Package 'datarobot'

December 1, 2016

Title DataRobot Predictive Modeling API

Version 2.4.0

Description
For working with the DataRobot predictive modeling platform's <https://app.datarobot.com> API.

Depends R (>= 3.1.1), methods

Imports httr (>= 1.0), jsonlite (>= 0.9.19), yaml (>= 2.1.13),

License MIT + file LICENSE

LazyData true

Author Ron Pearson [aut], Zachary Deane-Mayer [aut], David Chudzicki [aut], Dallin Akagi [aut]

Maintainer David Chudzicki <api-maintainer@datarobot.com>

Suggests knitr, MASS, testthat, beanplot, mlbench, car, rmarkdown,
insuranceData, doBy, lintr, stubthat

VignetteBuilder rmarkdown, knitr

RoxygenNote 5.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2016-12-01 20:08:55

R topics documented:

`datarobot-package` *DataRobot Predictive Modeling API*

Description

For working with the DataRobot predictive modeling platform's API.

`as.data.frame` *DataRobot S3 object methods for R's generic as.data.frame function*

Description

These functions extend R's generic as.data.frame function to the DataRobot S3 object classes listOfBlueprints, listOfFeaturelists, listOfModels, and projectSummaryList.

Usage

```
## S3 method for class 'listOfBlueprints'
as.data.frame(x, row.names = NULL,
  optional = FALSE, ...)

## S3 method for class 'listOfFeaturelists'
as.data.frame(x, row.names = NULL,
  optional = FALSE, ...)

## S3 method for class 'listOfModels'
as.data.frame(x, row.names = NULL, optional = FALSE,
  simple = TRUE, ...)

## S3 method for class 'projectSummaryList'
as.data.frame(x, row.names = NULL,
  optional = FALSE, simple = TRUE, ...)

## S3 method for class 'listOfDataRobotPredictionDatasets'
as.data.frame(x, row.names = NULL,
  optional = FALSE, ...)
```

Arguments

x	S3 object to be converted into a dataframe.
row.names	Optional row names for the dataframe returned by the method.
optional	Optional logical variable; if TRUE, setting row names and converting column names to syntactic names: see help for make.names function.
simple	Optional logical variable; if TRUE (the default), a simplified dataframe is returned for objects of class listOfModels or projectSummaryList.
...	Additional optional parameters to be passed to the generic as.data.frame function (not used at present).

Details

All of the DataRobot S3 'listOf' class objects have relatively complex structures and are often easier to work with as dataframes. The methods described here extend R's generic as.data.frame function to convert objects of these classes to convenient dataframes. For objects of class listOfBlueprints and listOfFeaturelists or objects of class listOfModels and projectSummaryList with simple = FALSE, the dataframes contain all information from the original S3 object. The default value simple = TRUE provides simpler dataframes for objects of class listOfModels and projectSummaryList.

Value

A dataframe containing some or all of the data from the original S3 object; see Details.

AutopilotMode *Autopilot modes*

Description

This is a list that contains the valid values for autopilot mode. If you wish, you can specify autopilot modes using the list values, e.g. AutopilotMode$FullAuto instead of typing the string 'auto'. This way you can benefit from autocomplete and not have to remember the valid optons.

Usage

```
AutopilotMode
```

Format

An object of class `list` of length 4.

ConnectToDataRobot *Establish a connection to the DataRobot modeling engine*

Description

This function initializes a DataRobot session. If a (YAML) config file (with keys for endpoint and token) is placed at $HOME/.config/datarobot/drconfig.yaml, then we attempt to establish a connection to DataRobot when the package loads, so (if successful) this function does not need to be called.

Usage

```
ConnectToDataRobot(endpoint = NULL, token = NULL, username = NULL,
  password = NULL, configPath = NULL)
```

Arguments

`endpoint`	URL specifying the DataRobot server to be used. It depends on DataRobot modeling engine implementation (cloud-based, on-prem...) you are using. Contact your DataRobot admin for endpoint to use and to turn on API access to your account. The endpoint for DataRobot cloud accounts is https://app.datarobot.com/api/v2
`token`	DataRobot API access token. It is unique for each DataRobot modeling engine account and can be accessed using DataRobot webapp in Account profile section.
`username`	(no longer supported)
`password`	(no longer supported)
`configPath`	Path to YAML config file specifying configuration (token and endpoint)

Details

The function creates the environment variables "DataRobot_URL" and "DataRobot_Token" used by other functions to access the DataRobot modeling engine.

