" is a categorical-type feature in the item catalogue, and "Drama" and "Comedy"

Inference Concept	Inference Parameter	Required	Data Type	Default Value	Description	Sample Payload
					items with at least one applied filter. For this parameter to be valid, the input has to: • Contain valid filterable feature-types (for example categorical and numerical features). • Contain valid features in each feature-type that are identical to the features in the item catalogue training data. • Contain valid filter values. which mean they exactly match at least one known item. Invalid values are ignored.	are its values to be filtered. Similarly, "tags" is also a categorical-type feature and "Disney" is its value to be filtered.

Inference Concept	Inference Parameter	Required	Data Type	Default Value	Description	Sample Payload
Cold start items and users	`metadata`	No	*Dictionary*	None	A dictionary of cold start items and users, with formatting identical to items in the item_catalogue.json or user_metadata.json files. These items and users will then be referenced in the `items_ls` or `user` parameters during inference. Cold start items refer to items that we want to include in our context (`items_ls` and/or `user`), that weren't part of the training data. For inference purposes, this feature aims to make meaningful recommendations using new items and users as input. Specify cold start items and users via the inference parameter metadata: only the current inference call is affected. Added	`{ "metadata":{ "items": { "coldstart_item_id1":{ "categoricalFeatures":{ "cat_feature1":{ "values":[ "v1", "v2" ] } , "cat_feature2":{ "..." } }, "textFeatures"{ "..." }`

Inference Concept	Inference Parameter	Required	Data Type	Default Value	Description	Sample Payload

					items and/or users can only be used as part of contextual input data.
					The goal of this feature is to ease DevOps loads by allowing the addition of new item and user metadata into the model without triggering a model retraining and redeployment, while still being able to use them in recommendations.
						<pre>},
						"coldstart_item_id2":{
						"..."
						}
						},
						"users":{
						"coldstart_user_id1":{
						"categoricalFeatures":{
						"cat_feature1":{
						"values":[
						"v1",
						"v2"
						]
						}
						,
						"cat_feature2":{
						"..."
						}
						},</pre>

Inference Concept	Inference Parameter	Required	Data Type	Default Value	Description	Sample Payload

						<pre>"textFeatures"{
						
						"..."
						}
						},
						
						"coldstart_user_id2":{
						
						"..."
						}
						}
						}
						}</pre>

Excluded items	`excluded_items`	No	*List*	None	A list of `item_id` specifying the items to be excluded from recommendation results. This parameter is used to exclude groups of items from the recommendation results from the current inference call only.	<pre>{
						
						"excluded_items":[
						
						"item_id1",
						
						"item_id2"
						]
						}</pre>

Exclude Past Items	`exclude_past_items`	No	*Boolean*	*true*	A boolean flag to exclude items found in the user's clickstream history (`items_ls`) from the recommendation results.	<pre>{
						
						"exclude_past_items":true
						}</pre>

Inference Concept	Inference Parameter	Required	Data Type	Default Value	Description	Sample Payload
ML Explainability	`explain`	No	*Boolean*	*false*	A boolean flag to specify the scores assigned by the model to each recommended item in the results list. The goal of this feature is to provide quantifiable insights into how the recommendation engine computes the recommendation results, in the form of scores. There are three main types of scores in the ML Explainability feature: • Confidence Score: the overall score of the recommended item. • Item Attribute Contribution: the contribution of each attribute in the item and/or user metadata towards the	`{ "explain":false }`

Inference Concept	Inference Parameter	Required	Data Type	Default Value	Description	Sample Payload
					recommendation results. • Sequence Attention: the contribution of each item in the clickstream (`items_ls`) towards the recommendation results.	

Inference Concept	Inference Parameter	Required	Data Type	Default Value	Description	Sample Payload
Past item interactions (clickstream)	`items_ls`	Yes (if `user` is empty)	*List*	None	A list of `item_id` that represents the past item interactions (clickstream) of the user for generating the recommendations. For this parameter to be valid, the input has to: • Correspond to an object entry in the item catalogue training data used to train the model, or. • To be provided as an entry in the metadata parameter as a coldstart item.	`{ "items_ls":[ "item_id1", "item_id2" ] }`

Inference Concept	Inference Parameter	Required	Data Type	Default Value	Description	Sample Payload

Ratings	`ratings`	No	*List*	None	A list of numerical rating scores corresponding to each item in the user's clickstream input (`items_ls`) that is used to assign a rating to each item in the input sequence.
					
					The allowed range for ratings is -1 to 1. Higher rated items contribute more to the output recommendations, while lower rated items represent negative sentiment for the item. A rating of 0 has the same result as omitting the item.
					
					Ratings result in the following behavior:
					
					• A positive rating (between 0 and 1) increases the weight of the corresponding item
					• A negative rating (be-
						<pre>{
						"items_ls":[
						"item_id1",
						"item_id2"
						],
						"ratings":[
						-0.5,
						1
						]
						}</pre>

Inference Concept	Inference Parameter	Required	Data Type	Default Value	Description	Sample Payload
					tween -1 and 0) represents negative sentiment for the corresponding item • A rating of 0 has a similar effect of omitting the corresponding item from the clickstream **Note** If `exclude_past_items` is set to *true*, the corresponding item is omitted from the recommendation results as it is still part of the input clickstream (`items_ls`).	

Inference Concept	Inference Parameter	Required	Data Type	Default Value	Description	Sample Payload
Top-k results	`k`	No	*Integer*	*10*	An integer specifying the number of recommended items to be returned in the inference output. The output contains the top-k recommendation results.	`{ "k":10 }`
User	`user`	Yes (if `items_ls` is empty)	*String*	None	The `user_id` of the user that is consuming the recommendations. For this parameter to be valid, the input has to: • Correspond to an object entry in the user metadata training data used to train the model, or. • To be provided as an entry in the metadata parameter as a cold start user.	`{ "user":"user_id1" }`

Related Information

Next-Item Recommendation Types [page 108]
Request [page 108]

Response [page 109]

8.4.1.1.1 Next-Item Recommendation Types

The next-item recommendation types are mutually exclusive, and only one type of recommendations can be selected from the following recommendation types:

Recommendation Type	Description	Example
Sequential (default)	Recommends items that the user wants to view next in a sequential manner. This recommendation type is useful for many e-commerce and content recommendation scenarios to retain user engagement.	Items A, B, C and D are known to be viewed in this exact sequence. If my current item interaction history contains the items A, B and C, the sequential recommendation type will recommend me item D.
Complementary	Recommends items that are known to be viewed together in a single session. This recommendation type is useful for group purchase or cart scenarios to boost conversion rates or improve up-sell or cross-selling opportunities.	Items A, B, C and D are known to be purchased together (complementary). If my current cart items include items A, B and D, the complementary recommendation type will recommend me item C.

8.4.1.1.2 Request

Request Example Payload

Use the following `payload` to receive recommendations with attribute boosting (`boosting_attributes`) but without attribute filtering (`filtering_attributes`) and cold start items and users (`metadata`).

```
{
   "user":"user_id1",
   "items_ls":[
      "item_id1",
      "item_id2"
   ],
   "boosting_attributes":{
      "categoricalFeatures":{
         "categories":{
            "values":[
               "Drama",
               "Comedy"
            ],
            "weights":[
               0.2,
```

```
                0.4
            ]
        }
    }
},
"k":3,
"explain":true
}
```

The endpoint returns three recommendation results ("k":3) based on the current clickstream ("items_ls": ["item_id1", "item_id2"]) of the user with user_id1 ("user":"user_id1").

The recommendation results are biased towards results with categories values of "Drama" and "Comedy" as they're the feature values that have been boosted ("boosting_attributes":[{"categoricalFeatures":{"categories": {"values":["Drama","Comedy"],"weights":[0.2,0.4]).

As ML explainability is enabled ("explain":true), in the response, you'll receive quantifiable insights into how the recommendations engine computes the recommendation results.

8.4.1.1.3 Response

Response Fields

JSON Field	Description
`recommendations`	Top-k recommendation results.
`scores`	Values between 0 and 1. The confidence `scores` represent how certain the model is about the recommendations. The higher the score the more confident the model is that the recommendation is correct. If the score is close to 1, the model is very certain.
`status`	The response is given in one of the following `status`: • `SUCCESS: Trained user. All items are trained.` The user and all items in their history were present in the training dataset. • `ERROR: Invalid user. All items are invalid.` The user and all items in their history were absent from the training dataset. • `WARNING: Cold-Start/Invalid user. All items are Invalid/Cold-Start.` Either the user is invalid and all items are cold start, or items are invalid and the user is cold start or both are cold start. • `WARNING: Trained/Cold-Start/Invalid user. All items are invalid/Items have mixed type (List of Trained/Cold-Start/Invalid).` The user and items provided are a mix of trained, cold start, and invalid.

