# Package ‘previsionio’

October 14, 2022

**Type**  Package  

**Title**  'Prevision.io' R SDK  

**Version**  11.7.0  

**Description**  For working with the 'Prevision.io' AI model management platform's API [https://prevision.io/](https://prevision.io/).  

**License**  MIT + file LICENSE  

**Encoding**  UTF-8  

**RoxygenNote**  7.1.2  

**Imports**  data.table, futile.logger, httr, jsonlite, Metrics, graphics,  

stats, utils, XML, magrittr, plotly  

**Suggests**  testthat  

**Depends**  R (>= 2.10)  

**NeedsCompilation**  no  

**Author**  Florian Laroumagne [aut, cre],  

Prevision.io Inc [cph]  

**Maintainer**  Florian Laroumagne <florian.laroumagne@prevision.io>  

**Repository**  CRAN  

**Date/Publication**  2022-03-09 13:50:02 UTC  

---

## R topics documented:

- `create_connector` .................................................. 4  
- `create_contact_point` ........................................... 5  
- `create_dataframe_from_dataset` ............................... 5  
- `create_dataset_embedding` ....................................... 6  
- `create_dataset_from_dataframe` ............................... 6  
- `create_dataset_from_datasource` .............................. 7  
- `create_dataset_from_file` ..................................... 7  
- `create_datasource` ................................................ 8  
- `create_deployment_api_key` ................................... 9  
- `create_deployment_model` ..................................... 9
topics documented:
create_deployment_predictions .................................................. 10
create_experiment ................................................................. 11
create_experiment_version ....................................................... 12
create_export ................................................................. 15
create_exporter ............................................................. 15
create_folder ............................................................... 16
create_pipeline_trigger ....................................................... 17
create_prediction .............................................................. 17
create_project ................................................................. 18
create_project_user ............................................................ 19
delete_connector ............................................................ 20
delete_contact_point .......................................................... 20
delete_dataset ............................................................... 21
delete_datasource ............................................................. 21
delete_deployment ............................................................ 22
delete_experiment .............................................................. 22
delete_exporter ............................................................... 23
delete_folder ................................................................. 23
delete_pipeline .............................................................. 24
delete_prediction .............................................................. 24
delete_project ................................................................. 25
delete_project_user ............................................................ 25
get_best_model_id ............................................................. 26
get_connectors ................................................................. 26
get_connector_id_from_name .................................................. 27
get_connector_info ............................................................. 27
get_contact_points ............................................................. 28
get_contact_point_info ........................................................ 28
get_datasets ................................................................. 29
get_dataset_embedding ........................................................ 29
get_dataset_head ............................................................. 30
get_dataset_id_from_name ................................................... 30
get_dataset_info ............................................................. 31
get_datasources .............................................................. 31
get_datasource_id_from_name .............................................. 32
get_datasource_info .......................................................... 32
get_deployments .............................................................. 33
get_deployment_alerts ......................................................... 33
get_deployment_alert_id_from_name ....................................... 34
get_deployment_alert_info .................................................... 34
get_deployment_api_keys ....................................................... 35
get_deployment_app_logs .................................................... 35
get_deployment_id_from_name .............................................. 36
get_deployment_info .......................................................... 36
get_deployment_predictions ................................................... 37
get_deployment_prediction_info ............................................ 37
get_deployment_usage ........................................................ 38
get_experiments .............................................................. 38
topics documented:

get_experiment_id_from_name ........................................... 39
get_experiment_info ..................................................... 39
get_experiment_version_features ....................................... 40
get_experiment_version_id .............................................. 40
get_experiment_version_info ........................................... 41
get_experiment_version_models ....................................... 41
get_experiment_version_predictions ................................. 42
get_exporters ............................................................. 42
get_exporter_exports .................................................... 43
get_exporter_id_from_name ............................................. 43
get_exporter_info ........................................................ 44
get_features_infos ...................................................... 44
get_folder ................................................................. 45
get_folders ................................................................ 45
get_folder_id_from_name ............................................... 46
get_model_cv ............................................................... 46
get_model_feature_importance ....................................... 47
get_model_hyperparameters ......................................... 47
get_model_infos .......................................................... 48
get_pipelines .............................................................. 48
get_pipeline_id_from_name ........................................... 49
get_pipeline_info ........................................................ 49
get_prediction ............................................................. 50
get_prediction_infos .................................................... 50
get_projects ............................................................... 51
get_project_id_from_name ............................................. 51
get_project_info .......................................................... 52
get_project_users ....................................................... 52
helper_cv_classif_analysis ........................................... 53
helper_drift_analysis .................................................... 53
helper_optimal_prediction ........................................... 54
helper_plot_classif_analysis ....................................... 55
pause_experiment_version ............................................. 56
pio_download ............................................................... 56
pio_init ....................................................................... 57
pio_list_to_df ............................................................ 57
pio_request ............................................................... 58
resume_experiment_version ........................................... 58
stop_experiment_version ............................................... 59
test_connector ........................................................... 59
test_contact_point ....................................................... 60
test_datasource .......................................................... 60
test_deployment_type ................................................... 61
test_pipeline_type ........................................................ 61
update_experiment_version_description ....................... 62
update_project_user_role ............................................. 62

Index ................................................................. 63
create_connector

Create a new connector of a supported type (among: "SQL", "FTP", "SFTP", "S3", "GCP"). If check_if_exist is enabled, the function will check if a connector with the same name already exists. If yes, it will return a message and the information of the existing connector instead of creating a new one.

Description

Create a new connector of a supported type (among: "SQL", "FTP", "SFTP", "S3", "GCP"). If check_if_exist is enabled, the function will check if a connector with the same name already exists. If yes, it will return a message and the information of the existing connector instead of creating a new one.

Usage

create_connector(
  project_id,
  type,
  name,
  host,
  port,
  username,
  password,
  google_credentials = NULL,
  check_if_exist = FALSE
)

Arguments

project_id    id of the project, can be obtained with get_projects().
type          connector type.
name          connector name.
host          connector host.
port          connector port.
username      connector username.
password      connector password.
google_credentials     google credentials JSON (for GCP only).
check_if_exist     boolean (FALSE by default). If TRUE, makes extra checks to see if a connector with the same name is already existing.

Value

list - parsed content of the connector.
create_contact_point

Create a new contact point of a supported type (among: "email", "slack").

Description

Create a new contact point of a supported type (among: "email", "slack").

Usage

create_contact_point(
    project_id,
    type,
    name,
    addresses = NULL,
    webhook_url = NULL
)

Arguments

project_id id of the project, can be obtained with get_projects().
type contact point type among "email" or "slack".
name contact point name.
addresses contact point addresses.
webhook_url contact point webhook_url.

Value

list - parsed content of the contact point.

create_dataframe_from_dataset

Create a dataframe from a dataset_id.

Description

Create a dataframe from a dataset_id.

Usage

create_dataframe_from_dataset(dataset_id)

Arguments

dataset_id dataset id.
create_dataset_from_dataframe

Value
data.frame - a R dataframe matching the dataset.

create_dataset_embedding

Create a dataset embedding from a dataset_id.

Description
Create a dataset embedding from a dataset_id.