`CreateDerivedFeatures` *Derived Features*

Description

These functions request that new features be created as transformations of existing features and wait for the new feature to be created.

Usage

```
CreateDerivedFeatureAsCategorical(project, parentName, name = NULL,
  dateExtraction = NULL, replacement = NULL, maxWait = 60)

CreateDerivedFeatureAsText(project, parentName, name = NULL,
  dateExtraction = NULL, replacement = NULL, maxWait = 60)

CreateDerivedFeatureAsNumeric(project, parentName, name = NULL,
  dateExtraction = NULL, replacement = NULL, maxWait = 60)
```

Arguments

`project`	Either (1) a character string giving the unique alphanumeric identifier for the project, or (2) a list containing the element projectId with this identifier.
`parentName`	The name of the parent feature.
`name`	The name of the new feature.
`dateExtraction`	dateExtraction: The value to extract from the date column: 'year', 'yearDay', 'month', 'monthDay', 'week', or 'weekDay'. Required for transformation of a date column. Otherwise must not be provided.
`replacement`	The replacement in case of a failed transformation. Optional.
`maxWait`	The maximum time (in seconds) to wait for feature creation.

Value

Details for the created feature; same schema as the object returned from GetFeatureInfo.

`CreateFeaturelist` *Create a new featurelist in a DataRobot project*

Description

This function allows the user to create a new featurelist in a project by specifying its name and a list of variables to be included

Usage

```
CreateFeaturelist(project, listName, featureNames)
```

Arguments

`project`	Either (1) a character string giving the unique alphanumeric identifier for the project, or (2) a list containing the element projectId with this identifier.
`listName`	Character string identifying the new featurelist to be created.
`featureNames`	Character vector listing the names of the variables to be included in the featurelist.

Details

DataRobot featurelists define the variables from the modeling dataset used in fitting each project model. Some functions (SetTarget, StartNewAutopilot) optionally accept a featurelist (and use a default featurelist if none is specified).

Value

A list with the following four elements describing the featurelist created:

featurelistId Character string giving the unique alphanumeric identifier for the new featurelist.

projectId Character string giving the projectId identifying the project to which the featurelist was added.

features Character vector with the names of the variables included in the new featurelist.

name Character string giving the name of the new featurelist.

`CreateGroupPartition` *Create a group-based S3 object of class partition for the SetTarget function*

Description

Group partitioning constructs data partitions such that all records with each level in the column or columns specified by the parameter partitionKeyCols occurs together in the same partition.

Usage

```
CreateGroupPartition(validationType, holdoutPct, partitionKeyCols,
  reps = NULL, validationPct = NULL)
```

Arguments

`validationType`	Character string specifying the type of partition generated, either 'TVH' or 'CV'.
`holdoutPct`	Integer, giving the percentage of data to be used as the holdout subset.
`partitionKeyCols`	Character string specifying the name or names of the variables used in defining the group partition.
`reps`	Integer, specifying the number of cross-validation folds to generate; only applicable when validationType = 'CV'.
`validationPct`	Integer, giving the percentage of data to be used as the validation subset.

Details

This function is one of several convenience functions provided to simplify the task of starting modeling projects with custom partitioning options. The other functions are CreateRandomPartition, CreateStratifiedPartition, and CreateUserPartition.

Value

An S3 object of class 'partition' including the parameters required by the SetTarget function to generate a group-based partitioning of the modeling dataset.

CreatePrimeCode *Create and validate the downloadable code for the ruleset associated with this model*

Description

Create and validate the downloadable code for the ruleset associated with this model

Usage

```
CreatePrimeCode(project, primeModelId, language)
```

Arguments

project	Either (1) a character string giving the unique alphanumeric identifier for the project, or (2) a list containing the element projectId with this identifier.
primeModelId	String. Id returned by GetPrimeModel(s) functions
language	Programming language to use for downloadable code (see PrimeLanguage)

Value

job Id

CreateRandomPartition *Create a random sampling-based S3 object of class partition for the SetTarget function*

Description

Random partitioning is supported for either Training/Validation/Holdout ('TVH') or cross-validation ('CV') splits. In either case, the holdout percentage (holdoutPct) must be specified; for the 'CV' method, the number of cross-validation folds (reps) must also be specified, while for the 'TVH' method, the validation subset percentage (validationPct) must be specified.