JSON Field	Description
`status_code`	The response is given in one of the following `status_code`: • **0**: invalid, the user and all items in their history were absent from the training dataset. • **1**: cold start, either the user is invalid and all items are cold start, or items are invalid and the user is cold start or both are cold start. • **2**: mixed, the user and items provided are a mix of trained, cold start, and invalid. • **3**: trained, the user and all items in their history were present in the training dataset.
`item_sequence`	The users' past clickstream.
`sequence_attention`	Each list tells the contribution of different past items in the users' history towards that recommendation. The values are between 0 and 1 and they sum up to 1. The higher the value the more confident the model is that the recommendation is correct. Each value can be interpreted as the "percentage of attention" given by the model to an element in the input sequence when recommending a certain item. Note that some values can be negative, since an item attribute can have a negative contribution when recommending the item (compensated by the other attributes). The outputs in the sequence attention layers are an average of the attention weights from the first and last attention layers of the model (input and output). Together they give a good understanding of the importance of the past items in the recommendation.
`item_attribute_contribution`	Each list tells the contribution of the different item attributes towards that recommendation. The values are between 0 and 1. The higher the value the more confident the model is that the recommendation is correct. The first element in the list is always from the item ID.
`user_attribute_contribution`	Each list tells the contribution of the different user attributes towards that recommendation. The values are between 0 and 1. The higher the value the more confident the model is that the recommendation is correct. The first element in the list is always from the user ID.
`relevant_cutoff_index`	The number of recommendations provided by the model.
`item_attributes`	Categorical attributes of items.
`user_attributes`	Categorical attributes of users.

Response Example

This response returns top-k recommendations results ("k":3 was used in the request `payload`) .

200 “OK”

```
{
   "recommendations":[
      "4967",
      "4603",
      "4969"
   ],
```

```
    "scores":[
       0.5481,
       0.5221,
       0.4597
    ],
    "status":"SUCCESS: Trained user. All items are trained.",
    "status_code":3,
    "item_sequence":[
       "4939",
       "4585"
    ],
    "sequence_attention":[
       [
          0.5791,
          0.4209
       ],
       [
          0.3155,
          0.6845
       ],
       [
          0.4333,
          0.5667
       ]
    ],
    "item_attribute_contribution":[
       [
          0.6366,
          0.245,
          0.0309,
          0.0875
       ],
       [
          0.6352,
          0.2572,
          0.0228,
          0.0847
       ],
       [
          0.5475,
          0.2921,
          0.0702,
          0.0902
       ]
    ],
    "user_attribute_contribution":[
       [
          0,
          -0.9998
       ],
       [
          0,
          0.9996
       ],
       [
          0,
          -0.9998
       ]
    ],
    "relevant_cutoff_index":3,
    "item_attributes":[
       "id",
       "domain",
       "description",
       "title"
    ],
    "user_attributes":[
       "id",
```

```
      "sig"
   ]
}
```

Based on the user's clickstream history "item_sequence", a number of recommendation results are returned ("recommendations":["4967", "4603", "4969"]) along with their scores ("scores":[0.5481, 0.5221, 0.4597]). In this example, the item "4967" has a score of 0.5481, the item "4603" has a score of 0.5221, and the item "4969" has a score of 0.4597.

When you enable ML explainability, you can better understand the "scores". For "sequence_attention", "item_attribute_contribution" and "user_attribute_contribution", the position of the nested list corresponds to the position of the item ID in the "recommendations" list. Using "sequence_attention" as an example, the first nested list [0.5791, 0.4209] is the "sequence_attention" for the first item ID in "recommendations" that is "4967". And the second nested list [0.3155, 0.6845] is the "sequence_attention" for the second item ID in "recommendations" that is "4603".

Matrix Representation

Sequence Attention

Recommendations	Score contributed by item ID 4939	Score contributed by item ID 4585
4967	0.5791	0.4209
4603	0.3155	0.6845
4969	0.4333	0.5667

Item Attribute Contribution

Recommendations	ID	Domain	Description	Title
4967	0.6366	0.245	0.0309	0.0875
4603	0.6352	0.2572	0.0228	0.0847
4969	0.5475	0.2921	0.0702	0.0902

User Attribute Contribution

Recommendations	ID	SIG
4967	0	-0.9998
4603	0	0.9996
4969	0	-0.9998

8.4.1.2 Similar-Item Recommendations

Get top-k recommended items for users based on their most recent single item interaction by receiving an item ID (`items`).

You can also add optional metadata to the inference results to either define new users and/or items unknown by the model (not found in the training dataset). Or to redefine existing users and/or items by overwriting

them with a different set of attributes and/or weights. This way, newly defined items address the cold start scenario, while redefined items can be used to temporarily alter the user and/or item metadata to skew recommendations towards some ongoing promotions (custom attribute recommendations). Optionally, apply boosting (boosting_attributes) and filtering (filtering_attributes) attributes to customize the inference call. You can exclude recommendations from the item catalogue (excluded_items).

Additionally, you can use the ML Explainability feature (explain) to get insights into recommendation results by breaking down the contribution of each item towards that recommendation.

For example, in a learning recommendation scenario, if you have the items "Data Science Essential Training" and "Optionally, apply boosting (boosting_attributes) and filtering (filtering_attributes) attributes to customize the inference call. You can excludeData Science Intermediate Training" in your clickstream, the similar-item recommendations inference option only considers the second item "Data Science Intermediate Training", and recommends items such as "Machine Learning Intermediate Training", "Artificial Intelligence Intermediate Training" and so on.

Inference Concept	Inference Parameter	Required	Data Type	Default Value	Description	Sample Payload

Attribute boosting	`boosting_attributes`	No	*Dictionary*	None	A list of objects that represents feature-types of boosting attributes to be applied in the recommendation results.
					
					Attribute boosting aims to increase the likelihood of recommending items containing certain categorical attributes in a controlled manner. While being similar to filtering, this feature aims to promote certain categories of items, rather than to remove certain items from recommendations altogether that don't contain certain attribute values.
					
					Each feature-type can contain multiple boosting attributes, represented by their features, values, and respective weight. Multiple boosting attributes can be applied simultaneously.
						<pre>{
						"boosting_attributes":[
						{
						"categoricalFeatures":{
						"categories":{
						"values":[
						"Drama",
						"Comedy"
						]
						,
						"weights":[
						0.2,
						0.4
						]
						},
						{
						"..."
						}
						}
						}
						]
						}</pre>
						Where "categories" is a categorical-type

Inference Concept	Inference Parameter	Required	Data Type	Default Value	Description	Sample Payload
					For this parameter to be valid, the input has to: • Contain valid boosting feature-types (for example, categorical features) • Contain valid features in each feature-type that are identical to the features in the item catalogue training data. • Contain valid booster values, which mean they exactly match at least one known item. Invalid values are ignored.	feature in the item catalogue, and "Drama" and "Comedy" are its values to be boosted. Each float value in "weights" corresponds to its respective value (for example, "Drama" has a boosting value of 0.2).

Inference Concept	Inference Parameter	Required	Data Type	Default Value	Description	Sample Payload

Attribute filtering	`filtering_attributes`	No	*Dictionary*	None	A list of objects that represents feature-types of filters to be applied in the recommendation results.
					
					Attribute filtering changes the order of the recommendation results based on the number of applied filters, where items matching more filters are ranked ahead of items matching fewer or no filters. This feature aims to remove certain items from recommendations that don't contain certain attribute values.
					
					Each feature-type can contain multiple filters, represented by their features and values. Multiple feature-types and multiple filters (within each feature-type) can be applied simultaneously. A cut-off index is provided to show the index of
						<pre>{
						"filtering_attributes":[
						{
						"categoricalFeatures":{
						"categories":{
						"values":[
						"Drama",
						"Comedy"
						]
						},
						"tags":{
						"values":[
						"Disney"
						]
						}
						}
						}
						]
						}</pre>
						Where "categories" is a categorical-type feature in the item catalogue, and "Drama" and "Comedy"

Inference Concept	Inference Parameter	Required	Data Type	Default Value	Description	Sample Payload
					items with at least one applied filter. For this parameter to be valid, the input has to: • Contain valid filterable feature-types (for example categorical and numerical features) • Contain valid features in each feature-type that are identical to the features in the item catalogue training data. • Contain valid filter values, which mean they exactly match at least one known item. Invalid values are ignored.	are its values to be filtered. Similarly, "tags" is also a categorical-type feature and "Disney" is its value to be filtered.

