R package for Prevision.io

June 7, 2021

`create_connector` *Create a new connector of a supported type (among: "SQL", "HIVE", "FTP", "SFTP", "S3", "GCP").*

Description

Create a new connector of a supported type (among: "SQL", "HIVE", "FTP", "SFTP", "S3", "GCP").

Usage

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

Arguments

`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).

Value

parsed content of the connector.

`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, path = getwd(), is_folder = FALSE)
```

Arguments

`dataset_id`	dataset id.
`path`	path (without / at the end) were to write the downloaded dataset.
`is_folder`	TRUE if it's a folder dataset, FALSE (by default) otherwise.

Value

a R dataframe.

`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.

`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 = F)
```

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 = F).

Value

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

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 = "."
)
```

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

parsed content of the dataset.

`create_datasource` *Create a new datasource*

Description

Create a new datasource

Usage

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

Arguments

project_id	id of the project, can be obtained with get_projects(project_id).
connector_id	connector_id linked to the datasource.
name	datasource name.
path	datasource path (for SFTP & FTP connector).
database	datasource database (for SQL & HIVE & HBASE connector).
table	datasource table (for SQL & HIVE & HBASE connector).
bucket	datasource bucket (for S3 connector).
request	datasource request (for SQL & HIVE connector).

Value

parsed content of the datasource

---

create_folder *Upload folder from a local file.*

Description

Upload folder from a local file.

Usage

```
create_folder(project_id, folder_name, file)
```

Arguments

project_id	id of the project, can be obtained with get_projects().
folder_name	given name of the folder on the platform.
file	path to the folder.

Value

parsed content of the folder.

`create_prediction` *Create a prediction on a specified usecase_version*

Description

Create a prediction on a specified usecase_version

Usage

```
create_prediction(
  usecase_version_id,
  dataset_id = NULL,
  folder_dataset_id = NULL,
  confidence = F,
  best_single = F,
  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
)
```

Arguments

`usecase_version_id`	id of the usecase_version, can be obtained with get_usecase_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 usefull 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 datasat 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 columln 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

parsed prediction list.

create_project *Create a new project.*

Description

Create a new project.

Usage

```
create_project(name, description = NULL, color = "#8D29CC")
```

Arguments

name	name of the project.
description	description of the project.
color	color of the project. #8D29CC by default.

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, can be obtained with get_users().
user_role	role to grand to the user among "admin", "contributor" and "viewer"

Value

list of users in the project

create_usecase *Create a new usecase on the platform.*

Description

Create a new usecase on the platform.

Usage

```
create_usecase(
  project_id,
  name,
  data_type,
  training_type,
  dataset_id,
  target_column = NULL,
  holdout_dataset_id = NULL,
  id_column = NULL,
  drop_list = NULL,
  profile = NULL,
  usecase_description = NULL,
  metric = NULL,
  fold_column = NULL,
  normal_models = NULL,
  lite_models = c("XGB"),
  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,
  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
)
```

Arguments

project_id	id of the project in which we create the usecase.
name	name of the usecase.
data_type	type of data ("tabular" or "images" or "timeseries").
training_type	type of the training you want to achieve ("regression", "classification", "multi-classification", "clustering", "object-detection", "text-similarity").
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".
usecase_description	usecase 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.
lite_models	list of (lite) models to select with lite FE & default hyperparameters.
simple_models	list of simple models to select.
with_blend	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).
ymin	name of the column matching the lower y value of the image (object detection).
ymax	name of the column matching the higher y value of the image (object detection).
xmin	name of the column matching the lower x value of the image (object detection).
xmax	name of the column matching the higher x value of the image (object detection).
time_column	name of column containing the timestamp (time series).
start_dw	value of the start of derivative window (time series), should be a strict negative integer.

end_dw	value of the end of derivative window (time series), should be a negative integer greater than start_dw.
start_fw	value of the start of forecast window (time series), should be a strict positive integer.
end_fw	value of the end of forecast window (time series), should be a strict positive integer greater than start_fw.
group_list	list of name of feature that describes groups (time series).
apriori_list	list of name of feature that are a priori (time series).
content_column	content column name (text-similarity).
queries_dataset_id	id of the dataset containing queries (text-similarity).
queries_dataset_content_column	name of the column containing queries in the query dataset (text-similarity).
queries_dataset_id_column	name of the ID column in the query dataset (text-similarity).
queries_dataset_matching_id_description_column	name of the column matching id in the description dataset (text-similarity).
top_k	top k individual to find (text-similarity).
lang	lang of the text (text-similarity).
models_params	parameters of the model (text-similarity).

Value

usecase information.

`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().

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().

delete_datasource *Delete a datasource*

Description

Delete a datasource

Usage