Usage
create_dataset_embedding(dataset_id)

Arguments
dataset_id dataset id.

Value
integer - 200 on success.

create_dataset_from_dataframe

Upload dataset from data frame.

Description
Upload dataset from data frame.

Usage
create_dataset_from_dataframe(project_id, dataset_name, dataframe, zip = FALSE)

Arguments
project_id id of the project, can be obtained with get_projects().
dataset_name given name of the dataset on the platform.
dataframe data.frame to upload.
zip is the temp file zipped before sending it to Prevision.io (default = FALSE).

Value
list - parsed content of the dataset.
create_dataset_from_datasource

Create a dataset from an existing datasource.

Description

Create a dataset from an existing datasource.

Usage

create_dataset_from_datasource(project_id, dataset_name, datasource_id)

Arguments

- project_id: id of the project, can be obtained with get_projects().
- dataset_name: given name of the dataset on the platform.
- datasource_id: datasource id.

Value

list - parsed content of the dataset.

create_dataset_from_file

Upload dataset from file name.

Description

Upload dataset from file name.

Usage

create_dataset_from_file(
    project_id,
    dataset_name,
    file,
    separator = ",",
    decimal = "."
)
create_datasource

Arguments

- **project_id**: id of the project, can be obtained with `get_projects()`.
- **dataset_name**: given name of the dataset on the platform.
- **file**: path to the dataset.
- **separator**: column separator in the file (default: ",")
- **decimal**: decimal separator in the file (default: ".")

Value

- list - parsed content of the dataset.

create_datasource  
Create a new datasource If check_if_exist is enabled, the function will check if a datasource with the same name already exists. If yes, it will return a message and the information of the existing datasource instead of creating a new one.

Description

Create a new datasource If check_if_exist is enabled, the function will check if a datasource with the same name already exists. If yes, it will return a message and the information of the existing datasource instead of creating a new one.

Usage

```r
create_datasource(
  project_id,
  connector_id,
  name,
  path = "",
  database = "",
  table = "",
  bucket = "",
  request = "",
  check_if_exist = FALSE
)
```

Arguments

- **project_id**: id of the project, can be obtained with `get_projects()`.
- **connector_id**: connector_id linked to the datasource.
- **name**: datasource name.
- **path**: datasource path (for SFTP & FTP connector).
- **database**: datasource database (for SQL connector).
**create_deployment_api_key**

- table: datasource table (for SQL connector).
- bucket: datasource bucket (for S3 connector).
- request: datasource request (for SQL connector).
- check_if_exist: boolean (FALSE by default). If TRUE, makes extra checks to see if a datasource with the same name is already existing.

**Value**

- list - parsed content of the datasource.

---

**create_deployment_api_key**

*Create a new API key for a deployed model.*

**Description**

Create a new API key for a deployed model.

**Usage**

`create_deployment_api_key(deployment_id)`

**Arguments**

- deployment_id: id of the deployment to create an API key on, can be obtained with get_deployments().

**Value**

- list - API key information.

---

**create_deployment_model**

*Create a new deployment for a model.*

**Description**

Create a new deployment for a model.
**create_deployment_predictions**

Create predictions on a deployed model using a dataset.

**Usage**

create_deployment_model(
  project_id,
  name,
  experiment_id,
  main_model_experiment_version_id,
  challenger_model_experiment_version_id = NULL,
  access_type = c("fine_grained"),
  type_violation_policy = c("best_effort"),
  description = NULL,
  main_model_id,
  challenger_model_id = NULL
)

**Arguments**

- **project_id**: id of the project, can be obtained with get_projects().
- **name**: name of the deployment.
- **experiment_id**: id of the experiment to deploy, can be obtained with get_experiment_id_from_name().
- **main_model_experiment_version_id**: id of the experiment_version to deploy, can be obtained with get_experiment_version_id().
- **challenger_model_experiment_version_id**: id of the challenger experiment_version to deploy, can be obtained with get_experiment_version_id().
- **access_type**: type of access of the deployment among "fine_grained" (project defined, default), "private" (instance) or "public" (everyone).
- **type_violation_policy**: handling of type violation when making predictions among "best_effort" (default) or "strict" (stops the prediction if there is a type violation).
- **description**: description of the deployment.
- **main_model_id**: id of the model to deploy
- **challenger_model_id**: id of the challenger model to deploy

**Value**

list - parsed content of the deployment.

---

create_deployment_predictions

Create predictions on a deployed model using a dataset.

**Description**

Create predictions on a deployed model using a dataset.
create_experiment

Usage

create_deployment_predictions(deployment_id, dataset_id)

Arguments

deployment_id id of the deployment, can be obtained with get_deployments().
dataset_id id of the dataset to predict, can be obtained with get_dataset_id_from_name().

Value

integer - 200 on success.

create_experiment Create a new experiment. If check_if_exist is enabled, the function
will check if an experiment with the same name already exists. If yes,
it will return a message and the information of the existing experiment
instead of creating a new one.

Description

Create a new experiment. If check_if_exist is enabled, the function will check if an experiment with
the same name already exists. If yes, it will return a message and the information of the existing
experiment instead of creating a new one.

Usage

create_experiment(
    project_id,
    name,
    provider,
    data_type,
    training_type,
    check_if_exist = FALSE
)

Arguments

project_id id of the project in which we create the experiment.
name name of the experiment.
provider provider of the experiment ("prevision-auto-ml" or "external")
data_type type of data ("tabular", "images" or "timeseries").
training_type type of the training you want to achieve ("regression", "classification", "multi-
classification", "clustering", "object-detection" or "text-similarity").
check_if_exist boolean (FALSE by default). If TRUE, makes extra checks to see if an experi-
ment with the same name is already existing.
Value

list - experiment information.

Description

Create a new version of an existing experiment.

Usage

create_experiment_version(
    experiment_id,
    dataset_id = NULL,
    target_column = NULL,
    holdout_dataset_id = NULL,
    id_column = NULL,
    drop_list = NULL,
    profile = NULL,
    experiment_description = NULL,
    metric = NULL,
    fold_column = NULL,
    normal_models = NULL,
    lite_models = NULL,
    simple_models = NULL,
    with_blend = NULL,
    weight_column = NULL,
    features_engineering_selected_list = NULL,
    features_selection_count = NULL,
    features_selection_time = NULL,
    folder_dataset_id = NULL,
    filename_column = NULL,
    ymin = NULL,
    ymax = NULL,
    xmin = NULL,
    xmax = NULL,
    time_column = NULL,
    start_dw = NULL,
    end_dw = NULL,
    start_fw = NULL,
    end_fw = NULL,
    group_list = NULL,
    apriori_list = NULL,
    content_column = NULL,
create_experiment_version

queries_dataset_id = NULL,
queries_dataset_content_column = NULL,
queries_dataset_id_column = NULL,
queries_dataset_matching_id_description_column = NULL,
top_k = NULL,
lang = NULL,
models_params = NULL,
name = NULL,
onnx_file = NULL,
yaml_file = NULL
)

