Package R for Prevision.io

December 23, 2020

`createApp` *Create a new app*

Description

Create a new app

Usage

```
createApp(type, name, tag, route, modelId = NULL, path = NULL)
```

Arguments

`type`	app type (can be "model", "notebook", "app")
`name`	app name.
`tag`	tag of the app (integer).
`route`	app route.
`modelId`	id of the model to deploy if you have selected "model" as type.
`path`	path of your code if you have selected "notebook" or "app" as type.

Value

parsed content of the app.

`createConnector` *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

```
createConnector(
  type,
  name,
  host,
  port,
  username,
  password,
  googleCredentials = NULL
)
```

Arguments

`type`	connector type.
`name`	connector name.
`host`	connector host.
`port`	connector port.
`username`	connector username.
`password`	connector password.
`googleCredentials`	google credentials JSON (for GCP only).

Value

parsed content of the connector.

`createDataframeFromDataset` *Create a dataframe from a datasetId.*

Description

Create a dataframe from a datasetId.

Usage

```
createDataframeFromDataset(datasetId)
```

Arguments

datasetId	dataset id.

Value

a R dataframe

`createDatasetFromDataframe`
Upload dataset from data frame.

Description

Upload dataset from data frame.

Usage

```
createDatasetFromDataframe(datasetName, dataframe, zip = F)
```

Arguments

datasetName	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?

Value

parsed content of the dataset object

`createDatasetFromDatasource`
Create a dataset from an existing datasource.

Description

Create a dataset from an existing datasource.

Usage

```
createDatasetFromDatasource(datasetName, datasourceId)
```

Arguments

datasetName	given name of the dataset on the platform.
datasourceId	datasource id.

Value

parsed content of the dataset object

createDatasetFromFilename *Upload dataset from file name.*

Description

Upload dataset from file name.

Usage

```
createDatasetFromFilename(datasetName, local_path)
```

Arguments

datasetName	given name of the dataset on the platform.
local_path	path to the dataset.

Value

parsed content of the dataset.

createDatasource *Create a new datasource*

Description

Create a new datasource

Usage

```
createDatasource(
  connectorId,
  name,
  path = "",
  database = "",
  table = "",
  bucket = "",
  request = ""
)
```

Arguments

connectorId	connectorId 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

`createFolder` *Upload folder from a local file.*

Description

Upload folder from a local file.

Usage

```
createFolder(folder_name, local_path)
```

Arguments

`folder_name`	given name of the folder on the platform.
`local_path`	path to the folder.

Value

parsed content of the folder.

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

Usage

```
cvClassifAnalysis(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
	data.frame with metrics computed on the CV
	Get metrics on a CV file retrieved by Prevision.io for a binary classification use case

deleteApp *Delete an app*

Description

Delete an app

Usage

```
deleteApp(appId)
```

Arguments

appId	id of the app to be deleted, can be obtained with listApps().

Value

response status code (200 on success)

deleteConnector *Delete a connector.*

Description

Delete a connector.

Usage

```
deleteConnector(connectorId)
```

Arguments

connectorId	id of the connector to be deleted, can be obtained with getConnectors().

deleteDataset *Delete a dataset on the platform.*

Description

Delete a dataset on the platform.

Usage

```
deleteDataset(datasetId)
```

Arguments

datasetId	id of the dataset, can be obtained with getDatasets().

deleteDatasource *Delete a datasource*

Description

Delete a datasource

Usage

```
deleteDatasource(datasourceId)
```

Arguments

datasourceId	id of the connector to be deleted, can be obtained with listConnectors().

deletePrediction *Delete a prediction of a given model and its version number*

Description

Delete a prediction of a given model and its version number

Usage

```
deletePrediction(usecaseId, predictionId, versionNumber = 1)
```

Arguments

usecaseId	id of the usecase, can be obtained with getUsecases().
predictionId	id of the prediction to be retrieved, can be obtained with usecasePredictions()
versionNumber	number of the version of the usecase. 1 by default.

Value

list of predictions of usecaseId

deleteScheduler *Delete a scheduler task*

Description

Delete a scheduler task

Usage