Usage

```
CreateRandomPartition(validationType, holdoutPct, reps = NULL,
  validationPct = NULL)
```

Arguments

validationType	Character string specifying the type of partition generated, either 'TVH' or 'CV'.
holdoutPct	Integer, giving the percentage of data to be used as the holdout subset.
reps	Integer, specifying the number of cross-validation folds to generate; only applicable when validationType = 'CV'.
validationPct	Integer, giving the percentage of data to be used as the validation subset.

Details

This function is one of several convenience functions provided to simplify the task of starting modeling projects with custom partitioning options. The other five functions are CreateGroupPartition, CreateStratifiedPartition, and CreateUserPartition.

Value

An S3 object of class partition including the parameters required by SetTarget to generate a random partitioning of the modeling dataset.

CreateStratifiedPartition

Create a stratified sampling-based S3 object of class partition for the SetTarget function

---

Description

Stratified partitioning is supported for binary classification problems and it randomly partitions the modeling data, keeping the percentage of positive class observations in each partition the same as in the original dataset. Stratified partitioning is supported for either Training/Validation/Holdout ('TVH') or cross-validation ('CV') splits. In either case, the holdout percentage (holdoutPct) must be specified; for the 'CV' method, the number of cross-validation folds (reps) must also be specified, while for the 'TVH' method, the validation subset percentage (validationPct) must be specified.

Usage

```
CreateStratifiedPartition(validationType, holdoutPct, reps = NULL,
  validationPct = NULL)
```

Arguments

`validationType`	Character string specifying the type of partition generated, either 'TVH' or 'CV'.
`holdoutPct`	Integer, giving the percentage of data to be used as the holdout subset.
`reps`	Integer, specifying the number of cross-validation folds to generate; only applicable when validationType = 'CV'.
`validationPct`	Integer, giving the percentage of data to be used as the validation subset.

Details

This function is one of several convenience functions provided to simplify the task of starting modeling projects with custom partitioning options. The other functions are CreateGroupPartition, CreateRandomPartition, and CreateUserPartition.

Value

An S3 object of class 'partition' including the parameters required by the SetTarget function to generate a stratified partitioning of the modeling dataset.

`CreateUserPartition` *Create a user-defined S3 object of class partition for the SetTarget function*

Description

Creates a list object used by the SetTarget function to specify either Training/Validation/Holdout (validationType = 'TVH') or cross-validation (validationType = 'CV') partitions of the modeling dataset based on the values included in a column from the dataset. In either case, the name of this data column must be specified (as userPartitionCol). For the 'TVH' option, the column must have either exactly 3 values (in which case the values used to specify each level must be given) or exactly 2 values (in which case training and validation levels should be specified, but), while for the 'CV' option, only the level that specifies the holdout subset must be given.

Usage

```
CreateUserPartition(validationType, userPartitionCol, cvHoldoutLevel = NULL,
  trainingLevel = NULL, holdoutLevel = NULL, validationLevel = NULL)
```

Arguments

`validationType`	Character string specifying the type of partition generated, either 'TVH' or 'CV'.
`userPartitionCol`	Character string naming the data column from the modeling dataset containing the subset designations.
`cvHoldoutLevel`	Data value from userPartitionCol that identifies the holdout subset under the 'CV' option.
`trainingLevel`	Data value from userPartitionCol that identifies the training subset under the 'TVH' option.
`holdoutLevel`	Data value from userPartitionCol that identifies the holdout subset under the 'TVH' option.
`validationLevel`	Data value from userPartitionCol that identifies the validation subset under the 'TVH' option.

Details

This function is one of several convenience functions provided to simplify the task of starting modeling projects with custom partitioning options. The other functions are CreateGroupPartition, CreateRandomPartition, and CreateStratifiedPartition.

Value

An S3 object of class 'partition' including the parameters required by the SetTarget function to generate a user-specified of the modeling dataset.

DeleteJob *Cancel a running job*

Description

Cancel a running job

Usage

```
DeleteJob(job)
```

Arguments

job	The job you want to cancel (one of the items in the list returned from ListJobs)

DeleteModel *Delete a specified DataRobot model*

Description

This function removes the model specified by the parameter model from its associated project.

Usage

```
DeleteModel(model)
```

Arguments

model	An S3 object of class dataRobotModel like that returned by the function GetModelObject, or each element of the list returned by the function GetAllModels.

DeleteModelJob *Delete a model job from the modeling queue*

Description

This function deletes the modeling job specified by modelJobId from the DataRobot modeling queue.