Arguments

experiment_id  id of the experiment that will host the new version.
dataset_id  id of the dataset used for the training phase.
target_column  name of the TARGET column.
holdout_dataset_id  id of the holdout dataset.
id_column  name of the id column.
drop_list  list of names of features to drop.
profile  chosen profil among "quick", "normal", "advanced".
experiment_description  experiment description.
metric  name of the metric to optimise.
fold_column  name of the fold column.
normal_models  list of (normal) models to select with full FE & hyperparameters search (among "LR", "RF", "ET", "XGB", "LGB", "NN", "CB").
lite_models  list of (lite) models to select with lite FE & default hyperparameters (among "LR", "RF", "ET", "XGB", "LGB", "NN", "CB", "NBC").
simple_models  list of simple models to select (among "LR", "DT").
with_blend  boolean, do we allow to include blend in the modelisation.
weight_column  name of the weight columns.
features_engineering_selected_list  list of feature engineering to select (among "Counter", "Date", "freq", "text_tfidf", "text_word2vec", "text_embedding", "tenc", "poly", "pca", "kmean").
features_selection_count  number of features to keep after the feature selection process.
features_selection_time  time budget in minutes of the feature selection process.
folder_dataset_id  id of the dataset folder (images).
filename_column  name of the file name path (images).
<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ymin</td>
<td>name of the column matching the lower y value of the image (object detection).</td>
</tr>
<tr>
<td>ymax</td>
<td>name of the column matching the higher y value of the image (object detection).</td>
</tr>
<tr>
<td>xmin</td>
<td>name of the column matching the lower x value of the image (object detection).</td>
</tr>
<tr>
<td>xmax</td>
<td>name of the column matching the higher x value of the image (object detection).</td>
</tr>
<tr>
<td>time_column</td>
<td>name of column containing the timestamp (time series).</td>
</tr>
<tr>
<td>start_dw</td>
<td>value of the start of derivative window (time series), should be a strict negative integer.</td>
</tr>
<tr>
<td>end_dw</td>
<td>value of the end of derivative window (time series), should be a negative integer greater than start_dw.</td>
</tr>
<tr>
<td>start_fw</td>
<td>value of the start of forecast window (time series), should be a strict positive integer.</td>
</tr>
<tr>
<td>end_fw</td>
<td>value of the end of forecast window (time series), should be a strict positive integer greater than start_fw.</td>
</tr>
<tr>
<td>group_list</td>
<td>list of name of feature that describes groups (time series).</td>
</tr>
<tr>
<td>apriori_list</td>
<td>list of name of feature that are a priori (time series).</td>
</tr>
<tr>
<td>content_column</td>
<td>content column name (text-similarity).</td>
</tr>
<tr>
<td>queries_dataset_id</td>
<td>id of the dataset containing queries (text-similarity).</td>
</tr>
<tr>
<td>queries_dataset_content_column</td>
<td>name of the column containing queries in the query dataset (text-similarity).</td>
</tr>
<tr>
<td>queries_dataset_id_column</td>
<td>name of the ID column in the query dataset (text-similarity).</td>
</tr>
<tr>
<td>queries_dataset_matching_id_description_column</td>
<td>name of the column matching id in the description dataset (text-similarity).</td>
</tr>
<tr>
<td>top_k</td>
<td>top k individual to find (text-similarity).</td>
</tr>
<tr>
<td>lang</td>
<td>lang of the text (text-similarity).</td>
</tr>
<tr>
<td>models_params</td>
<td>parameters of the model (text-similarity).</td>
</tr>
<tr>
<td>name</td>
<td>name of the external model (external model).</td>
</tr>
<tr>
<td>onnx_file</td>
<td>path to the onnx file (external model).</td>
</tr>
<tr>
<td>yaml_file</td>
<td>path to the yaml file (external model).</td>
</tr>
</tbody>
</table>

**Value**

list - experiment information.
create_export

Export data using an existing exporter and the resource to export

Description

Export data using an existing exporter and the resource to export

Usage

create_export(exporter_id, type, dataset_id = NULL, prediction_id = NULL)

Arguments

exporter_id id of the exporter, can be obtained with get_exporters().
type type of data to export among "dataset", "validation-prediction" or "deployment-prediction"
dataset_id id of the dataset to export (only for type == "dataset")
prediction_id id of the prediction to export (only for type == "validation-prediction" or type == "deployment-prediction")

Value

list - parsed content of the export.

create_exporter

Create a new exporter

Description

Create a new exporter

Usage

create_exporter(
    project_id,
    connector_id,
    name,
    description = "",
    filepath = "",
    file_write_mode = "timestamp",
    database = "",
    table = "",
    database_write_mode = "append",
    bucket = ""
)
### Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>project_id</code></td>
<td>id of the project, can be obtained with <code>get_projects()</code>.</td>
</tr>
<tr>
<td><code>connector_id</code></td>
<td>connector_id linked to the exporter.</td>
</tr>
<tr>
<td><code>name</code></td>
<td>exporter name.</td>
</tr>
<tr>
<td><code>description</code></td>
<td>description of the exporter.</td>
</tr>
<tr>
<td><code>filepath</code></td>
<td>exporter path (for SFTP &amp; FTP connector).</td>
</tr>
<tr>
<td><code>file_write_mode</code></td>
<td>writing type when exporting a file (for SFT &amp; FTP connector, among &quot;times-tamp&quot;, &quot;safe&quot; or &quot;replace&quot;)</td>
</tr>
<tr>
<td><code>database</code></td>
<td>exporter database (for SQL connector).</td>
</tr>
<tr>
<td><code>table</code></td>
<td>exporter table (for SQL connector).</td>
</tr>
<tr>
<td><code>database_write_mode</code></td>
<td>writing type when exporting data within a database (for SQL connector, among &quot;append&quot; or &quot;replace&quot;)</td>
</tr>
<tr>
<td><code>bucket</code></td>
<td>exporter bucket (for S3 connector).</td>
</tr>
</tbody>
</table>

### Value

list - parsed content of the folder.

---

**create_folder**

Upload folder from a local file.

**Description**

Upload folder from a local file.

**Usage**

`create_folder(project_id, folder_name, file)`

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>project_id</code></td>
<td>id of the project, can be obtained with <code>get_projects()</code>.</td>
</tr>
<tr>
<td><code>folder_name</code></td>
<td>given name of the folder on the platform.</td>
</tr>
<tr>
<td><code>file</code></td>
<td>path to the folder.</td>
</tr>
</tbody>
</table>

**Value**

list - parsed content of the folder.
create_pipeline_trigger

Trigger an existing pipeline run.

Description

Trigger an existing pipeline run.

Usage

create_pipeline_trigger(pipeline_id)

Arguments

pipeline_id  id of the pipeline run to trigger, can be obtained with get_pipelines().