```
deleteScheduler(schedulerId)
```

Arguments

schedulerId	id of the scheduler task to be deleted, can be obtained with listScheduler().

Value

response status code (200 on success)

deleteUsecase *Delete a version of a usecase on the platform*

Description

Delete a version of a usecase on the platform

Usage

```
deleteUsecase(usecaseId, versionNumber = 1)
```

Arguments

usecaseId	id of the usecase, can be obtained with getUsecases().
versionNumber	number of the version of the usecase. 1 by default.

`downloadDataset` *Download a dataset from a datasetId and write it in a folder.*

Description

Download a dataset from a datasetId and write it in a folder.

Usage

```
downloadDataset(datasetId, path = getwd(), isFolder = FALSE)
```

Arguments

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

Value

the complete path to the file written

`driftAnalysis` *[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

```
driftAnalysis(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 \itemfeaturesa vector of features names that should be tested. If NULL, only the intersection of the names() will be kept a vector of suspicious features [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.

`getApp` *Get an app from its id.*

Description

Get an app from its id.

Usage

```
getApp(appId)
```

Arguments

`appId`	id of the app to be retrieved, can be obtained with listApps().

Value

parsed content of the app.

`getAppIdFromName` *Get an appId from an appName If duplicated name, the first appId that match it is retrieved*

Description

Get an appId from an appName If duplicated name, the first appId that match it is retrieved

Usage

```
getAppIdFromName(appName)
```

Arguments

`appName`	name of the app we are searching its id from. Can be obtained with listApps().

Value

id of the app if found.

getApps — *Retrieves all applications available*

Description

Retrieves all applications available

Usage

```
getApps()
```

Value

an app list.

getBestModelId — *Get the modelId that provide the best predictive performance given usecaseId and versionNumber. If includeBlend is false, it will return the modelId from the best "non blended" model.*

Description

Get the modelId that provide the best predictive performance given usecaseId and versionNumber. If includeBlend is false, it will return the modelId from the best "non blended" model.

Usage

```
getBestModelId(usecaseId, versionNumber = 1, includeBlend = TRUE)
```

Arguments

usecaseId	id of the usecase, can be obtained with getUsecases().
versionNumber	number of the version of the usecase. 1 by default.
includeBlend	boolean, indicating if you want to retrieve the best model among blended models too.

Value

modelId

`getConnectorIdFromName` *Get a connectorId from a connectorName. If duplicated name, the first connectorId that match it is retrieved*

Description

Get a connectorId from a connectorName. If duplicated name, the first connectorId that match it is retrieved

Usage

```
getConnectorIdFromName(connectorName)
```

Arguments

`connectorName`	name of the connector we are searching its id from. Can be obtained with getConnectors().

Value

id of the connector if found.

`getConnectorInfos` *Get informations about connector from its id.*

Description

Get informations about connector from its id.

Usage

```
getConnectorInfos(connectorId)
```

Arguments

`connectorId`	id of the connector to be retrieved, can be obtained with getConnectors().

Value

parsed content of the connector.

getConnectors *Get informations of all connectors available.*

Description

Get informations of all connectors available.

Usage

```
getConnectors()
```

Value

parsed content of all connectors

getDatasetEmbedding *Get a dataset embedding from a datasetId.*

Description

Get a dataset embedding from a datasetId.

Usage

```
getDatasetEmbedding(datasetId)
```

Arguments

datasetId	dataset id.

getDatasetHead *Show the head of a dataset from its id.*

Description

Show the head of a dataset from its id.

Usage

```
getDatasetHead(datasetId)
```

Arguments

datasetId	id of the dataset, can be obtained with getDatasets().

Value

head of the dataset as a data.frame object.

`getDatasetIdFromName` *Get a datasetId from a datasetName If duplicated name, the first datasetId that match it is retrieved*

Description

Get a datasetId from a datasetName If duplicated name, the first datasetId that match it is retrieved

Usage

```
getDatasetIdFromName(datasetName)
```

Arguments

`datasetName`	name of the dataset we are searching its id from. Can be obtained with getDatasets().

Value

id of the dataset if found.

`getDatasetInfos` *Get a dataset from its id.*

Description

Get a dataset from its id.