Inference Concept	Inference Parameter	Required	Data Type	Default Value	Description	Sample Payload
Cold start items	`metadata`	No	*Dictionary*	None	A dictionary of cold start items, with formatting identical to items in the item_catalogue.json file. These items will then be referenced in the item parameter during inference. Cold start items refer to items that we want to include in our context (`item`), that weren't part of the training data. For inference purposes, this feature aims to make meaningful recommendations using new items as input. Specifying cold start items via the inference parameter metadata: only the current inference call is affected. Added items can only be used as part of contextual input data. The goal of this feature is to ease DevOps	`{ "metadata":{ "items": { "coldstart_item_id1":{ "categoricalFeatures":{ "cat_feature1":{ "values":[ "v1", "v2" ] } , "cat_feature2":{ "..." } }, "textFeatures"{ "..." }`

Inference Concept	Inference Parameter	Required	Data Type	Default Value	Description	Sample Payload

					loads by allowing the addition of new item metadata into the model without triggering a model retraining and redeployment, while still being able to use them in recommendations.	<pre>},
						
						"coldstart_item_id2":{
						
						"..."
						}
						}
						}
						}</pre>

Excluded items	`excluded_items`	No	*List*	None	A list of `item_id` specifying the items to be excluded from recommendation results. This parameter is used to exclude groups of items from the recommendation results from the current inference call only.	<pre>{
						"excluded_items":[
						"item_id1",
						"item_id2"
						]
						}</pre>

Inference Concept	Inference Parameter	Required	Data Type	Default Value	Description	Sample Payload
ML Explainability	`explain`	No	*Boolean*	*false*	A boolean flag to specify the scores assigned by the model to each recommended item in the results list. The goal of this feature is to provide quantifiable insights into how the recommendation engine computes the recommendation results, in the form of scores. There are three main types of scores in the ML Explainability feature: • Confidence Score: the overall score of the recommended item • Item Attribute Contribution: the contribution of each attribute in the item and/or user metadata towards the	`{ "explain":false }`

Inference Concept	Inference Parameter	Required	Data Type	Default Value	Description	Sample Payload
					recommendation results	
Past item interactions (single-item)	`item`	Yes	*String*	None	An `item_id` that represents the item of which the user wants to get similar-item recommendations. For this parameter to be valid, the input has to: • Correspond to an object entry in the item catalogue training data used to train the model, or. • To be provided as an entry in the metadata parameter as a cold start item.	`{ "item":"item_id1" }`
Top-k results	`k`	No	*Integer*	*10*	An integer specifying the number of recommended items to be returned in the inference output. The output contains the top-k recommendation results.	`{ "k":10 }`

Related Information

Request [page 122]
Response [page 123]

8.4.1.2.1 Request

Request Example Payload

Use the following `payload` to receive recommendations with attribute boosting (`boosting_attributes`) but without attribute filtering (`filtering_attributes`) and cold start items (`metadata`).

```
{
   "item":"item_id1",
   "boosting_attributes":[
      {
         "categoricalFeatures":{
            "categories":{
               "values":[
                  "Drama",
                  "Comedy"
               ],
               "weights":[
                  0.2,
                  0.4
               ]
            }
         }
      }
   ],
   "k":3,
   "explain":true
}
```

The endpoint returns three recommendation results ("k":3) based on the item the user has selected ("item":"item_id1").

The recommendation results are biased towards results with categories values of "Drama" and "Comedy" as they're the feature values that have been boosted ("boosting_attributes":[{"categoricalFeatures":{"categories":{"values":["Drama","Comedy"],"weights":[0.2,0.4]).

As ML explainability is enabled ("explain":true), in the response, you'll receive quantifiable insights into how the recommendations engine computes the recommendation results.

8.4.1.2.2 Response

Response Fields

JSON Field	Description
`recommendations`	Top-k recommendation results.
`scores`	Values between 0 and 1. The confidence `scores` represent how certain the model is about the recommendations. The higher the score the more confident the model is that the recommendation is correct. If the score is close to 1, the model is very certain.
`status`	The response is given in one of the following `status`: • `SUCCESS: Trained user. All items are trained.` The user and all items in their history were present in the training dataset. • `ERROR: Invalid user. All items are invalid.` The user and all items in their history were absent from the training dataset. • `WARNING: Cold-Start/Invalid user. All items are Invalid/Cold-Start.` Either the user is invalid and all items are cold start, or items are invalid and the user is cold start or both are cold start. • `WARNING: Trained/Cold-Start/Invalid user. All items are invalid/Items have mixed type (List of Trained/Cold-Start/Invalid).` The user and items provided are a mix of trained, cold start, and invalid.
`status_code`	The response is given in one of the following `status_code`: • **0**: invalid, the user and all items in their history were absent from the training dataset. • **1**: cold start, either the user is invalid and all items are cold start, or items are invalid and the user is cold start or both are cold start. • **2**: mixed, the user and items provided are a mix of trained, cold start, and invalid. • **3**: trained, the user and all items in their history were present in the training dataset.
`item_attribute_contribution`	Each list tells the contribution of the different item attributes towards that recommendation. The values are between 0 and 1. The higher the value the more confident the model is that the recommendation is correct. The first element in the list is always from the item ID.
`relevant_cutoff_index`	The number of recommendations provided by the model.
`item_attributes`	Categorical attributes of items.

Response Example

This response returns top-k recommendations results ("k":3 was used in the request `payload`) .

200 “OK”


```
{
   "recommendations":[
      "560784",
      "525781",
      "489784"
   ],
   "scores":[
      0.757,
      0.7307,
      0.7268
   ],
   "status":"SUCCESS: Trained item.",
   "status_code":3,
   "item_attribute_contribution":[
      [
         0.4403,
         0.1734,
         0.3862
      ],
      [
         0.4562,
         0.2137,
         0.3301
      ],
      [
         0.4586,
         0.1352,
         0.4062
      ]
   ],
   "relevant_cutoff_index":3,
   "item_attributes":[
      "domain",
      "description",
      "title"
   ]
}
```


Based on the user's most-recent (single) item interaction, a number of recommendation results are returned ("recommendations":["560784","525781","489784"]) along with their scores ("scores":[0.757,0.7307,0.7268]). In this example, the item "560784" has a score of 0.757, the item "525781" has a score of 0.7307, and the item "489784" has a score of 0.7268.

When you enable ML explainability, you can better understand the "scores". For "item_attribute_contribution", the position of the nested list corresponds to the position of the item ID in the "recommendations" list. The first nested list [0.4403,0.1734,0.3862] is the "item_attribute_contribution" for the first item ID in "recommendations" that is "560784". And the second nested list [0.4562,0.2137,0.3301] is the "item_attribute_contribution" for the second item ID in "recommendations" that is "525781".

Matrix Representation

Item Attribute Contribution

Recommendations	Domain	Description	Title
560784	0.4403	0.1734	0.3862
525781	0.4562	0.2137	0.3301
489784	0.4586	0.1352	0.4062

8.4.1.3 Smart-Search Results

Get top-k recommended items (in the form of search results) based on a list of free form text input (`string_queries`), and/or a list of attribute values (`attribute_queries`).

You can further personalize these queries by providing a user context, either a user ID or a list of past interactions for a user (items_ls). Having flexible inputs, this function enables the smart-search use case for simple queries (for example, keyword or attribute value), as well as for complex queries combining multiple attribute values with free form text and user personalization.

This function also has some configuration parameters to modify the handling of free form text (expand_synonyms and expand_ngrams). If past item interactions are provided, they can be customised by assigning ratings to each interaction to adjust its influence on the search recommendation results.

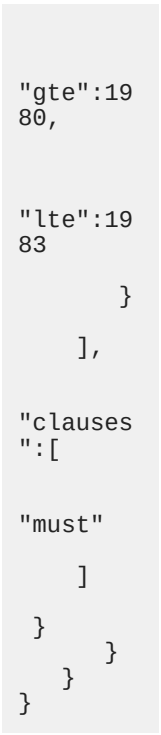
Additionally, you can use the ML Explainability feature (explain) to get insights into recommendation results by breaking down the contribution of each item towards that recommendation.

For example, you have “Jungle Book” in your context (clickstream) and search for the text query “king”, the search engine returns “Lion King”. And if you have “The Hobbit” in your context (clickstream) and search for the text query “king”, the search engine returns “Lord of the Rings - Return of the King”.