Value

integer - 200 on success.

create_prediction

Create a prediction on a specified experiment_version

Description

Create a prediction on a specified experiment_version

Usage

create_prediction(
    experiment_version_id,
    dataset_id = NULL,
    folder_dataset_id = NULL,
    confidence = FALSE,
    best_single = FALSE,
    model_id = NULL,
    queries_dataset_id = NULL,
    queries_dataset_content_column = NULL,
    queries_dataset_id_column = NULL,
    queries_dataset_matching_id_description_column = NULL,
    top_k = NULL
)
create_project

Arguments

- **experiment_version_id**: id of the experiment_version, can be obtained with `get_experiment_version_id()`.
- **dataset_id**: id of the dataset to start the prediction on, can be obtained with `get_datasets()`.
- **folder_dataset_id**: id of the folder dataset to start prediction on, can be obtained with `get_folders()`.
  Only useful for images use cases.
- **confidence**: boolean. If enable, confidence interval will be added to predictions.
- **best_single**: boolean. If enable, best single model (non blend) will be used for making predictions other wise, best model will be used unless if model_id is fed.
- **model_id**: id of the model to start the prediction on. If provided, it will overwrite the "best single" params.
- **queries_dataset_id**: id of the dataset containing queries (text-similarity).
- **queries_dataset_content_column**: name of the content column in the queries dataset (text-similarity).
- **queries_dataset_id_column**: name of the id column in the queries dataset (text-similarity).
- **queries_dataset_matching_id_description_column**: name of the column matching the id (text-similarity).
- **top_k**: number of class to retrieve (text-similarity).

Value

- list - parsed prediction information.

create_project

Create a new project. If check_if_exist is enabled, the function will check if a project with the same name already exists. If yes, it will return a message and the information of the existing project instead of creating a new one.

Description

Create a new project. If check_if_exist is enabled, the function will check if a project with the same name already exists. If yes, it will return a message and the information of the existing project instead of creating a new one.

Usage

```r
create_project(
  name,
  description = NULL,
  color = "#a748f5",
  check_if_exist = FALSE
)```
**create_project_user**

Add user in and existing project.

**Description**

Add user in and existing project.

**Usage**

`create_project_user(project_id, user_mail, user_role)`

**Arguments**

- **project_id** id of the project, can be obtained with `get_projects()`.
- **user_mail** email of the user to be add.
- **user_role** role to grand to the user among "admin", "contributor", "viewer" or "end_user".

**Value**

list - information of project’s users.

---

**create_project_user**

Add user in and existing project.

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>name of the project.</td>
</tr>
<tr>
<td>description</td>
<td>description of the project.</td>
</tr>
<tr>
<td>color</td>
<td>color of the project among &quot;#4876be&quot;, &quot;#4ab6eb&quot;, &quot;#49cf7d&quot;, &quot;#dc8218&quot;, &quot;#ecba35&quot;, &quot;#f45b69&quot;, &quot;#a748f5&quot;, &quot;#b3ca2&quot;, \or &quot;#2fe6d0&quot; (#a748f5 by default).</td>
</tr>
<tr>
<td>check_if_exist</td>
<td>boolean (FALSE by default). If TRUE, makes extra checks to see if a project with the same name is already existing.</td>
</tr>
</tbody>
</table>

**Value**

list - information of the created project.
delete_connector  
*Delete an existing connector.*

**Description**
Delete an existing connector.

**Usage**
delete_connector(connector_id)

**Arguments**
connector_id  id of the connector to be deleted, can be obtained with get_connectors().

**Value**
integer - 200 on success.

delete_contact_point  
*Delete an existing contact_point*

**Description**
Delete an existing contact_point

**Usage**
delete_contact_point(contact_point_id)

**Arguments**
contact_point_id  id of the contact point to be deleted, can be obtained with get_contact_points().

**Value**
integer - 204 on success.
**delete_dataset**

*Delete an existing dataset.*

**Description**

Delete an existing dataset.

**Usage**

`delete_dataset(dataset_id)`

**Arguments**

- `dataset_id`  
  id of the dataset, can be obtained with get_datasets().

**Value**

integer - 204 on success.

---

**delete_datasource**

*Delete a datasource*

**Description**

Delete a datasource

**Usage**

`delete_datasource(datasource_id)`

**Arguments**

- `datasource_id`  
  id of the datasource to be deleted, can be obtained with get_datasources().

**Value**

integer - 200 on success.
**delete_deployment**  
*Delete an existing deployment.*

**Description**
Delete an existing deployment.

**Usage**
```
delete_deployment(deployment_id)
```

**Arguments**
- `deployment_id`: id of the deployment, can be obtained with `get_deployments()`.

**Value**
integer - 204 on success.

---

**delete_experiment**  
*Delete an experiment on the platform.*

**Description**
Delete an experiment on the platform.

**Usage**
```
delete_experiment(experiment_id)
```

**Arguments**
- `experiment_id`: id of the experiment, can be obtained with `get_experiments()`.

**Value**
integer - 204 on success.
**delete_exporter**  
*Delete an exporter*

**Description**  
Delete an exporter

**Usage**  
`delete_exporter(exporter_id)`

**Arguments**  
`exporter_id`  
id of the exporter to be deleted, can be obtained with `get_exporters()`.

**Value**  
integer - 204 on success.

**delete_folder**  
*Delete an existing folder.*

**Description**  
Delete an existing folder.

**Usage**  
`delete_folder(folder_id)`

**Arguments**  
`folder_id`  
id of the folder to be deleted.

**Value**  
integer - 200 on success.
delete_pipeline  
Delete an existing pipeline

**Description**
Delete an existing pipeline

**Usage**
delete_pipeline(pipeline_id, type)

**Arguments**
- pipeline_id: id of the pipeline to be retrieved, can be obtained with get_pipelines().
- type: type of the pipeline to be retrieved among "component", "template", "run".

**Value**
integer - 204 on success.

deleate_prediction  
Delete a prediction.

**Description**
Delete a prediction.

**Usage**
delete_prediction(prediction_id)

**Arguments**
- prediction_id: id of the prediction to be deleted, can be obtained with get_experiment_version_predictions().

**Value**
integer - 204 on success.
list of predictions of experiment_id.
**delete_project**  
*Delete an existing project.*

**Description**
Delete an existing project.

**Usage**
delete_project(project_id)

**Arguments**
- **project_id**  
id of the project, can be obtained with get_projects().

**Value**
integer - 204 on success.

---

**delete_project_user**  
*Delete user in an existing project.*

**Description**
Delete user in and existing project.

**Usage**
delete_project_user(project_id, user_id)

**Arguments**
- **project_id**  
id of the project, can be obtained with get_projects().
- **user_id**  
user_id of the user to be delete, can be obtained with get_project_users().

**Value**
integer - 200 on success.
get_best_model_id

Get the model_id that provide the best predictive performance given experiment_version_id. If include_blend is false, it will return the model_id from the best "non blended" model.

**Description**

Get the model_id that provide the best predictive performance given experiment_version_id. If include_blend is false, it will return the model_id from the best "non blended" model.

**Usage**

get_best_model_id(experiment_version_id, include_blend = TRUE)

**Arguments**

- experiment_version_id: id of the experiment_version, can be obtained with get_experiment_version_id().
- include_blend: boolean, indicating if you want to retrieve the best model among blended models too.

**Value**

character - model_id.

get_connectors

Get information of all connectors available for a given project_id.

**Description**

Get information of all connectors available for a given project_id.

**Usage**

get_connectors(project_id)

**Arguments**

- project_id: id of the project, can be obtained with get_projects().

**Value**

list - parsed content of all connectors for the supplied project_id.
**get_connector_id_from_name**

Get a connector_id from a connector_name for a given project_id. If duplicated name, the first connector_id that match it is retrieved.

---

**Description**

Get a connector_id from a connector_name for a given project_id. If duplicated name, the first connector_id that match it is retrieved.

**Usage**