Usage

```
getDatasetInfos(datasetId)
```

Arguments

`datasetId`	id of the dataset, can be obtained with getDatasets().

Value

parsed content of the dataset.

`getDatasets` *Get informations of all tabular datasets availables.*

Description

Get informations of all tabular datasets availables.

Usage

```
getDatasets()
```

Value

parsed content of all datasets.

`getDatasourceIdFromName` *Get a datasourceId from a datasourceName If duplicated name, the first datasourceId that match it is retrieved*

Description

Get a datasourceId from a datasourceName If duplicated name, the first datasourceId that match it is retrieved

Usage

```
getDatasourceIdFromName(datasourceName)
```

Arguments

`datasourceName`	name of the connector we are searching its id from. Can be obtained with getDatasources().

Value

id of the datasource if found.

getDatasourceInfos *Get a datasource from its id.*

Description

Get a datasource from its id.

Usage

```
getDatasourceInfos(datasourceId)
```

Arguments

datasourceId	id of the datasources to be retrieved, can be obtained with getDatasources().

Value

parsed content of the datasources

getDatasources *Get informations of all data sources available.*

Description

Get informations of all data sources available.

Usage

```
getDatasources()
```

Value

parsed content of all datasources

getFeaturesInfos *Get information of a given feature related to a usecaseId and its version number*

Description

Get information of a given feature related to a usecaseId and its version number

Usage

```
getFeaturesInfos(usecaseId, featureName, versionNumber = 1)
```

Arguments

usecaseId	id of the usecase, can be obtained with getUsecases().
featureName	name of the feature to retrive information
versionNumber	number of the version of the usecase. 1 by default.

Value

parsed content of the specific feature

---

getFolder *Get a folder from its id.*

Description

Get a folder from its id.

Usage

```
getFolder(folder_id)
```

Arguments

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

Value

parsed content of the dataset.

---

getFolders *Get informations of all folders availables.*

Description

Get informations of all folders availables.

Usage

```
getFolders()
```

Value

parsed content of all folders.

`getModelFeatureImportance` *Get feature importance corresponding to usecaseId and modelId and its version number*

Description

Get feature importance corresponding to usecaseId and modelId and its version number

Usage

```
getModelFeatureImportance(usecaseId, modelId, versionNumber = 1, mode = "raw")
```

Arguments

`usecaseId`	id of the usecase, can be obtained with getUsecases().
`modelId`	id of the model, can be obtained with usecaseModels(usecaseId).
`versionNumber`	number of the version of the usecase. 1 by default.
`mode`	character indicating the type of feature importance among "raw" (default) or "engineered"

Value

dataset of the model's feature importance

`getModelHyperparameters` *Get hyperparameters corresponding to usecaseId and modelId and its version number*

Description

Get hyperparameters corresponding to usecaseId and modelId and its version number

Usage

```
getModelHyperparameters(usecaseId, modelId, versionNumber = 1)
```

Arguments

`usecaseId`	id of the usecase, can be obtained with getUsecases().
`modelId`	id of the model, can be obtained with usecaseModels(usecaseId).
`versionNumber`	number of the version of the usecase. 1 by default.

Value

parsed content of the model's hyperparameters

`getModelInfos` *Get model informations corresponding to usecaseId and modelId and its version number*

Description

Get model informations corresponding to usecaseId and modelId and its version number

Usage

```
getModelInfos(usecaseId, modelId, versionNumber = 1)
```

Arguments

`usecaseId`	id of the usecase, can be obtained with getUsecases().
`modelId`	id of the model, can be obtained with usecaseModels(usecaseId).
`versionNumber`	number of the version of the usecase. 1 by default.

Value

parsed content of the model

`getPrediction` *Get a specific prediction from a specific usecase / version number. Wait up until timeOut is reached and wait waitTime between each retry.*

Description

Get a specific prediction from a specific usecase / version number. Wait up until timeOut is reached and wait waitTime between each retry.