```
delete_datasource(datasource_id)
```

Arguments

datasource_id	id of the connector to be deleted, can be obtained with get_datasources().

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.

delete_prediction *Delete a prediction.*

Description

Delete a prediction.

Usage

```
delete_prediction(prediction_id)
```

Arguments

prediction_id	id of the prediction to be retrieved, can be obtained with get_usecase_version_predictions()

Value

list of predictions of usecase_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

200 on success

delete_project_user *Delete user in and 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

200 on success

delete_usecase_version *Delete a usecase_version on the platform.*

Description

Delete a usecase_version on the platform.

Usage

```
delete_usecase_version(usecase_version_id)
```

Arguments

usecase_version_id	id of the usecase_version, can be obtained with get_usecase_version_id().

`get_best_model_id` *Get the model_id that provide the best predictive performance given usecase_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 usecase_version_id. If include_blend is false, it will return the model_id from the best "non blended" model.

Usage

```
get_best_model_id(usecase_version_id, include_blend = TRUE)
```

Arguments

`usecase_version_id`	id of the usecase_version, can be obtained with get_usecase_version_id().
`include_blend`	boolean, indicating if you want to retrieve the best model among blended models too.

Value

model_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

```
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

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

```
get_connector_info(connector_id)
```

Arguments

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

Value

parsed content of the connector.

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

parsed content of all connectors for the suppled project_id.

get_dataset_embedding *Get a dataset embedding from a dataset_id.*

Description

Get a dataset embedding from a dataset_id.

Usage

```
get_dataset_embedding(dataset_id)
```

Arguments

dataset_id	dataset id.

`get_dataset_head` *Show the head of a dataset from its id.*

Description

Show the head of a dataset from its id.

Usage

```
get_dataset_head(dataset_id)
```

Arguments

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

Value

head of the dataset as a data.frame object.

`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

```
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

id of the dataset if found.

get_dataset_info *Get a dataset from its id.*

Description

Get a dataset from its id.

Usage

```
get_dataset_info(dataset_id)
```

Arguments

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

Value

parsed content of the dataset.

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

```
get_datasets(project_id)
```

Arguments

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

Value

parsed content of all datasets for the suppled 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(project_id).
`datasource_name`	name of the connector we are searching its id from. Can be obtained with get_datasources().

Value

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

parsed content of the data_sources

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

```
get_datasources(project_id)
```

Arguments

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

Value

parsed content of all data_sources for the suppled project_id.

get_features_infos *Get information of a given feature related to a usecase_version_id.*

Description

Get information of a given feature related to a usecase_version_id.

Usage

```
get_features_infos(usecase_version_id, feature_name)
```

Arguments

usecase_version_id	id of the usecase_version, can be obtained with get_usecase_version_id().
feature_name	name of the feature to retrive information.

Value

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

parsed content of the folder.

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

id of the folder if found.

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

parsed content of all folders.

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_usecase_version_models().

Value

a dataframe containing cross validation data.

`get_model_feature_importance` *Get feature importance corresponding to a model_id.*

Description

Get feature importance corresponding to a model_id.

Usage

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

Arguments

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

Value

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

```
get_model_hyperparameters(model_id)
```

Arguments

`model_id`	id of the model, can be obtained with usecaseModels(usecase_id).

Value

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_usecase_models().

Value

parsed content of the model.

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, time_out = 3600, wait_time = 10)
```

Arguments

prediction_id	id of the prediction to be retrieved, can be obtained with get_usecase_version_predictions().
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

a data.frame with the predictions.

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_usecase_version_predictions().

Value

list of prediction information.

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

```
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

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

parsed content 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

parsed content of the project's users

get_projects *Retrieves all projects.*

Description

Retrieves all projects.

Usage

```
get_projects()
```

Value

a project list.

get_usecase_id_from_name *Get a usecase_id from a usecase_name If duplicated name, the first usecase_id that match it is retrieved.*

Description

Get a usecase_id from a usecase_name If duplicated name, the first usecase_id that match it is retrieved.

Usage

```
get_usecase_id_from_name(project_id, usecase_name)
```

Arguments

project_id	id of the project, can be obtained with get_projects().
usecase_name	name of the usecase we are searching its id from. Can be obtained with get_usecases().

Value

usecase_id of the usecase_name if found.

get_usecase_info *Get a usecase from its usecase_id and its version number.*

Description

Get a usecase from its usecase_id and its version number.

Usage