Inference Concept	Inference Parameter	Required	Data Type	Default Value	Description	Sample Payload

Attribute query	`attribute_queries`	No	*Dictionary*	None	A dictionary of objects that represent feature-types of attribute queries to be applied in the search results. There can be multiple attribute queries applied, each specifying an attribute, as well as its query values and clauses.
					
					There are two ways to use attribute queries:
					• If used with string queries, they restrict the scope of the string query to a set of attributes, acting as a filter.
					• If used on their own, they just act as a filter for items matching attributes, the results are ranked based on the number of at-
						<pre>{
						"attribute_queries":{
						"categoricalFeatures":{
						"tags":{
						"values":[
						"star wars",
						"imdb top 250",
						"trilogy"
						],
						"clauses":[
						"must",
						"must",
						"should"
						]
						}
						},
						"numericalFeatures":{
						"year":{
						"values":[
						{</pre>

Inference Concept	Inference Parameter	Required	Data Type	Default Value	Description	Sample Payload

					tributes matched.
					Valid clauses include:
					• Categorical features: ['must', 'should']
					  • If clauses aren't defined, then the default value is used.
					  • If clauses are defined, the length of clauses must be the same as the length of values.
					• Numerical features: ['gt', 'gte', 'lt' lte']
						

Inference Concept	Inference Parameter	Required	Data Type	Default Value	Description	Sample Payload
ML Explainability	`explain`	No	*Boolean*	*false*	A boolean flag to specify the scores assigned by the model to each recommended item in the results list. The goal of this feature is to provide quantifiable insights into how the recommendation engine computes the recommendation results, in the form of scores. There are three main types of scores in the ML Explainability feature: • Confidence Score: the overall score of the recommended item • Item Attribute Contribution: the contribution of each attribute in the item and/or user metadata towards the	`{ "explain":false }`

Inference Concept	Inference Parameter	Required	Data Type	Default Value	Description	Sample Payload
					recommendation results • Sequence Attention: the contribution of each item in the clickstream (`items_ls`) towards the recommendation results	
Past item interactions (clickstream)	`items_ls`	No	*List*	None	A list of `item_id` that represents the past item interactions (clickstream) of the user for generating the recommendations. For this parameter to be valid, the input has to correspond to an object entry in the item catalogue training data used to train the model.	`{ "items_ls":[ "item_id1", "item_id2" ] }`

Inference Concept	Inference Parameter	Required	Data Type	Default Value	Description	Sample Payload

String query	`string_queries`	Yes (if `attribute_queries` is empty)	*String*	None	A string input of search terms to be used by the search engine to generate search results.
					
					The input can be in the following two forms:
					
					• Free text query: the default search method. The search engine tries to extract all recognized tokens and return ranked relevant candidates.
					• Exact `item_id`: an `item_id` input can be used to search for a particular item. If the item is available, it's returned as the top result.
						`{ "string_queries" :"red socks for men" }`
						
						`{ "string_queries" :"item_id1" }`
						
						Where "item_id1" corresponds to an object entry in the item catalogue training data used to train the model.

Inference Concept	Inference Parameter	Required	Data Type	Default Value	Description	Sample Payload

Ratings	`ratings`	No	*List*	None	A list of numerical rating scores corresponding to each item in the user's clickstream input (`items_ls`) that is used to assign a rating to each item in the input sequence.
					
					The allowed range for ratings is -1 to 1. Higher rated items contribute more to the output recommendations, while lower rated items represent negative sentiment for the item. A rating of 0 has the same result as omitting the item.
					
					Ratings result in the following behavior:
					
					• A positive rating (between 0 and 1) increases the weight of the corresponding item
					• A negative rating (be-
						<pre>{
						"items_ls":[
						"item_id1",
						"item_id2"
						],
						"ratings":[
						-0.5,
						1
						]
						}</pre>

Inference Concept	Inference Parameter	Required	Data Type	Default Value	Description	Sample Payload
					tween -1 and 0) represents negative sentiment for the corresponding item • A rating of 0 has a similar effect of omitting the corresponding item from the clickstream	
Top-k results	`k`	No	*Integer*	*10*	An integer specifying the number of recommended items to be returned in the inference output. The output contains the top-k recommendation results.	`{ "k":10 }`
User	`user`	No	*String*	None	The `user_id` of the user that is consuming the recommendations. For this parameter to be valid, the input has to correspond to an object entry in the user metadata training data used to train the model.	`{ "user":"user_id1" }`

Related Information

Request [page 133]
Response [page 134]

8.4.1.3.1 Request

Request Example Payload

Use the following `payload` to receive search results with string queries (`string_queries`), optionally with past item interactions (`items_ls`) and/or item attribute queries (`attribute_queries`).

```
{
   "string_queries":"Lord of the Rings",
   "user":"user_id1",
   "items_ls":[
      "item_id1",
      "item_id2"
   ],
   "explain":true,
   "k":3,
   "attribute_queries":{
      "categoricalFeatures":{
         "tags":{
            "values":[
               "trilogy",
               "imdb top 250"
            ],
            "clauses":[
               "must",
               "should"
            ]
         }
      }
   }
}
```

When you provide free-text query ("string_queries"), the search engine tries to extract all recognized tokens and returns three recommendation results ("k":3). The recommendations are personalized by taking into account the user ID ("user":"user_id1") based on the current clickstream ("items_ls":["item_id1","item_id2"]).

In this example, the scope of recommendations is restricted by a set of attributes that acts as a filter ("attribute_queries"). The recommendations **must** have the tag value "trilogy" and **should** have the tag value "imdb top 250".

As ML explainability is enabled ("explain":true), in the response, you'll receive quantifiable insights into how the recommendations engine computes the recommendation results.

8.4.1.3.2 Response

Response Fields

JSON Field	Description
`recommendations`	Top-k recommendation results.
`scores`	Values between 0 and 1. The confidence `scores` represent how certain the model is about the recommendations. The higher the score the more confident the model is that the recommendation is correct. If the score is close to 1, the model is very certain.
`status`	The response is given in one of the following `status`: • `SUCCESS: Trained user. All items are trained.` The user and all items in their history were present in the training dataset. • `ERROR: Invalid user. All items are invalid.` The user and all items in their history were absent from the training dataset. • `WARNING: Cold-Start/Invalid user. All items are Invalid/Cold-Start.` Either the user is invalid and all items are cold start, or items are invalid and the user is cold start or both are cold start. • `WARNING: Trained/Cold-Start/Invalid user. All items are invalid/Items have mixed type (List of Trained/Cold-Start/Invalid).` The user and items provided are a mix of trained, cold start, and invalid.
`status_code`	The response is given in one of the following `status_code`: • **0**: invalid, the user and all items in their history were absent from the training dataset. • **1**: cold start, either the user is invalid and all items are cold start, or items are invalid and the user is cold start or both are cold start. • **2**: mixed, the user and items provided are a mix of trained, cold start, and invalid. • **3**: trained, the user and all items in their history were present in the training dataset.
`item_attribute_contribution`	Each list tells the contribution of the different item attributes towards that recommendation. The values are between 0 and 1. The higher the value the more confident the model is that the recommendation is correct. The first element in the list is always from the item ID.
`relevant_cutoff_index`	The number of recommendations provided by the model.
`item_attributes`	Categorical attributes of items.

Response Example

This response returns top-k recommendations results ("k":3 was used in the request `payload`) .

200 “OK”


```
{
   "recommendations":[
      "9032",
      "386785",
      "710781"
   ],
   "scores":[
      0.3466,
      0.3096,
      0.1519
   ],
   "status":"SUCCESS: Valid query.",
   "status_code":3,
   "item_attribute_contribution":[
      [
         0,
         0.3934,
         0.6066
      ],
      [
         0,
         0.327,
         0.673
      ],
      [
         0,
         0.1698,
         0.8302
      ]
   ],
   "relevant_cutoff_index":3,
   "item_attributes":[
      "domain",
      "description",
      "title"
   ]
}
```


Based on a list of free-form text input (string_queries), and/or a list of attribute values (attribute_queries), a number of recommendation results are returned ("recommendations":["9032","386785","710781"]) along with their scores ("scores": [0.3466, 0.3096, 0.1519]). In this example, the item "9032" has a score of 0.3466, the item "386785" has a score of 0.3096, and the item "710781" has a score of 0.1519.

When you enable ML explainability, you can better understand the "scores". For "item_attribute_contribution", the position of the nested list corresponds to the position of the item ID in the "recommendations" list. The first nested list [0,0.3934,0.6066] is the "item_attribute_contribution" for the first item ID in "recommendations" that is "9032". And the second nested list [0,0.327,0.673], is the "item_attribute_contribution" for the second item ID in "recommendations" that is "386785".