Usage

```
getPrediction(
  usecaseId,
  predictionId,
  versionNumber = 1,
  timeOut = 3600,
  waitTime = 10
)
```

Arguments

`usecaseId`	id of the usecase, can be obtained with getUsecases().
`predictionId`	id of the prediction to be retrieved, can be obtained with usecasePredictions().
`versionNumber`	number of the version of the usecase. 1 by default.
`timeOut`	maximum number of seconds to wait for the prediction. 3 600 by default.
`waitTime`	number of seconds to wait between each retry. 10 by default.

Value

a data.frame with the predictions

`getPredictionEvents` *Get events for prediction of usecaseId and its version number*

Description

Get events for prediction of usecaseId and its version number

Usage

```
getPredictionEvents(usacaseId, versionNumber = 1)
```

Arguments

`versionNumber`	number of the version of the usecase. 1 by default.
`usecaseId`	id of the usecase, can be obtained with getUsecases().

Value

parsed event list

`getPredictionInfos` *Get a informations about a prediction from a specific usecase / version number*

Description

Get a informations about a prediction from a specific usecase / version number

Usage

```
getPredictionInfos(usecaseId, predictionId, versionNumber = 1)
```

Arguments

`usecaseId`	id of the usecase, can be obtained with getUsecases().
`predictionId`	id of the prediction to be retrieved, can be obtained with usecasePredictions()
`versionNumber`	number of the version of the usecase. 1 by default.

Value

list of prediction informations

`getProfile` *Get user profile informations*

Description

Get user profile informations

Usage

```
getProfile()
```

Value

informations about user (email, company...)

`getScheduler` *Retrieves all scheduler tasks*

Description

Retrieves all scheduler tasks

Usage

```
getScheduler()
```

Value

a scheduler tasks list.

`getSchedulerIdFromName` *Get an schedulerId from an schedulerName If duplicated name, the first schedulerId that match it is retrieved*

Description

Get an schedulerId from an schedulerName If duplicated name, the first schedulerId that match it is retrieved

Usage

```
getSchedulerIdFromName(schedulerName)
```

Arguments

`schedulerName`	name of the scheduler task we are searching its id from. Can be obtained with listScheduler().

Value

id of the scheduler task if found.

getSharedUsecaseUsers *Get the list of users that can access the usecase*

Description

Get the list of users that can access the usecase

Usage

```
getSharedUsecaseUsers(usecaseId)
```

Arguments

usecaseId	id of the usecase, can be obtained with getUsecases().

getUsecaseCV *Get the cross validation file from a specific model of a given usecase / version*

Description

Get the cross validation file from a specific model of a given usecase / version

Usage

```
getUsecaseCV(usecaseId, modelId, versionNumber = 1)
```

Arguments

usecaseId	id of the usecase, can be obtained with getUsecases().
modelId	id of the model to get the CV, can be obtained with usecaseModels(usecaseId).
versionNumber	number of the version of the usecase. 1 by default.

Value

a dataframe

`getUsecaseEvents` *Get events of a version of ausecase. If no usecaseId provided, get events from all usecases*

Description

Get events of a version of ausecase. If no usecaseId provided, get events from all usecases

Usage

```
getUsecaseEvents(usecaseId = NULL, versionNumber = 1)
```

Arguments

`usecaseId`	id of the usecase, can be obtained with getUsecases().
`versionNumber`	number of the version of the usecase. 1 by default.

Value

parsed event list

`getUsecaseFeatures` *Get features informations related to a usecaseId and its version number*

Description

Get features informations related to a usecaseId and its version number

Usage

```
getUsecaseFeatures(usecaseId, versionNumber = 1)
```

Arguments

`usecaseId`	id of the usecase, can be obtained with getUsecases().
`versionNumber`	number of the version of the usecase. 1 by default.

Value

parsed content of the usecase features informations

`getUsecaseIdFromName` *Get a usecaseId from a usecaseName If duplicated name, the first usecaseId that match it is retrieved*

Description

Get a usecaseId from a usecaseName If duplicated name, the first usecaseId that match it is retrieved

Usage

```
getUsecaseIdFromName(usecaseName)
```

Arguments

`usecaseName`	name of the usecase we are searching its id from. Can be obtained with getUsecases().

Value

usecaseId of the usecaseName if found.