```python
get_connector_id_from_name(project_id, connector_name)
```

**Arguments**

- `project_id`: id of the project, can be obtained with `get_projects(project_id)`.
- `connector_name`: name of the connector we are searching its id from.

**Value**

- character: id of the connector if found.

---

**get_connector_info**

Get information about connector from its id.

---

**Description**

Get information about connector from its id.

**Usage**

```python
get_connector_info(connector_id)
```

**Arguments**

- `connector_id`: id of the connector to be retrieved, can be obtained with `get_connectors()`.

**Value**

- list: parsed content of the connector.
**get_contact_points**

Get information of all contact points available for a given project_id.

**Description**

Get information of all contact points available for a given project_id.

**Usage**

```python
get_contact_points(project_id)
```

**Arguments**

- `project_id`
  
  id of the project, can be obtained with `get_projects()`.

**Value**

- `list`
  
  parsed content of all contact points for the supplied project_id.

---

**get_contact_point_info**

Get a contact point information from its contact_point_id.

**Description**

Get a contact point information from its contact_point_id.

**Usage**

```python
get_contact_point_info(contact_point_id)
```

**Arguments**

- `contact_point_id`
  
  id of the contact point, can be obtained with `get_contact_points()`.

**Value**

- `list`
  
  information of the contact point.
**get_datasets**

Get information of all datasets available for a given `project_id`.

**Description**

Get information of all datasets available for a given `project_id`.

**Usage**

```python
get_datasets(project_id)
```

**Arguments**

- `project_id`  : id of the project, can be obtained with `get_projects()`.

**Value**

- list : parsed content of all datasets for the supplied `project_id`.

---

**get_dataset_embedding**

Get a dataset embedding from a dataset `_id`.

**Description**

Get a dataset embedding from a dataset `_id`.

**Usage**

```python
get_dataset_embedding(dataset_id)
```

**Arguments**

- `dataset_id` : dataset id.

**Value**

- integer : 200 on success.
### get_dataset_id_from_name

Get a dataset_id from a dataset_name. If duplicated name, the first dataset_id that match it is retrieved.

**Description**

Get a dataset_id from a dataset_name. If duplicated name, the first dataset_id that match it is retrieved.

**Usage**

```r
get_dataset_id_from_name(project_id, dataset_name)
```

**Arguments**

- `project_id` id of the project, can be obtained with get_projects().
- `dataset_name` name of the dataset we are searching its id from. Can be obtained with get_datasets().

**Value**

character - id of the dataset if found.
### get_dataset_info

Get a dataset from its id.

**Description**

Get a dataset from its id.

**Usage**

```python
get_dataset_info(dataset_id)
```

**Arguments**

- `dataset_id` id of the dataset, can be obtained with `get_datasets()`.

**Value**

- list - parsed content of the dataset.

---

### get_datasources

Get information of all data sources available for a given `project_id`.

**Description**

Get information of all data sources available for a given `project_id`.

**Usage**

```python
get_datasources(project_id)
```

**Arguments**

- `project_id` id of the project, can be obtained with `get_projects()`.

**Value**

- list - parsed content of all data_sources for the supplied `project_id`. 
get_datasource_id_from_name

Get a datasource_id from a datasource_name. If duplicated name, the first datasource_id that match it is retrieved.

Description
Get a datasource_id from a datasource_name. If duplicated name, the first datasource_id that match it is retrieved.

Usage
get_datasource_id_from_name(project_id, datasource_name)

Arguments
- project_id: id of the project, can be obtained with get_projects().
- datasource_name: name of the datasource we are searching its id from. Can be obtained with get_datasources().

Value
character - id of the datasource if found.

get_datasource_info
Get a datasource from its id.

Description
Get a datasource from its id.

Usage
get_datasource_info(datasource_id)

Arguments
- datasource_id: id of the data_sources to be retrieved, can be obtained with get_datasources().

Value
list - parsed content of the data_sources.
get_deployments

Get information of all deployments of a given type available for a given project_id.

**Description**

Get information of all deployments of a given type available for a given project_id.

**Usage**

`get_deployments(project_id, type)`

**Arguments**

- `project_id`: id of the project, can be obtained with `get_projects()`.
- `type`: type of the deployment to retrieve among "model" or "app".

**Value**

list - parsed content of all deployments of the given type for the supplied project_id.

get_deployment_alerts

Get information of all alerts related to a deployment_id.

**Description**

Get information of all alerts related to a deployment_id.

**Usage**

`get_deployment_alerts(deployment_id)`

**Arguments**

- `deployment_id`: id of the project, can be obtained with `get_deployments()`.

**Value**

list - parsed content of all alerts for the supplied deployment_id
get_deployment_alert_id_from_name

Get a deployment_alert_id from a name and type for a given deployment_id.

Description
Get a deployment_alert_id from a name and type for a given deployment_id.

Usage
get_deployment_alert_id_from_name(deployment_id, name)

Arguments

- deployment_id id of the deployment, can be obtained with get_deployments().
- name name of the deployment_alert we are searching its id from.

Value
character - id of the deployment_alert if found.

get_deployment_alert_info

Get information about a deployment_alert for a given deployed model.

Description
Get information about a deployment_alert for a given deployed model.

Usage
get_deployment_alert_info(deployment_id, deployment_alert_id)

Arguments

- deployment_id id of the deployment, can be obtained with get_deployments().
- deployment_alert_id id of the deployment_alert to be retrieved, can be obtained with get_deployment_alerts().

Value
list - parsed content of the deployment_alert.
get_deployment_api_keys

Get API keys for a deployed model.

Description
Get API keys for a deployed model.

Usage
get_deployment_api_keys(deployment_id)

Arguments
deployment_id id of the deployment to get API keys, can be obtained with get_deployments().

Value
data.frame - API keys available for deployment_id.

get_deployment_app_logs

Get logs from a deployed app.

Description
Get logs from a deployed app.

Usage
get_deployment_app_logs(deployment_id, log_type)

Arguments
deployment_id id of the deployment to get the log, can be obtained with get_deployments().
log_type type of logs we want to get among "build", "deploy" or "run".

Value
list - logs from deployed apps.
get_deployment_id_from_name

Get a deployment_id from a name and type for a given project_id. If duplicated name, the first deployment_id that match it is retrieved.

Description

Get a deployment_id from a name and type for a given project_id. If duplicated name, the first deployment_id that match it is retrieved.

Usage

get_deployment_id_from_name(project_id, name, type)

Arguments

project_id id of the project, can be obtained with get_projects().
name name of the deployment we are searching its id from.
type type of the deployment to be retrieved among "model" or "app".

Value

character - id of the deployment if found.

generate_deployment_info

Get information about a deployment from its id.

Description

Get information about a deployment from its id.

Usage

get_deployment_info(deployment_id)

Arguments

deployment_id id of the deployment to be retrieved, can be obtained with get_deployments().

Value

list - parsed content of the deployment.
**get_deployment_predictions**

*Get listing of predictions related to a deployment_id.*

**Description**

Get listing of predictions related to a deployment_id.

**Usage**

`get_deployment_predictions(deployment_id)`

**Arguments**

- `deployment_id`  id of the deployment, can be obtained with `get_deployments()`.

**Value**

list - predictions available for a deployed model.

---

**get_deployment_prediction_info**

*Get information related to predictions of a prediction_id.*

**Description**

Get information related to predictions of a prediction_id.

**Usage**

`get_deployment_prediction_info(prediction_id)`

**Arguments**

- `prediction_id`  id of the prediction returned by `create_deployment_predictions` or that can be obtained with `get_deployment_predictions()`.

**Value**

list - prediction information for a deployed model.
get_deployment_usage

Get usage (calls, errors and response time) of the last version of a deployed model.

Usage

get_deployment_usage(deployment_id, usage_type)

Arguments

deployment_id  id of the deployment to get usage, can be obtained with get_deployments().
usage_type  type of usage to get, among "calls", "errors", "response_time".

Value

list - plotly object.

get_experiments

Get information of all experiments available for a given project_id.

Description

Get information of all experiments available for a given project_id.

Usage

get_experiments(project_id)

Arguments

project_id  id of the project, can be obtained with get_projects().

Value

list - parsed content of all experiments for the supplied project_id.
**get_experiment_id_from_name**

*Get a experiment_id from a experiment_name If duplicated name, the first experiment_id that match it is retrieved.*

**Description**

Get a experiment_id from a experiment_name If duplicated name, the first experiment_id that match it is retrieved.

**Usage**

`get_experiment_id_from_name(project_id, experiment_name)`

**Arguments**

- **project_id** id of the project, can be obtained with `get_projects()`.
- **experiment_name** name of the experiment we are searching its id from. Can be obtained with `get_experiments()`.

**Value**

character - id matching the experiment_name if found.

**get_experiment_info**

*Get a experiment from its experiment_id.*

**Description**

Get a experiment from its experiment_id.

**Usage**

`get_experiment_info(experiment_id)`

**Arguments**

- **experiment_id** id of the experiment, can be obtained with `get_experiments()`.

**Value**

list - parsed content of the experiment.
get_experiment_version_features

Get features information related to a experiment_version_id.

Description

Get features information related to a experiment_version_id.

Usage

get_experiment_version_features(experiment_version_id)

Arguments

experiment_version_id

id of the experiment_version, can be obtained with get_experiment_version_id().

Value

list - parsed content of the experiment_version features information.

generate_question

get_experiment_version_id

Get a experiment version id from experiment_id and its version number.

Description

Get a experiment version id from experiment_id and its version number.

Usage

get_experiment_version_id(experiment_id, version_number = 1)

Arguments

experiment_id  id of the experiment, can be obtained with get_experiments().

version_number  number of the version of the experiment. 1 by default

Value

character - experiment version id.
get_experiment_version_info

Get a experiment_version info from its experiment_version_id

Description
Get a experiment_version info from its experiment_version_id

Usage
get_experiment_version_info(experiment_version_id)

Arguments
experiment_version_id
id of the experiment_version, can be obtained with get_experiment_version_id().

Value
list - parsed content of the experiment_version.

get_experiment_version_models

Get a model list related to a experiment_version_id.

Description
Get a model list related to a experiment_version_id.

Usage
get_experiment_version_models(experiment_version_id)

Arguments
experiment_version_id
id of the experiment_version, can be obtained with get_experiment_version_id().

Value
list - parsed content of models attached to experiment_version_id.
get_experiment_version_predictions

*Get a list of prediction from a experiment_version_id.*

**Description**

Get a list of prediction from a experiment_version_id.

**Usage**