Matrix Representation

Item Attribute Contribution

Recommendations	Domain	Description	Title
9032	0	0.3934	0.6066
386785	0	0.327	0.673
710781	0	0.1698	0.8302

8.4.1.4 User-Affinity Recommendations

Get top-k recommended feature affinity (selected by the `features` parameter) for a batch of users based on their user IDs, and/or a list of past item interactions (`items_ls`).

The item interactions can be customised by assigning ratings to each interaction to adjust its influence on the recommendation results.

Optionally provide `metadata` to either define new users or items unknown by the model (not encountered in the training dataset) or to redefine existing users or items by overwriting them with a different set of attributes and/or weights. Therefore, newly defined items address the cold start scenario, while redefined items can be used to temporarily alter the user or item metadata to introduce bias towards certain attributes (for example, promotions, marketing, prioritization).

For example, in a learning recommendation scenario, if you have “Jungle Book” in your context (clickstream), the recommendation engine returns the following categories: [“Animation”, “Disney”, ...]. And if you have “The Hobbit” in your context (clickstream), the recommendation engine returns the following categories: [“Fantasy”, “Adventure”, ...].

Inference Concept	Inference Parameter	Required	Data Type	Default Value	Description	Sample Payload
Attribute feature filtering	`features`	No	*List*	None	A list of objects that represents categorical feature-types of filters to be applied in the recommendation results. Attribute filtering changes the order of the recommendation results based on the number of applied filters, where items matching more filters are ranked ahead of items matching fewer or no filters. This feature aims to remove certain items from recommendations that don't contain certain attribute values. For this parameter to be valid, the input has to: • Contain valid filterable feature-types (for example categorical features) • Contain valid features in	`{ "features":[ "cat_feature1", "cat_feature2" ] }`

Inference Concept	Inference Parameter	Required	Data Type	Default Value	Description	Sample Payload
					each feature-type that are identical to the features in the item catalogue training data. • Contain valid filter values, which mean they exactly match at least one known item. Invalid values are ignored.	

Inference Concept	Inference Parameter	Required	Data Type	Default Value	Description	Sample Payload
Cold start items and users	`metadata`	No	*Dictionary*	None	A dictionary of cold start items and users, with formatting identical to items in the item_catalogue.json or user_metadata.json files. These items and users will then be referenced in the `items_ls` or `user` parameters during inference. Cold start items refer to items that we want to include in our context (`items_ls` and/or `user`), that weren't part of the training data. For inference purposes, this feature aims to make meaningful recommendations using new items and users as input, or to modify the metadata of existing items or users temporarily to introduce bias towards certain affinities.	`{ "metadata":{ "items": { "coldstart_item_id1":{ "categoricalFeatures":{ "cat_feature1":{ "values":[ "v1", "v2" ] } , "cat_feature2":{ "..." } }, "textFeatures"{ "..." } }`

Inference Concept	Inference Parameter	Required	Data Type	Default Value	Description	Sample Payload

						<pre>},
						
						"coldstart_item_id2":{
						
						"..."
						}
						},
						"users":{
						
						"coldstart_user_id1":{
						
						"categoricalFeatures":{
						
						"cat_feature1":{
						
						"values":[
						
						"v1",
						
						"v2"
						]
						}
						,
						
						"cat_feature2":{
						
						"..."
						}
						},</pre>

Inference Concept	Inference Parameter	Required	Data Type	Default Value	Description	Sample Payload

						<pre>"textFeatures"{
						
						"..."
						}
						},
						
						"coldstart_user_id2":{
						
						"..."
						}
						}
						}
						}</pre>

Inference Concept	Inference Parameter	Required	Data Type	Default Value	Description	Sample Payload
Past item interactions (clickstream)	`items_ls`	Yes (if `user` is empty)	*List*	None	A list of `item_id` that represents the past item interactions (clickstream) of the user for generating the recommendations. For this parameter to be valid, the input has to: • Correspond to an object entry in the item catalogue training data used to train the model, or. • To be provided as an entry in the metadata parameter as a colds tart item.	`{ "items_ls":[ "item_id1", "item_id2" ] }`

Inference Concept	Inference Parameter	Required	Data Type	Default Value	Description	Sample Payload

Ratings	`ratings`	No	*List*	None	A list of numerical rating scores corresponding to each item in the user's clickstream input (`items_ls`) that is used to assign a rating to each item in the input sequence.
					
					The allowed range for ratings is -1 to 1. Higher rated items contribute more to the output recommendations, while lower rated items represent negative sentiment for the item. A rating of 0 has the same result as omitting the item.
					
					Ratings result in the following behavior:
					
					• A positive rating (between 0 and 1) increases the weight of the corresponding item
					• A negative rating (be-
						`{`
						`"items_ls":[`
						`"item_id1",`
						`"item_id2"`
						`],`
						`"ratings":[`
						`-0.5,`
						`1`
						`]`
						`}`

Inference Concept	Inference Parameter	Required	Data Type	Default Value	Description	Sample Payload
					tween -1 and 0) represents negative sentiment for the corresponding item • A rating of 0 has a similar effect of omitting the corresponding item from the clickstream	
Top-k results	k	No	*Integer*	*10*	An integer specifying the number of recommended items to be returned in the inference output. The output contains the top-k recommendation results.	`{ "k":10 }`

Inference Concept	Inference Parameter	Required	Data Type	Default Value	Description	Sample Payload
User	`user`	Yes (if `items_ls` is empty)	*String*	None	The `user_id` of the user that is consuming the recommendations. For this parameter to be valid, the input has to: • Correspond to an object entry in the user metadata training data used to train the model, or. • To be provided as an entry in the metadata parameter as a cold start user.	`{ "user":"user_id1" }`

Related Information

Request [page 146]
Response [page 146]

8.4.1.4.1 Request

Request Example Payload

Use the following `payload` to receive recommendations with attribute feature filtering (`features`) but without cold start items and users (`metadata`).

```
{
   "user":"user_id1",
   "items_ls":[
      "item_id1",
      "item_id2"
   ],
   "explain":true,
   "k":3,
   "features":[
      "domain"
   ]
}
```

The endpoint returns three recommendation results ("k":3) based on the current clickstream ("items_ls": ["item_id1", "item_id2"]) of the user with user_id1 ("user":"user_id1").

The attribute feature filtering ("features":["domain"]) only recommends items that contain the "domain" feature.

As ML explainability is enabled ("explain":true), in the response, you'll receive quantifiable insights into how the recommendations engine computes the recommendation results.

8.4.1.4.2 Response

Response Fields

JSON Field	Description
`recommendations`	Top-k recommendation results.
`scores`	Values between 0 and 1. The confidence `scores` represent how certain the model is about the recommendations. The higher the score the more confident the model is that the recommendation is correct. If the score is close to 1, the model is very certain.

JSON Field	Description
`status`	The response is given in one of the following `status`: • `SUCCESS: Trained user. All items are trained.` The user and all items in their history were present in the training dataset. • `ERROR: Invalid user. All items are invalid.` The user and all items in their history were absent from the training dataset. • `WARNING: Cold-Start/Invalid user. All items are Invalid/Cold-Start.` Either the user is invalid and all items are cold start, or items are invalid and the user is cold start or both are cold start. • `WARNING: Trained/Cold-Start/Invalid user. All items are invalid/Items have mixed type (List of Trained/Cold-Start/Invalid).` The user and items provided are a mix of trained, cold start, and invalid.
`status_code`	The response is given in one of the following `status_code`: • **0**: invalid, the user and all items in their history were absent from the training dataset. • **1**: cold start, either the user is invalid and all items are cold start, or items are invalid and the user is cold start or both are cold start. • **2**: mixed, the user and items provided are a mix of trained, cold start, and invalid. • **3**: trained, the user and all items in their history were present in the training dataset.

Response Example

This response returns top-k recommendations results ("k":3 was used in the request `payload`) .

200 “OK”

```
{
   "recommendations":{
      "domain":[
         "AGCS",
         "AZL",
         "AWP"
      ]
   },
   "scores":{
      "domain":[
         0.8302,
         0.3965,
         0.2559
      ]
   },
   "status_code":3,
   "status":"SUCCESS: Trained user. All items are trained."
}
```

Based on the user ID ("user":"user_id1") and/or past item interactions ("items_ls":["item_id1","item_id2"]) from the request payload, a number of recommended feature affinities are returned ("recommendations":

{"domain":["AGCS","AZL","AWP"]) along with their scores ("scores":{"domain":[0.8302,0.3965,0.2559]). In this example, the feature "AGCS" has an affinity score of 0.8302, the feature "AZL" has a score of 0.3965, and the feature "AWP" has a score of 0.2559.