`getUsecaseInfos` *Get a usecase from its usecaseId and its version number*

Description

Get a usecase from its usecaseId and its version number

Usage

```
getUsecaseInfos(usecaseId, versionNumber = 1)
```

Arguments

`usecaseId`	id of the usecase, can be obtained with getUsecases().
`versionNumber`	number of the version of the usecase. 1 by default.

Value

parsed content of the usecase

getUsecaseModels *Get a model list related to a usecaseId and its version number*

Description

Get a model list related to a usecaseId and its version number

Usage

```
getUsecaseModels(usecaseId, versionNumber = 1)
```

Arguments

usecaseId	id of the usecase, can be obtained with getUsecases().
versionNumber	number of the version of the usecase. 1 by default.

Value

parsed content of models attached to usecaseId

getUsecasePredictions *Get a usecase from its usecaseId and its version number*

Description

Get a usecase from its usecaseId and its version number

Usage

```
getUsecasePredictions(usecaseId, generatingType = "user", versionNumber = 1)
```

Arguments

usecaseId	id of the usecase, can be obtained with getUsecases().
generatingType	can be "user" (= user predictions) or "auto" (= hold out predictions)
versionNumber	number of the version of the usecase. 1 by default.

Value

parsed prediction list items

getUsecaseSchema *Get schema related to a usecaseId and its version number*

Description

Get schema related to a usecaseId and its version number

Usage

```
getUsecaseSchema(usecaseId, versionNumber = 1)
```

Arguments

usecaseId	id of the usecase, can be obtained with getUsecases().
versionNumber	number of the version of the usecase. 1 by default.

Value

parsed content of the usecase schema

getUsecaseTasks *Get all tasks related to a usecaseId and its version number*

Description

Get all tasks related to a usecaseId and its version number

Usage

```
getUsecaseTasks(usecaseId, versionNumber = 1)
```

Arguments

usecaseId	id of the usecase, can be obtained with getUsecases().
versionNumber	number of the version of the usecase. 1 by default.

Value

parsed content of the usecase tasks

getUsecases *Retrieves all usecases*

Description

Retrieves all usecases

Usage

```
getUsecases()
```

Value

a usecase list.

getVersion *Get Prevision.io version number of the given instance*

Description

Get Prevision.io version number of the given instance

Usage

```
getVersion()
```

Value

Prevision.io version number

initPrevisionioClient *Initialization of the Client on the platform.*

Description

Thanks to the token and the url, you can create a Prevision.io Client.

Usage

```
initPrevisionioClient(token, url)
```

Arguments

token	String, your master token available on the platform.
url	String, the url of your access page to the platform.

Examples

```
## Not run: initPrevisionioClient(token, 'https://xxx.prevision.io')
```

kc_house *House sales in king country*

Description

This dataset comes from the website kaggle <https://www.kaggle.com/harlfoxem/housesalesprediction> Dataset for regression with target "price"

Usage

```
kc_house
```

Format

A data.frame with 10000 rows and 21 variables

Examples

```
kc_house
```

optimalPrediction *[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

```
optimalPrediction(
  usecaseId,
  modelId,
  df,
  actionable,
  nbSample,
  maximize,
  zip = F
)
```

Arguments

usecaseId	the id of the usecase to be predicted on
modelId	the id of the model to be predicted on
df	a data frame to be predicted on
actionable	a list of actionables features contained in the names of the data frame
nbSample	a vector of number of sample for each actionable 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

---

`pauseUsecase` *Pause a version of a usecase on the platform*

Description

Pause a version of a usecase on the platform

Usage

```
pauseUsecase(usecaseId, versionNumber = 1)
```

Arguments

usecaseId	id of the usecase, can be obtained with getUsecases().
versionNumber	number of the version of the usecase. 1 by default.

`plotClassifAnalysis` *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

```
plotClassifAnalysis(actual, predicted, top)
```

Arguments

actual	true value (0 or 1 only)
predicted	prediction vector (probability)
top	top individual to analyse

Value

data.frame with metrics computed on the CV

previsionDownload *Download according specific parameters*

Description

Download according specific parameters

Usage

```
previsionDownload(endpoint, tempFile)
```

Arguments

endpoint	string, end of the url of the API call.
tempFile	file, tempory file to be download.