Usage

```
DeleteModelJob(project, modelJobId)
```

Arguments

project	Either (1) a character string giving the unique alphanumeric identifier for the project, or (2) a list containing the element projectId with this identifier.
modelJobId	Integer, identifier for the modeling job to be deleted; can be obtained from the results returned by the function GetModelJobs.

DeletePredictionDataset

Delete a specified prediction dataset

Description

This function removes a prediction dataset

Usage

```
DeletePredictionDataset(project, datasetId)
```

Arguments

project	Either (1) a character string giving the unique alphanumeric identifier for the project, or (2) a list containing the element projectId with this identifier.
datasetId	The id of the dataset to delete

DeletePredictJob *Function to delete one predict job from the DataRobot queue*

Description

This function deletes the predict job specified by predictJobId from the DataRobot queue.

Usage

```
DeletePredictJob(project, predictJobId)
```

Arguments

project	Either (1) a character string giving the unique alphanumeric identifier for the project, or (2) a list containing the element projectId with this identifier.
predictJobId	Integer, identifying the prediction job created by the call to RequestPredictions.

Value

Logical TRUE and displays a message to the user if the delete request was successful; otherwise, execution halts and an error message is displayed.

DeleteProject *Delete a specified element from the DataRobot project list*

Description

This function deletes the project defined by project, described under Arguments. This parameter may be obtained in several ways, including: (1), as one of the projectId elements of the list returned by GetProjectList; (2), as the S3 object returned by the GetProject function; or (3), as the list returned by the SetupProject function.

Usage

```
DeleteProject(project)
```

Arguments

project	Either (1) a character string giving the unique alphanumeric identifier for the project, or (2) a list containing the element projectId with this identifier.

DeleteTransferrableModel
Delete this imported model.

Description

Delete this imported model.

Usage

```
DeleteTransferrableModel(importId)
```

Arguments

importId	Character string unique id of the import

DownloadPrimeCode *Download the code of DataRobot Prime model and save it to a file*

Description

Training a model using a ruleset is a necessary prerequisite for being able to download the code for a ruleset

Usage

```
DownloadPrimeCode(project, primeFileId, filepath)
```

Arguments

project	Either (1) a character string giving the unique alphanumeric identifier for the project, or (2) a list containing the element projectId with this identifier.
primeFileId	Prime file Id (can be aquired using ListPrimeFiles function)
filepath	String. The location to save the file to

DownloadTransferrableModel *Download an transferrable model file for use in an on-premise DataRobot standalone prediction environment. This function can only be used if model export is enabled, and will only be useful if you have an on-premise environment in which to import it.*

Description

Download an transferrable model file for use in an on-premise DataRobot standalone prediction environment. This function can only be used if model export is enabled, and will only be useful if you have an on-premise environment in which to import it.

Usage

```
DownloadTransferrableModel(project, modelId, modelFile)
```

Arguments

project	Either (1) a character string giving the unique alphanumeric identifier for the project, or (2) a list containing the element projectId with this identifier.
modelId	Unique alphanumeric identifier for the model of interest.
modelFile	file name to be use for tranferrable model

FeatureFromAsyncUrl *Retrieve a feature from the creation URL*

Description

If feature creation times out, the error message includes a URL corresponding to the creation task. That URL can be passed to this function (which will return the feature details when finished) to resume waiting for feature creation.

Usage

```
FeatureFromAsyncUrl(asyncUrl, maxWait = 60)
```

Arguments

asyncUrl	The temporary status URL
maxWait	The maximum time to wait (in seconds) for project creation before aborting.

GetAllModels *Retrieve all available model information for a DataRobot project*

Description

This function requests the model information for the DataRobot project specified by the project argument, described under Arguments. This parameter may be obtained in several ways, including: (1), from the projectId element of the list returned by GetProjectList; (2), as the object returned by the GetProject function; or (3), as the list returned by the SetupProject function. The function returns an S3 object of class 'listOfModels'.

Usage

```
GetAllModels(project)
```

Arguments

project	Either (1) a character string giving the unique alphanumeric identifier for the project, or (2) a list containing the element projectId with this identifier.

Value

An S3 object of class listOfModels, which may be characterized using R's generic summary function or converted to a dataframe with the as.data.frame method.

GetFeatureImpactForJobId *Retrieve completed Feature Impact results given a job ID*

Description

This will wait for the Feature Impact job to be completed (giving an error if the job is not a Feature Impact job and an error if the job errors).