Matrix Representation

Item Attribute Contribution

Recommendations	Affinity Score
"AGCS"	0.8302
"AZL"	0.3965
"AWP"	0.2559

8.4.1.5 Offline (Scenario) Recommendations

Scenarios are offline (batch) jobs that can be executed with a trained Personalized Recommendation model that has already been trained successfully during the training job.

With offline (scenario) recommendations, you can get downloadable recommendations without the need to deploy an online serving instance. In addition, offline recommendations also enable customization of inference parameters similar to online recommendations, which you can download or receive after the training job is complete.

Personalized Recommendation supports the following offline (scenario) recommendations types:

Scenario Type	How recommendations are generated
Offline (Scenario) Next-Item Recommendations [page 149]	Recommendations are generated per unique user provided in the training Clickstream [page 26] data, User Metadata [page 36] and Scenario Configurations and Data [page 42]. It requires at least one user ID with valid item interactions. If clickstream is provided, recommendations are generated using the item interactions provided in the clickstream. If clickstream isn't provided, `items_ls` parameter must either be broadcasted or provided for each unique user ID in user metadata and scenario configurations and data.

Scenario Type	How recommendations are generated
Offline (Scenario) Similar-Item Recommendations [page 150]	Recommendations are generated per unique item provided in the training Item Catalogue [page 30] and Scenario Configurations and Data [page 42]. It requires at least one item ID with valid item metadata. If item catalogue isn't provided, recommendations are generated for each unique item ID provided in the scenario configurations and data.
Offline (Scenario) Propensity Recommendations [page 151]	Propensity recommendations provides `k` user recommendations for an item given an item ID (`item`). This recommendation type supports the ML Explainability [page 152] (`explain`) feature to provide insights into the recommendation results by breaking the contribution of each user with respect to the recommendations provided. This recommendation type can be useful for understanding why certain users were recommended and ways to improve the recommendation results. For cold-start item and user scenarios, item and/or user metadata may also be passed during inference to either define new users/items unknown to the model (not provided in training data) or to redefine existing users/items by overwriting them with a different set of attributes and/or weights.

Related Information

Training Data Requirements [page 24]

8.4.1.5.1 Offline (Scenario) Next-Item Recommendations

Offline next-item recommendations are a set of Next-Item Recommendations [page 93] generated per unique user provided in the training clickstream data, user metadata and scenario configurations and data, which can then be downloaded via batch recommendations endpoint once a training job has completed successfully.

Offline recommendations are generated after a successful model training job, and are customisable via training job options (scenario configurations). These offline recommendations can then be used for batch processes, getting customised recommendation results for offline use cases, or as a fallback in the event of training failure or service downtime.

The main difference between offline and online recommendations is that offline recommendations are only generated during training job executions, while online recommendations are generated on demand with a serving instance.

→ Tip

If the `unavailable` flag is set to *true* in item catalogue training data, offline recommendations results aren't provided for that item ID.

To get updated recommendations, rerun the training job execution.

To get offline next-item recommendations, you need to create up to 4 scenario configurations and data files with `scenario_type` `next-items`. The scenario configurations and data file also allows you to customize the next-item recommendation results by providing scenario-level configurations and user-level data to be used for the batch inference. The scenarios are executed during the training job.

To receive offline recommendation results you can:

- **Download the results via batch recommendations endpoint**: Once training is successful, the recommendations generated based on the scenario configurations and data files can be consumed via the endpoint `GET /standard/rs/v1/tenants/{tenant}/recommendations/batch-results` (see Inference [page 91]). To access the offline recommendations, use **`scenario_name`** as the `filter` parameter.
- **Access object store to see the results**: Once training is successful, the recommendations generated based on the scenario configurations and data files are stored in the customer's object store with the same name as the scenario configurations and data file (**`<scenario_name>.scenario.json`**) that was provided. For example, if the scenario configurations and data file is called: **`test_scen_1.scenario.json`**, the offline recommendation results file is going to be called: `test_scen_1-page1.txt.gz`.

  ⓘ Note

  To upload the results to object store integration, the training job must've been submitted via the endpoint `POST /standard/rs/v1/tenants/{tenant}/jobs` (see Training [page 22]).

Related Information

Scenario Parameters for Offline Next-Item Recommendations [page 46]

8.4.1.5.2 Offline (Scenario) Similar-Item Recommendations

Offline similar-item recommendations are a set of Similar-Item Recommendations [page 112] generated per unique item provided in the training item catalogue and scenario configurations and data, which can then be downloaded via batch recommendations endpoint once a training job has completed successfully.

Offline recommendations are generated after a successful model training job, and are customisable via training job options (scenario configurations). These offline recommendations can then be used for certain cold start scenarios, or as a fallback in the event of training failure or service downtime.

The main difference between offline and online recommendations is that offline recommendations are only generated during training job executions, while online recommendations are generated on demand with a serving instance.

> → Tip
>
> If the `unavailable` flag is set to *true* in user metadata training data, offline recommendations results aren't provided.
>
> If `keep_user_history` flag is set to *false* during training job submission, the item interaction history for all users aren't stored.
>
> To get updated recommendations, rerun the training job execution.

To get offline similar-item recommendations, you need to create up to 4 scenario configurations and data files with `scenario_type similar-items`. The scenario configurations and data file also allows you to customize the similar-item recommendation results by providing scenario-level configurations and item-level data to be used for the batch inference. The scenarios are executed during the training job.

To receive offline recommendation results you can:

- **Download the results via batch recommendations endpoint**: Once training is successful, the recommendations generated based on the scenario configurations and data files can be consumed via the endpoint `GET /standard/rs/v1/tenants/{tenant}/recommendations/batch-results` (see Inference [page 91]). To access the offline recommendations, use **`scenario_name`** as the `filter` parameter.
- **Access object store to see the results**: Once training is successful, the recommendations generated based on the scenario configurations and data files are stored in the customer's object store with the same name as the scenario configurations and data file (**`<scenario_name>.scenario.json`**) that was provided. For example, if the scenario configurations and data file is called: **`test_scen_1.scenario.json`**, the offline recommendation results file is going to be called: `test_scen_1-page1.txt.gz`.

  > ⓘ Note
  >
  > To upload the results to object store integration, the training job must've been submitted via the endpoint `POST /standard/rs/v1/tenants/{tenant}/jobs` (see Training [page 22]).

Related Information

Scenario Parameters for Offline Similar-Item Recommendations [page 60]

8.4.1.5.3 Offline (Scenario) Propensity Recommendations

Offline propensity recommendations provide recommendations that include users with the highest affinity to a given item. This recommendation feature is useful for identifying a subset of users that are likely to favor a particular item, and can be used to create targeted outreach opportunities.

For example, if a business wants to promote item A, this recommendation type recommends users U1, U2, U3 and so on, that are most likely to purchase item A. The scores are based on the item properties of item A, as well as users whose likely next-item interactions are of close proximity to item A.

This recommendation type can be useful in enabling several use-cases where recommendations can narrow down item preference (of an existing or new item, or cold start item) to a subset of users to streamline outreach workflows. For example, targeted user segment for high-value sales outreach.

Related Information

Scenario Parameters for Offline Propensity Recommendations [page 73]

8.4.2 ML Explainability

Get top-k recommended items for users based on their users' clickstream history (`item_sequence`).

Gets the contribution scores for each recommended item by setting parameter `"explain":true` for training endpoint (see Training Job Options [page 79]) and then the inference endpoints (see Inference Options [page 93]). The ML explainability feature comprises three types of contribution scores justifying each item recommendation from different angles:

- Item attribute contribution is which item attributes of item X led to it being recommended.
- Sequence attention (only for next-item recommendations) is which past item interactions were most relevant for this recommendation.
- User attribute contribution (only for next-item recommendations) is which user attributes were most important when recommending item X.

Detailed explanations on each concept can be found below.

ML explainability is the concept of interpreting, or 'explaining', the output of a machine learning (or specifically 'deep learning') models in a human-interpretable manner. This feature allows us to interpret and understand the contributing factors leading to the output of a machine learning solution, by that providing model transparency and resolving the paradigm that machine learning solutions are proverbial 'black-box' functions. The goal of ML explainability in Personalized Recommendation is to build confidence and trust in our recommendations, while also allowing users to make sense of the recommendations based on the provided context and `metadata`.

ML explainability can be used to elevate the user experience of recommendation users by providing additional context to the recommendations, which can be helpful in certain scenarios such as learning or content recommendations. In addition, ML explainability allows administrative users (for example, merchandisers, content curators, etc.) to evaluate the recommendation results. It can also be used in coordination with our other features, such as Attribute Boosting and Feature Importance, to better fine-tune the customizable parameters and curation of training data.