Value

write on disk the content of the temporary file.

previsionioRequest *Request the platform*

Description

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

Usage

```
previsionioRequest(endpoint, method, data = NULL, upload = FALSE)
```

Arguments

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

Examples

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

resumeUsecase *Resume a version of usecase on the platform*

Description

Resume a version of usecase on the platform

Usage

```
resumeUsecase(usecaseId, versionNumber = 1)
```

Arguments

usecaseId	id of the usecase, can be obtained with getUsecases().
versionNumber	number of the version of the usecase. 1 by default.

shareUsecase *Share a usecase to a specific user*

Description

Share a usecase to a specific user

Usage

```
shareUsecase(usecaseId, email)
```

Arguments

usecaseId	id of the usecase, can be obtained with getUsecases().
email	email adress of the user you want to share the usecase with. Should be a platform user.

Value

list of shared users

startDatasetEmbedding *Start a dataset embedding from a datasetId.*

Description

Start a dataset embedding from a datasetId.

Usage

```
startDatasetEmbedding(datasetId)
```

Arguments

datasetId	dataset id.

`startPrediction` *Start a prediction on a existing usecase with a given datasetId and a given version number*

Description

Start a prediction on a existing usecase with a given datasetId and a given version number

Usage

```
startPrediction(
  usecaseId,
  datasetId,
  datasetFolderId = NULL,
  confidence = F,
  bestSingle = F,
  modelId = NULL,
  versionNumber = 1
)
```

Arguments

`usecaseId`	id of the usecase, can be obtained with getUsecases().
`datasetId`	id of the dataset to start the prediction on, can be obtained with getUsecases()
`datasetFolderId`	id of the folder dataset to start prediction on, can be obtained with getUsecases(). Only usefull for images use cases
`confidence`	boolean. If enable, confidence intervalle will be added to predictions
`bestSingle`	boolean. If enable, best single model (non blend) will be used for making predictions other wise, best model will be used unless if modelId is fed
`modelId`	id of the model to start the prediction on. If provided, it will overwrite the "best single" params
`versionNumber`	number of the version of the usecase. 1 by default.

Value

parsed prediction list

`startScheduler` *Start a scheduled task*

Description

Start a scheduled task

Usage

```
startScheduler(schedulerId)
```

Arguments

schedulerId	id of the scheduler task to be run. Can be obtained with listScheduler().

Value

response status code (200 on success)

---

startUsecase *Start a new usecase on the platform.*

Description

Start a new usecase on the platform.

Usage

```
startUsecase(
  name,
  dataType,
  trainingType,
  datasetId,
  targetColumn = NULL,
  holdoutDatasetId = NULL,
  idColumn = NULL,
  dropList = NULL,
  profile = NULL,
  metric = NULL,
  foldColumn = NULL,
  normalModels = NULL,
  liteModels = c("XGB"),
  simpleModels = NULL,
  withBlend = NULL,
  weightColumn = NULL,
  featuresEngineeringSelectedList = NULL,
  featuresSelectionCount = NULL,
  featuresSelectionTime = NULL,
  datasetFolderId = NULL,
  filenameColumn = NULL,
  topColumn = NULL,
  bottomColumn = NULL,
  leftColumn = NULL,
  rightColumn = NULL,
  timeColumn = NULL,
  startDW = NULL,
  endDW = NULL,
  startFW = NULL,
  endFW = NULL,
  groupList = NULL,
  aprioriList = NULL
)
```