```python
get_experiment_version_predictions(
    experiment_version_id,
    generating_type = "user"
)
```

**Arguments**

- `experiment_version_id`
  - id of the experiment_version, can be obtained with `get_experiment_version_id()`.
- `generating_type`
  - can be "user" (= user predictions) or "auto" (= hold out predictions).

**Value**

list - parsed prediction list items.

get_exporters

*Get information of all exporters available for a given project_id.*

**Description**

Get information of all exporters available for a given project_id.

**Usage**

```python
get_exporters(project_id)
```

**Arguments**

- `project_id`
  - id of the project, can be obtained with `get_projects()`.

**Value**

list - parsed content of all exporters for the supplied project_id.
**get_exporter_exports**  
Get all exports done from an exporter_id

**Description**
Get all exports done from an exporter_id

**Usage**
get_exporter_exports(exporter_id)

**Arguments**
- exporter_id: id of the exporter to retrieve information, can be obtained with get_exporters().

**Value**
- list - list of exports of the supplied exporter_id.

---

**get_exporter_id_from_name**  
Get a exporter_id from a exporter_name. If duplicated name, the first exporter_id that match it is retrieved

**Description**
Get a exporter_id from a exporter_name. If duplicated name, the first exporter_id that match it is retrieved

**Usage**
get_exporter_id_from_name(project_id, exporter_name)

**Arguments**
- project_id: id of the project, can be obtained with get_projects().
- exporter_name: name of the exporter we are searching its id from. Can be obtained with get_exporters().

**Value**
- character - id of the exporter if found.
get_exporter_info  
*Get an exporter from its id.*

**Description**

Get an exporter from its id.

**Usage**

```python
get_exporter_info(exporter_id)
```

**Arguments**

`exporter_id`  
Id of the exporter to be retrieved, can be obtained with `get_exporters()`.

**Value**

`list - parsed content of the exporter.`

---

get_features_infos  
*Get information of a given feature related to an experiment_version_id.*

**Description**

Get information of a given feature related to an experiment_version_id.

**Usage**

```python
get_features_infos(experiment_version_id, feature_name)
```

**Arguments**

- `experiment_version_id`  
  Id of the experiment_version, can be obtained with `get_experiment_version_id()`.

- `feature_name`  
  Name of the feature to retrieve information.

**Value**

`list - parsed content of the specific feature.`
get_folder  Get a folder from its id.

Description
Get a folder from its id.

Usage
get_folder(folder_id)

Arguments
folder_id  id of the image folder, can be obtained with get_folders().

Value
list - parsed content of the folder.

get_folders  Get information of all image folders available for a given project_id.

Description
Get information of all image folders available for a given project_id.

Usage
get_folders(project_id)

Arguments
project_id  id of the project, can be obtained with get_projects().

Value
list - parsed content of all folders.
get_folder_id_from_name

Get a folder_id from a folder_name. If duplicated name, the first folder_id that match it is retrieved.

Description

Get a folder_id from a folder_name. If duplicated name, the first folder_id that match it is retrieved.

Usage

get_folder_id_from_name(project_id, folder_name)

Arguments

- project_id: id of the project, can be obtained with get_projects().
- folder_name: name of the folder we are searching its id from. Can be obtained with get_folders().

Value

character - id of the folder if found.

get_model_cv

Get the cross validation file from a specific model.

Description

Get the cross validation file from a specific model.

Usage

get_model_cv(model_id)

Arguments

- model_id: id of the model to get the CV, can be obtained with get_experiment_version_models().

Value

data.frame - cross validation data coming from model_id.
**get_model_feature_importance**

*Get feature importance corresponding to a model_id.*

**Description**

Get feature importance corresponding to a model_id.

**Usage**