Examples:

- Next-Item recommendations recommend item D because I have previously viewed item A, B, C, and because I have a preference towards item categories 1, 2 and 3.

- While using Attribute Boosting, I can view the difference in contribution scores while using different boosting weights, so that I don't under or over boost a certain attribute.
- With Feature Importance, I can better understand the training data and identify potential trends in attribute contributions towards recommendations.

ML explainability is displayed for each of the recommended movies. In this case, we're looking at the ML explainability of the highest ranked recommended movie, Harry Potter and the Sorcerer's Stone. From it, the user can see that from the clickstream history, Harry Potter and the Chamber of Secrets contributes the most to this particular recommendation.

- Confidence Score: This recommendation has a confidence score higher than 0.7 (70%).
- Item Attribute Contribution: From the item attributes, the weightage of contribution by each of the attribute, with the ID attribute (0.52) contributing the most to this particular recommendation.
- Sequence Attention: From the clickstream history, the weightage of contribution by each of the items, with Harry Potter and the Chamber of Secrets (0.87) contributing the most to this particular recommendation.

Explainability Concept	Inference Option	Description	Sample Response

Confidence Score	All Inference Options [page 93]	The overall score of an item. Recommendations are ranked based on this score.
		
		Each item in `scores` corresponds to an item in the `recommendations` list.
			`{`
			`"recommendations": ["4967", "4603", "4969"],`
			`"scores": [0.5481, 0.5221, 0.4597],`
			`...`
			`}`
			
			Where top-ranked item "4967" has a score of 0.5481, and the third-ranked item "4969" has a score of 0.4597.

Explainability Concept	Inference Option	Description	Sample Response

Item Attribute Contribution	All Inference Options [page 93]	The contribution of each item's text `text` and `scores` categorical attributes (provided in item metadata) towards the final score of the item (see User Metadata [page 36]).
		
		Each list in `item_attribute_contribution` corresponds to an item in the `recommendations` list, and each item within each list (in `item_attribute_contribution`) corresponds to an attribute in the `item_attributes`item_attributes list, as shown in the example provided.
		
		In Item Attribute Contribution, `id` is an added component that represents the contextual contribution (relationship between two items) towards the recommendation as provided in the `clickstream` data (see Clickstream [page 26]).
		
		The attribute scores are normalized and refer to the respective attribute in the `item_attributes` list. The attribute scores can be interpreted as contribution percentages to the recommendation score for the respective item.
			<pre>{
			
			"recommendations" : ["4967", "4603", "4969"],
			
			"item_attributes" : ["id", "category", "description", ..
			.],
			
			"item_attribute_contribution": [
			[0.6366, 0.245, 0.0309, ...],
			# Belongs to "4967"
			[0.6352, 0.2572, 0.0228, ...], # Belongs to "4603"
			[0.5475, 0.2921, 0.0702, ...] # Belongs to "4969"
			],
			...
			}</pre>
			Where top-ranked item "4967" has the item attribute contributions ID (0.6366), category (0.245), and description (0.0309). And the third-ranked item "4969" has the item attribute contributions ID (0.5475), category (0.2921), and description (0.0702).

Explainability Concept	Inference Option	Description	Sample Response

Sequence Attention	Next-Item Recommendations [page 93] only	The contribution of the user's `current context` (provided in `items_ls` during inference) towards the final score of the item
		
		Each list in `sequence_attention`corresponds to an item in the `recommendations` list, and each item within each list (in `sequence_attention`) corresponds to an item in the `item_sequence` list, as shown in the example provided.
		
		The attribute scores are normalized and refer to the respective attribute in the `item_sequence` list. The attribute scores can be interpreted as contribution percentages to the recommendation score for the respective item.
			<pre>{
			
			"recommendations"
			: ["4967",
			"4603", "4969"],
			
			"item_sequence":
			["4939",
			"4585"],
			
			"sequence_attenti
			on": [
			[0.5791,
			0.4209], #
			Belongs to "4967"
			[0.3155,
			0.6845], #
			Belongs to "4603"
			[0.4333,
			0.5667] #
			Belongs to "4969"
			],
			...
			}</pre>
			There are currently two items [item 4939] and [item 4585] in the user's context (item_ls).
			
			The top-ranked item "4967" has the sequence attention contributions [item 4939] (0.5791) and [item 4585] (0.4209) and the third-ranked item "4969" has the sequence attention contributions [item 4939] (0.4333) and [item 4585] (0.5667).

Explainability Concept	Inference Option	Description	Sample Response

User Attribute Contribution	Next-Item Recommendations [page 93] only	The contribution of the user's `text` and `categorical` attributes (provided in `user metadata`) towards the final score of the item (see Item Catalogue [page 30]).
		
		Each list in `user_attribute_contribution` corresponds to an item in the `recommendations` list, and each item within each list (in `user_attribute_contribution`) corresponds to an attribute in the user_attributes list, as shown in the example provided.
		
		The attribute scores are normalized and refer to the respective attribute in the `user_attributes` list. The attribute scores can be interpreted as contribution percentages to the recommendation score for the respective item.
			`{`
			
			`"recommendations" : ["4967", "4603", "4969"],`
			
			`"user_attributes" : [ "age", "gender", ...],`
			
			`"user_attribute_contribution": [`
			`[ 0.6074, 0.1973, ...],`
			`# Belongs to "4967"`
			`[ 0.6712, 0.1422, ...],`
			`# Belongs to "4603"`
			`[ 0.452, 0.3721, ...]`
			`# Belongs to "4969"`
			`],`
			`...`
			`}`
			
			Where top-ranked item "4967" has the user attribute contributions age (0.6074) and gender (0.1973). And the third-ranked item "4969" has the user attribute contributions age (0.452) and gender (0.3721).

8.5 Serving

Use the Serving APIs to manage online model serving instances and get the serving statuses of their models. Consume the endpoints listed below, only after obtaining a successful training job.

Request

Base URL: url value from outside the uaa section of the service key

URL Path Extension: `/doc`

Serving APIs

HTTP Method	URL Endpoint Path	Description
GET	`/standard/rs/v1/tenants/{tenant}/servings`	Get the deployment and serving status of a model serving instance, identified by its subaccount, tenant, and site. → Remember A model must be successfully trained before it can be served.
DELETE	`/standard/rs/v1/tenants/{tenant}/servings`	Delete a model serving instance of a tenant (site). → Remember Once a model serving instance is deleted, it can be restarted via the **PUT** method with the same path.
PUT	`/standard/rs/v1/tenants/{tenant}/servings`	Deploy a model serving instance of a tenant (site). → Remember The deployment and serving status of a model serving instance can be obtained via the **GET** method with the same path.

8.6 Common Status and Error Codes

Code	Reason
200	Request was successful.
400	Bad request. Process could not be submitted or completed; possibly due to parameter error, bad request syntax, or invalid request message framing.
401	Unauthorized. No token, bad or invalid token.

Code	Reason
404	Not found. URL used is incorrect.
409	Conflict. Request cannot be processed due to a conflict in the current state of the resource, for example, when you have already submitted a job that is not finished, and try to submit another one.
425	Results not ready. Inference results are still pending.
500	Internal server error. Process could not be submitted or completed; possibly due to an internal error.
502	Bad gateway. The server, while acting as a gateway or proxy, received an invalid response from the upstream server it accessed in attempting to fulfill the request.

9 Technical Constraints

All Personalized Recommendation endpoints exposed to the end user have strict technical limits. See details in the following table.

> **Note**
>
> The technical limits listed here are relevant only to users of the **Standard** service plan for enterprise accounts. See Service Plans [page 14].

Content Type	Variable	Minimum Limit
Clickstream	Number of valid events	1000
	Number of unique users	100
	Number of trained items **→ Tip** Refer to the unique items clicked by users to be used for training.	100
	Clickstream data size	None
Item Catalogue	Number of catalogue items	100
	Number of categorical and text attributes	0
User Metadata	Number of metadata users	5
	Number of categorical and text attributes	0

> **Restriction**
>
> When using the `POST /standard/rs/v1/tenants/{tenant}/jobs/file-upload` endpoint to upload the ZIP training data file, the maximum file size limit is 1 GB. See Training [page 22].

Related Information

Free Tier Option Technical Constraints [page 160]

9.1 Free Tier Option Technical Constraints

When using the free tier option for Personalized Recommendation, be aware of the following technical limits:

> **Note**
>
> The technical limits listed here are relevant only to users of the **Free** service plan for enterprise accounts. See Service Plans [page 14].