Arguments

name	name of the usecase.
dataType	type of data ("tabular" or "images" or "timeseries").
trainingType	type of the training you want to achieve ("regression", "classification", "multi-classification", "clustering", "object-detection").
datasetId	id of the dataset used for the training phase.
targetColumn	name of the TARGET column
holdoutDatasetId	id of the houldout dataset
idColumn	name of the id column
dropList	list of names of features to drop
profile	chosen profil among "quick", "normal", "advanced"
metric	name of the metric to optimise
foldColumn	name of the fold column
normalModels	list of (normal) models to select with full FE & hyperparameters search
liteModels	list of (lite) models to select with lite FE & default hyperparameters
simpleModels	list of simple models to select
withBlend	do we allow to include blend in the modelisation
weightColumn	name of the weight columns
featuresEngineeringSelectedList	list of feature engineering to select among "Counter", "Date", "freq", "text_tfidf", "text_word2vec", "text_embedding", "tenc", "poly", "pca", "kmean"
featuresSelectionCount	number of features to keep after the feature selection process
featuresSelectionTime	time budget in minutes of the feature selection process
datasetFolderId	id of the dataset fold (images)
filenameColumn	name of the file name path (images)
topColumn	name of the top column (object detection)
bottomColumn	name of the bottom column (object detection)
leftColumn	name of the left column (object detection)
rightColumn	name of the right column (object detection)
timeColumn	name of the time column (time series)
startDW	value of the start of derivative window (time series), should be a strict negative integer
endDW	value of the end of derivative window (time series), should be a negative integer greater than startDW
startFW	value of the start of forecast window (time series), should be a strict positive integer
endFW	value of the end of forecast window (time series), should be a strict positive integer greater than startFW
groupList	list of name of feature that describes groups (time series)
aprioriList	list of name of feature that are a priori (time series)

Value

usecase infos

`stopUsecase` *Stop a version of a usecase on the platform*

Description

Stop a version of a usecase on the platform

Usage

```
stopUsecase(usecaseId, versionNumber = 1)
```

Arguments

`usecaseId`	id of the usecase, can be obtained with getUsecases().
`versionNumber`	number of the version of the usecase. 1 by default.

`stores_test` *Stores sales test*

Description

Dataset for time series with target "Sales" and group variable "Store"

Usage

```
stores_test
```

Format

A data.frame with 118 rows and 6 variables

Examples

```
stores_test
```

stores_train *Stores sales train*

Description

Dataset for time series with target "Sales" and group variable "Store"

Usage

```
stores_train
```

Format

A data.frame with 1766 rows and 6 variables

Examples

```
stores_train
```

telecom *Telecom*

Description

This dataset comes from the website kaggle <https://www.kaggle.com/blastchar/telco-customer-churn> Dataset for clustering

Usage

```
telecom
```

Format

A data.frame with 7032 rows and 22 variables

Examples

```
telecom
```

testConnector *Test a connector.*

Description

Test a connector.

Usage

```
testConnector(connectorId)
```

Arguments

connectorId	id of the connector to be tested, can be obtained with getConnectors().

testDatasource *Test a datasource*

Description

Test a datasource

Usage

```
testDatasource(datasourceId)
```

Arguments

datasourceId	id of the datasource to be tested, can be obtained with getDatasources().

titanic *Titanic*

Description

This dataset comes from the website kaggle <https://www.kaggle.com/c/titanic/data> Dataset for classification with target "Survived"

Usage

```
titanic
```

Format

A data.frame with 891 rows and 6 variables

Examples

```
titanic
```

unshareUsecase *Unshare a use case for the specified email adress. If missing, the use-case will be unshared from all users*

Description

Unshare a use case for the specified email adress. If missing, the usecase will be unshared from all users

Usage

```
unshareUsecase(usecaseId, email = NULL)
```

Arguments

usecaseId	id of the usecase, can be obtained with getUsecases().
email	email adress of the user you want to unshare the usecase with. Should be an instance user.

Value

list of shared users

updateAppName *Update an app name.*

Description

Update an app name.

Usage

```
updateAppName(appId, appName)
```

Arguments

appId	id of the app to be updated, can be obtained with listApps().
appName	new name of the app to be set.

Value

response status code (200 on success)

updateUsecaseDescription

Update the description of a given usecase and its version number

Description

Update the description of a given usecase and its version number

Usage

```
updateUsecaseDescription(usecaseId, description = "", versionNumber = 1)
```

Arguments

usecaseId	id of the usecase, can be obtained with getUsecases().
description	Description of the usecase.
versionNumber	number of the version of the usecase. 1 by default.

wines

Wines quality

Description

This dataset comes from the website kaggle <https://www.kaggle.com/piyushgoyal443/red-wine-dataset> Dataset for multiclassification with target "quality"

Usage

```
wines
```

Format

A data.frame with 1483 rows and 12 variables

Examples

```
wines
```

Index