```r
get_model_feature_importance(model_id, mode = "raw")
```

**Arguments**

- `model_id`: id of the model, can be obtained with `get_experiment_models()`.
- `mode`: character indicating the type of feature importance among "raw" (default) or "engineered".

**Value**

- `data.frame`: dataset of the model's feature importance.

**get_model_hyperparameters**

*Get hyperparameters corresponding to a model_id.*

**Description**

Get hyperparameters corresponding to a model_id.

**Usage**

```r
get_model_hyperparameters(model_id)
```

**Arguments**

- `model_id`: id of the model, can be obtained with `experimentModels(experiment_id)`.

**Value**

- `list`: parsed content of the model's hyperparameters.
**get_model_infos**

Get model information corresponding to a model_id.

**Description**

Get model information corresponding to a model_id.

**Usage**

get_model_infos(model_id)

**Arguments**

- **model_id**: id of the model, can be obtained with get_experiment_models().

**Value**

list - parsed content of the model.

---

**get_pipelines**

Get information of all pipelines of a given type available for a given project_id.

**Description**

Get information of all pipelines of a given type available for a given project_id.

**Usage**

get_pipelines(project_id, type)

**Arguments**

- **project_id**: id of the project, can be obtained with get_projects().
- **type**: type of the pipeline to retrieve among "component", "template", or "run".

**Value**

list - parsed content of all pipelines of the given type for the supplied project_id.
get_pipeline_id_from_name

Get a pipeline_id from a pipeline_name and type for a given project_id. If duplicated name, the first pipeline_id that match it is retrieved.

Description

Get a pipeline_id from a pipeline_name and type for a given project_id. If duplicated name, the first pipeline_id that match it is retrieved.

Usage

get_pipeline_id_from_name(project_id, name, type)

Arguments

- project_id: id of the project, can be obtained with get_projects().
- name: name of the pipeline we are searching its id from.
- type: type of the pipeline to be retrieved among "component", "template", "run".

Value

- character: id of the connector if found.

get_pipeline_info

Get information about a pipeline from its id and its type.

Description

Get information about a pipeline from its id and its type.

Usage

get_pipeline_info(pipeline_id, type)

Arguments

- pipeline_id: id of the pipeline to be retrieved, can be obtained with get_pipelines().
- type: type of the pipeline to be retrieved among "component", "template", "run".

Value

- list: parsed content of the pipeline.
get_prediction

Get a specific prediction from a prediction_id. Wait up until time_out is reached and wait wait_time between each retry.

Description

Get a specific prediction from a prediction_id. Wait up until time_out is reached and wait wait_time between each retry.

Usage

get_prediction(prediction_id, prediction_type, time_out = 3600, wait_time = 10)

Arguments

- prediction_id: id of the prediction to be retrieved, can be obtained with get_experiment_version_predictions().
- prediction_type: type of prediction among "validation" (not deployed model) and "deployment" (deployed model).
- time_out: maximum number of seconds to wait for the prediction. 3 600 by default.
- wait_time: number of seconds to wait between each retry. 10 by default.

Value

data.frame - predictions coming from prediction_id.

get_prediction_infos

Get a information about a prediction_id.

Description

Get a information about a prediction_id.

Usage

get_prediction_infos(prediction_id)

Arguments

- prediction_id: id of the prediction to be retrieved, can be obtained with get_experiment_version_predictions().

Value

list - parsed prediction information.
**get_projects**

Retrieves all projects.

**Description**

Retrieves all projects.

**Usage**

```python
get_projects()
```

**Value**

- list - list of existing projects.

**get_project_id_from_name**

Get a project_id from a project_name. If duplicated name, the first project_id that match it is retrieved.

**Description**

Get a project_id from a project_name. If duplicated name, the first project_id that match it is retrieved.

**Usage**

```python
get_project_id_from_name(project_name)
```

**Arguments**

- `project_name` - name of the project we are searching its id from. Can be obtained with `get_projects()`.

**Value**

- character - project_id of the project_name if found.
get_project_info  
Get a project from its project_id.

Description
Get a project from its project_id.

Usage
get_project_info(project_id)

Arguments
  project_id  id of the project, can be obtained with get_projects().

Value
list - information of the project.

get_project_users  
Get users from a project.

Description
Get users from a project.

Usage
get_project_users(project_id)

Arguments
  project_id  id of the project, can be obtained with get_projects().

Value
list - information of project's users.
helper_cv_classif_analysis

*Get metrics on a CV file retrieved by Prevision.io for a binary classification use case*

**Description**

Get metrics on a CV file retrieved by Prevision.io for a binary classification use case

**Usage**

```
helper_cv_classif_analysis(actual, predicted, fold, thresh = NULL, step = 1000)
```

**Arguments**

- `actual`: target coming from the cross Validation dataframe retrieved by Prevision.io
- `predicted`: prediction coming from the cross Validation dataframe retrieved by Prevision.io
- `fold`: fold number coming from the cross Validation dataframe retrieved by Prevision.io
- `thresh`: threshold to use. If not provided optimal threshold given F1 score will be computed
- `step`: number of iteration done to find optimal thresh (1000 by default = 0.1% resolution per fold)

**Value**

data.frame - metrics computed between actual and predicted vectors.

---

helper_drift_analysis

[BETA] Return a data.frame that contains features, a boolean indicating if the feature may have a different distribution between the submitted datasets (if p-value < threshold), their exact p-value and the test used to compute it.

**Description**

[BETA] Return a data.frame that contains features, a boolean indicating if the feature may have a different distribution between the submitted datasets (if p-value < threshold), their exact p-value and the test used to compute it.

**Usage**

```
helper_drift_analysis(dataset_1, dataset_2, p_value = 0.05, features = NULL)
```
helper_optimal_prediction

Arguments

- **dataset_1**: the first data set
- **dataset_2**: the second data set
- **p_value**: a p-value that will be the decision criteria for deciding if a feature is suspicious 5% by default
- **features**: a vector of features names that should be tested. If NULL, only the intersection of the names() will be kept

Value

- vector - a vector of suspicious features.

Description

[BETA] Compute the optimal prediction for each rows in a data frame, for a given model, a list of actionable features and a number of samples for each features to be tested.

Usage

```r
helper_optimal_prediction(
  project_id,
  experiment_id,
  model_id,
  df,
  actionable_features,
  nb_sample,
  maximize,
  zip = FALSE,
  version = 1
)
```

Arguments

- **project_id**: id of the project containing the use case.
- **experiment_id**: id of the experiment to be predicted on.
- **model_id**: id of the model to be predicted on.
- **df**: a data frame to be predicted on.
- **actionable_features**: a list of actionable_features features contained in the names of the data frame.
```
helper_plot_classif_analysis

nb_sample a vector of number of sample for each actionable_features features.
maximize a boolean indicating if we maximize or minimize the predicted target.
zip a boolean indicating if the data frame to predict should be zipped prior sending to the instance.
version version of the use case we want to make the prediction on.

Value
data.frame - optimal vector and the prediction associated with for each rows in the original data frame.