Variable	Maximum Limit
Total number of trainings per month **Note** Only successfully completed trainings are considered.	2
Total number of inference requests per month	1000

10 Security

Get an overview on the security information that applies to Personalized Recommendation. Learn about the main security aspects of the service and its components.

General Information

The Personalized Recommendation service provides a set of RESTful application programming interfaces (APIs) over which client applications can directly communicate with the service, in order to provide personalized recommendations from the machine learning model trained with customer data. All communication to the APIs is secured via the HTTPS protocol. The Personalized Recommendation service is a pure API-based service and provides no graphical user interface and has no dedicated front end.

Related Information

User Administration, Authentication, and Authorization [page 161]
Data Protection and Privacy [page 162]
Auditing and Logging Information [page 164]
Front-End Security [page 167]

10.1 User Administration, Authentication, and Authorization

Introduction

Cloud Foundry

For the Personalized Recommendation service, the standard user authentication and authorization mechanisms provided by SAP Business Technology Platform (SAP BTP) for Cloud Foundry is used. Following the standard mechanism of Cloud Foundry, the service consumer can create an instance of the Personalized Recommendation service and then generate credentials to communicate with the service instance (see Using Services in the Cloud Foundry Environment in the SAP BTP documentation for details on this process).

The credentials enable the user to retrieve a JSON Web Token (JWT), which is necessary for the secure communication between any client and the service. Overall, the communication with the service is secured by the OAuth 2.0 protocol. For more information on this topic, see Security in the SAP BTP documentation.

User Administration and Provisioning

The application does not manage or provision users.

10.2 Data Protection and Privacy

Introduction

Data protection is associated with numerous legal requirements and privacy concerns. In addition to compliance with general data privacy acts, it is necessary to consider compliance with industry-specific legislation in different countries/regions. This section describes the specific features and functions that SAP provides to support compliance with the relevant legal requirements and data privacy.

This section and any other sections in this Security Guide do not give any advice on whether these features and functions are the best method to support company, industry, regional or country/region-specific requirements. Furthermore, this guide does not give any advice or recommendations with regard to additional features that would be required in a particular environment; decisions related to data protection must be made on a case-by-case basis and under consideration of the given system landscape and the applicable legal requirements.

> ⓘ Note
>
> SAP software supports data protection by providing security features and specific data protection-relevant functions such as functions for the simplified blocking and deletion of personal data. SAP does not provide legal advice in any form. The definitions and other terms used in this document are not taken from any given legal source.

The Personalized Recommendation service generally requires the following type of data:

Data	Purpose
Data for training	Consists of user item interaction data, item attribute and user metadata, that are used in the deep learning training process to create a machine learning model that can provide similar or next items for user to interact with. ⚠ Caution The Personalized Recommendation service doesn't require (sensitive) personal data for training. Personal data is subject to the data protection laws applicable in specific countries/regions. We therefore recommend that you don't include personal data in your user metadata training data file.

Data	Purpose
Data for inference	Consists of user recent interaction (view, click, rate in sequence), new item metadata, filtering and boosting conditions, which are submitted by a user in order to receive machine learning predictions. Inference data is processed immediately, it is not stored temporarily anywhere.

Read Access Logging

The data used by the Personalized Recommendation service is controlled and managed by the consuming application or customer that calls the service APIs. However, the service does not have any means to verify whether the data uploaded to the service contains any personal information. Therefore, the Personalized Recommendation service does not support logging of read access to (sensitive) personal data.

Information Report

The data used by the Personalized Recommendation service is controlled and managed by the consuming application or customer that calls the service APIs. The Personalized Recommendation service does not have any means to verify whether the data uploaded to the service contains any personal information. Therefore, the service does not provide any means to retrieve personal data of specific individuals. It is recommended that the consuming application or customer, that uses the service, provides personal data reports to its users about the data being stored and transferred to the Personalized Recommendation service for processing.

Deletion of Personal Data

The Personalized Recommendation service has no mechanism for determining the type of data it processes. This means that the service can't determine whether this data includes personal data, which is subject to the data protection laws applicable in specific countries/regions. Therefore, no dedicated functionality for the deletion of personal data is available. You can delete data uploaded to or created by the service any time. See the table below for more details on how to delete data for training and data for inference.

Data	Deletion
Data for training	Training data is encrypted and stored in the cloud storage during training process. After training is completed, the data is deleted permanently, unless customer specifically sets a flag for support purpose. When using the Personalized Recommendation service, you can use endpoints to delete trained models. When you delete your Personalized Recommendation service instance, all the uploaded data is deleted.
Data for inference	Inference data is processed immediately, it is not stored temporarily anywhere.

Change Log

The data used by the Personalized Recommendation service is controlled and managed by the consuming application or customer that calls the service APIs. The service itself does not allow any change to the content of the uploaded data. Therefore, the Personalized Recommendation service does not support logging of data change.

Consent

According to Personal Data Processing Agreement for SAP Cloud Services, SAP acts as data processor. Thus, customers are responsible for obtaining relevant consent to process personal data, including when applicable approval by controllers to use SAP as a processor.

10.3 Auditing and Logging Information

Here you can find a list of the security events that are logged by the Personalized Recommendation service.

Security events written in audit logs

Event grouping	What events are logged	How to identify related log events
Start and authorization related events	Application start	"Application start" • User ID • Source IP

Event grouping	What events are logged	How to identify related log events
	Authorization success	"Authorized access" • User ID • Source IP
	Authorization failure	"Unauthorized access attempted" • User ID • Source IP • Message, if any
Tenant related events	Tenant deletion	"Tenant data deleted" • User ID • Source IP • Tenant • Message, if any
	Site deletion	"Site deleted" • User ID • Source IP • Tenant • Site • Message, if any
	Get tenant files count	"Get tenant files successful" • User ID • Source IP • Tenant • Message, if any
Training related events	Training data upload / Job submission	"training_request" • Customer ID • Tenant ID • Site • File List • Job Options • is_successful
	Training data deletion	"Training data deleted" • Customer ID • Tenant ID • Site

Event grouping	What events are logged	How to identify related log events
	Training job (model creation)	"model_creation" • Tenant ID • Site • Message (Dep. Exc. Fw) • is_successful
	Get training status	"get_training_job_status" • Tenant ID • Site • message • data • is_successful
Model lifecycle related events	Batch inference result download	"Batch inference downloaded" • Customer ID • Tenant ID • Site • Result Type • Filter • Page
	Persistent metadata data upload	"Metadata delta uploaded" • Customer ID • Tenant ID • Site • File List • Receipt (Validation)
	Model download	"Model downloaded" • Customer ID • Tenant ID • Site • File List

Related Information

Audit Logging in the Cloud Foundry Environment

10.4 Front-End Security

The Personalized Recommendation service does not have any user interface component. All functionalities are delivered via web services, JSON over HTTPS.

The service is a backend-only service component and not designed to be invoked by a web browser. Additionally, outputs returned by the service depend on the data submitted to it. Therefore, a consumer of the service should sanitize the data submitted to the service and returned by it to avoid script injection attack.

11 Accessibility Features in Personalized Recommendation

To optimize your experience of Personalized Recommendation, SAP Business Technology Platform (SAP BTP) provides features and settings that help you use the software efficiently.

> **Note**
>
> Personalized Recommendation runs on the SAP BTP cockpit. For this reason, the accessibility features for SAP BTP cockpit apply. For more information, see the accessibility documentation for SAP BTP cockpit on SAP Help Portal at Accessibility Features in SAP BTP Cockpit.

For more information on keyboard handling for SAPUI5 UI elements and screen-reader support for SAPUI5 controls, see Accessibility for End Users.

12 Monitoring and Troubleshooting

Find out how to get support.

Getting Support

If you encounter an issue with this service, we recommend following the procedure below.

Check Platform Status

Check the availability of the platform at SAP Trust Center.

For more information about selected platform incidents, see Root Cause Analyses.

Check Guided Answers

In the SAP Support Portal, check the Guided Answers section for SAP Business Technology Platform. You can find solutions for general platform issues as well as for specific services there.

Contact SAP Support

You can report an incident or error through the SAP Support Portal. For more information, see Getting Support.

Use the following component for your incident:

Component Name	Component Description
CA-ML-PR	Personalized Recommendation

When submitting the incident, we recommend including the following information:

- Region information (Canary, EU10, US10, for example)
- Subaccount technical name
- The URL of the page where the incident or error occurs
- The steps or clicks used to replicate the error
- Screenshots, videos, or the code entered

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