```
get_usecase_info(usecase_id, version_number = NULL)
```

Arguments

usecase_id	id of the usecase, can be obtained with get_usecases().
version_number	number of the version of the usecase. If not supplied, retrieve all usecase information

Value

parsed content of the usecase.

get_usecase_version_features

Get features information related to a usecase_version_id.

Description

Get features information related to a usecase_version_id.

Usage

```
get_usecase_version_features(usecase_version_id)
```

Arguments

usecase_version_id	id of the usecase_version, can be obtained with get_usecase_version_id().

Value

parsed content of the usecase_version features information.

get_usecase_version_id

Get a usecase version id from a usecase_id and its version number.

Description

Get a usecase version id from a usecase_id and its version number.

Usage

```
get_usecase_version_id(usecase_id, version_number = 1)
```

Arguments

usecase_id	id of the usecase, can be obtained with get_usecases().
version_number	number of the version of the usecase. 1 by default

Value

usecase version id.

get_usecase_version_models

Get a model list related to a usecase_version_id.

Description

Get a model list related to a usecase_version_id.

Usage

```
get_usecase_version_models(usecase_version_id)
```

Arguments

usecase_version_id	id of the usecase_version, can be obtained with get_usecase_version_id().

Value

parsed content of models attached to usecase_version_id.

get_usecase_version_predictions

Get a list of prediction from a usecase_version_id.

Description

Get a list of prediction from a usecase_version_id.

Usage

```
get_usecase_version_predictions(usecase_version_id, generating_type = "user")
```

Arguments

usecase_version_id	id of the usecase_version, can be obtained with get_usecase_version_id().
generating_type	can be "user" (= user predictions) or "auto" (= hold out predictions).

Value

parsed prediction list items.

get_usecases *Get information of all usecases available for a given project_id.*

Description

Get information of all usecases available for a given project_id.

Usage

```
get_usecases(project_id)
```

Arguments

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

Value

parsed content of all usecases for the suppled project_id.

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 comming from the cross Validation dataframe retrieved by Prevision.io
predicted	prediction comming from the cross Validation dataframe retrieved by Prevision.io
fold	fold number comming 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 with metrics computed on the CV

`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)
```

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

a vector of suspicious features

`helper_optimal_prediction` *[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*

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

```
helper_optimal_prediction(
  usecase_id,
  model_id,
  df,
  actionable_features,
  nb_sample,
  maximize,
  zip = F
)
```

Arguments

usecase_id	the id of the usecase to be predicted on
model_id	the id of the model to be predicted on
df	a data frame to be predicted on
actionable_features	a list of actionable_featuress features contained in the names of the data frame
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

Value

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

---

helper_plot_classif_analysis

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

---

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 with metrics computed on the CV

pause_usecase_version *Pause a running usecase_version on the platform.*

Description

Pause a running usecase_version on the platform.

Usage

```
pause_usecase_version(usecase_version_id)
```

Arguments

usecase_version_id	id of the usecase_version, can be obtained with get_usecase_version_id().

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

write on disk the content of the temporary file.

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.

Examples

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

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

endpoint	end of the url of the API call.
method	the method needed according the API (Available: POST, GET, DELETE).
data	object to upload when using method POST.
upload	used parameter when uploading dataset (for encoding in API call), don't use it.

Examples

```
## Not run: pio_request(paste0('/jobs/', usecase$jobId), DELETE)
```

resume_usecase_version *Resume a paused usecase_version on the platform.*

Description

Resume a paused usecase_version on the platform.

Usage

```
resume_usecase_version(usecase_version_id)
```

Arguments

usecase_version_id	id of the usecase_version, can be obtained with get_usecase_version_id().

stop_usecase_version *Stop a running or paused usecase_version on the platform.*

Description

Stop a running or paused usecase_version on the platform.

Usage

```
stop_usecase_version(usecase_version_id)
```

Arguments

usecase_version_id	id of the usecase_version, can be obtained with get_usecase_version_id().

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().

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().

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" and "viewer"

Value

list of users in the project

`update_usecase_version_description` *Update the description of a given usecase_version_id.*

Description

Update the description of a given usecase_version_id.

Usage

```
update_usecase_version_description(usecase_version_id, description = "")
```

Arguments

`usecase_version_id`	id of the usecase_version, can be obtained with get_usecase_version_id().
`description`	Description of the usecase.

Index