Description
Plot RECALL, PRECISION & F1 SCORE versus top n predictions for a binary classification use case

Usage
helper_plot_classif_analysis(actual, predicted, top, compute_every_n = 1)

Arguments
actual true value (0 or 1 only)
predicted prediction vector (probability)
top top individual to analyse
compute_every_n compute indicators every n individuals (1 by default)

Value
data.frame - metrics computed between actual and predicted vectors.
```
pause_experiment_version

*Pause a running experiment_version on the platform.*

**Description**

Pause a running experiment_version on the platform.

**Usage**

```
pause_experiment_version(experiment_version_id)
```

**Arguments**

- `experiment_version_id`
  
  id of the experiment_version, can be obtained with `get_experiment_version_id()`.

**Value**

integer - 200 on success.

pio_download

*Download resources according specific parameters.*

**Description**

Download resources according specific parameters.

**Usage**

```
pio_download(endpoint, tempFile)
```

**Arguments**

- `endpoint`
  
  end of the url of the API call.

- `tempFile`
  
  temporary file to download.

**Value**

list - response from the request.
**pio_init**

*Initialization of the connection to your instance Prevision.io.*

**Description**

Initialization of the connection to your instance Prevision.io.

**Usage**

`pio_init(token, url)`

**Arguments**

- **token**
  - your master token, can be found on your instance on the "API KEY" page.
- **url**
  - the url of your instance.

**Value**

list - url and token needed for connecting to the Prevision.io environment.

**Examples**

```
## Not run: pio_init('eyJhbGciOiJIUlIu', 'https://xxx.prevision.io')
```

---

**pio_list_to_df**

*Convert a list returned from APIs to a dataframe. Only working for consistent list (same naming and number of columns).*

**Description**

Convert a list returned from APIs to a dataframe. Only working for consistent list (same naming and number of columns).

**Usage**

`pio_list_to_df(list)`

**Arguments**

- **list**
  - named list comming from an API call.

**Value**

data.frame - cast a consistent list to a data.frame.

pio_request  

Request the platform. Thanks to an endpoint, the url and the API, you can create request.

Description

Request the platform. Thanks to an endpoint, the url and the API, you can create request.

Usage

pio_request(endpoint, method, data = NULL, upload = FALSE)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>endpoint</td>
<td>end of the url of the API call.</td>
</tr>
<tr>
<td>method</td>
<td>the method needed according the API (Available: POST, GET, DELETE).</td>
</tr>
<tr>
<td>data</td>
<td>object to upload when using method POST.</td>
</tr>
<tr>
<td>upload</td>
<td>used parameter when uploading dataset (for encoding in API call), don’t use it.</td>
</tr>
</tbody>
</table>

Value

list - response from the request.

Examples

## Not run: pio_request(paste0('/quotesingle.Var/jobs/', experiment$jobId), DELETE)

resume_experiment_version  

Resume a paused experiment_version on the platform.

Description

Resume a paused experiment_version on the platform.

Usage

resume_experiment_version(experiment_version_id)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>experiment_version_id</td>
<td>id of the experiment_version, can be obtained with get_experiment_version_id().</td>
</tr>
</tbody>
</table>

Value

integer - 200 on success.
stop_experiment_version

Stop a running or paused experiment_version on the platform.

Description
Stop a running or paused experiment_version on the platform.

Usage
stop_experiment_version(experiment_version_id)

Arguments
experiment_version_id
id of the experiment_version, can be obtained with get_experiment_version_id().

Value
integer - 200 on success.

test_connector
Test an existing connector.

Description
Test an existing connector.

Usage
test_connector(connector_id)

Arguments
connector_id
id of the connector to be tested, can be obtained with get_connectors().

Value
integer - 200 on success.
**test_contact_point**  
*Test an existing contact point*

**Description**
Test an existing contact point

**Usage**
```
test_contact_point(contact_point_id)
```

**Arguments**
- **contact_point_id**
  id of the contact point to be tested, can be obtained with get_contact_points().

**Value**
integer - 200 on success.

---

**test_datasource**  
*Test a datasource*

**Description**
Test a datasource

**Usage**
```
test_datasource(datasource_id)
```

**Arguments**
- **datasource_id**
  id of the datasource to be tested, can be obtained with get_datasources().

**Value**
integer - 200 on success.
test_deployment_type  Check if a type of a deployment is supported

Description
Check if a type of a deployment is supported

Usage
test_deployment_type(type)

Arguments
type  type of the deployment among "model" or "app".

Value
no return value, called for side effects.

test_pipeline_type  Check if a type of a pipeline is supported

Description
Check if a type of a pipeline is supported

Usage
test_pipeline_type(type)

Arguments
type  type of the pipeline among "component", "template", "run".

Value
no return value, called for side effects.
update_experiment_version_description

Update the description of a given experiment_version_id.

**Description**

Update the description of a given experiment_version_id.

**Usage**

```
update_experiment_version_description(experiment_version_id, description = "")
```

**Arguments**

- `experiment_version_id`
  - id of the experiment_version, can be obtained with get_experiment_version_id().
- `description`
  - Description of the experiment.

**Value**

integer - 200 on success.

update_project_user_role

Update user role in and existing project.

**Description**

Update user role in and existing project.

**Usage**

```
update_project_user_role(project_id, user_id, user_role)
```

**Arguments**

- `project_id`
  - id of the project, can be obtained with get_projects().
- `user_id`
  - user_id of the user to be delete, can be obtained with get_project_users().
- `user_role`
  - role to grand to the user among "admin", "contributor", "viewer" and "end_user".

**Value**

list - information of project’s users.
Index

create_connector, 4
create_contact_point, 5
create_dataframe_from_dataset, 5
create_dataset_embedding, 6
create_dataset_from_dataframe, 6
create_dataset_from_datasource, 7
create_dataset_from_file, 7
create_datasource, 8
create_deployment_api_key, 9
create_deployment_model, 9
create_deployment_predictions, 10
create_experiment, 11
create_experiment_version, 12
create_export, 15
create_exporter, 15
create_folder, 16
create_pipeline_trigger, 17
create_prediction, 17
create_project, 18
create_project_user, 19

delete_connector, 20
delete_contact_point, 20
delete_dataset, 21
delete_datasource, 21
delete_deployment, 22
delete_experiment, 22
delete_exporter, 23
delete_folder, 23
delete_pipeline, 24
delete_prediction, 24
delete_project, 25
delete_project_user, 25

generate_best_model_id, 26
generate_connector_id_from_name, 27
generate_connector_info, 27
generate_connectors, 26
generate_contact_point_info, 28
generate_contact_points, 28
get_pipeline_id_from_name, 49
get_pipeline_info, 49
get_pipelines, 48
get_prediction, 50
get_prediction_infos, 50
get_project_id_from_name, 51
get_project_info, 52
get_project_users, 52
get_projects, 51

helper_cv_classif_analysis, 53
helper_drift_analysis, 53
helper_optimal_prediction, 54
helper_plot_classif_analysis, 55

pause_experiment_version, 56
pio_download, 56
pio_init, 57
pio_list_to_df, 57
pio_request, 58

resume_experiment_version, 58

stop_experiment_version, 59

test_connector, 59
test_contact_point, 60
test_datasource, 60
test_deployment_type, 61
test_pipeline_type, 61

update_experiment_version_description, 62
update_project_user_role, 62