Usage

```
GetFeatureImpactForJobId(project, jobId, maxWait = 60)
```

Arguments

project	The project the Feature Impact is part of.
jobId	The ID of the job (e.g. as returned from RequestFeatureImpact)
maxWait	Integer, The maximum time (in seconds) to wait for the model job to complete

Value

A data frame with the following columns:

featureName The name of the feature

impactNormalized The normalized impact score (largest value is 1)

impactUnnormalized The unnormalized impact score

Examples

```
## Not run:
model <- GetAllModels(project)[[1]]
featureImpactJobId <- RequestFeatureImpact(model)
featureImpact <- GetFeatureImpactForJobId(project, featureImpactJobId)

## End(Not run)
```

`GetFeatureImpactForModel` *Retrieve completed Feature Impact results given a model*

Description

This will only succeed if the Feature Impact computation has completed.

Usage

```
GetFeatureImpactForModel(model)
```

Arguments

`model`	The model for which you want to retrieve Feature Impact

Details

Feature Impact is computed for each column by creating new data with that column randomly permuted (but the others left unchanged), and seeing how the error metric score for the predictions is affected. The 'impactUnnormalized' is how much worse the error metric score is when making predictions on this modified data. The 'impactNormalized' is normalized so that the largest value is 1. In both cases, larger values indicate more important features. Elsewhere this technique is sometimes called 'Permutation Importance'.

Value

A data frame with the following columns:

featureName The name of the feature

impactNormalized The normalized impact score (largest value is 1)

impactUnnormalized The unnormalized impact score

Examples

```
## Not run:
model <- GetAllModels(project)[[1]]
featureImpactJobId <- RequestFeatureImpact(model)
# Note: This will only work after the feature impact job has completed. Use
#       GetFeatureImpactFromIobId to automatically wait for the job.\
featureImpact <- GetFeatureImpactForModel(model)

## End(Not run)
```

`GetFeatureInfo` *Details about a feature*

Description

Details about a feature

Usage

```
GetFeatureInfo(project, featureName)
```

Arguments

`project`	Either (1) a character string giving the unique alphanumeric identifier for the project, or (2) a list containing the element projectId with this identifier.
`featureName`	id of the feature to be retrieve. Note: DataRobot renames some features, so the feature name may not be the one from your original data. You can use ListFeatureInfo to list the features and check the name. Deprecation note: If the name given does not match any feature names, we will treat it as an integer feature ID, and return the feature with the matching ID (if any). This is for backwards-compatibility and will be removed in v3.0.

Value

A named list which contains:

id feature id - note: Throughout the API, features are specified using their names, not this ID.

name feature name

featureType feature type: 'Numeric', 'Categorical', etc.

importance numeric measure of the strength of relationship between the feature and target (independent of any model or other features).

lowInformation whether feature has too few values to be informative

unique_count number of unique values

naCount number of missing values

dateFormat format of the feature if it is date-time feature

`GetFeaturelist` *Retrieve a specific featurelist from a DataRobot project*

Description

This function returns information about and the contents of a specified featurelist from a specified project.

Usage

```
GetFeaturelist(project, featurelistId)
```

Arguments

`project`	Either (1) a character string giving the unique alphanumeric identifier for the project, or (2) a list containing the element projectId with this identifier.
`featurelistId`	Unique alphanumeric identifier for the featurelist to be retrieved.

Details

DataRobot featurelists define the variables from the modeling dataset used in fitting each project model. In most cases, the same featurelist is used in fitting all project models, but models can be fit using alternative featurelists using the RequestNewModel function. To do this, featurelistId is required, and this is one of the elements returned by the GetFeaturelist function.

DataRobot featurelists define the variables from the modeling dataset used in fitting each project model. In most cases, the same featurelist is used in fitting all project models, but models can be fit using alternative featurelists using the RequestNewModel function. To do this, featurelistId is required, and this is one of the elements returned by the GetFeaturelist function.

Value

A list with the following four elements describing the requested featurelist:

featurelistId Character string giving the unique alphanumeric identifier for the featurelist

projectId Character string identifying the project to which the featurelist belongs

features Character vector with the names of the variables included in the featurelist

name Character string giving the name of the featurelist

GetModelFromJobId *Retrieve a new or updated model defined by modelJobId*

Description

The functions RequestNewModel and RequestSampleSizeUpdate initiate the creation of new models in a DataRobot project. Both functions submit requests to the DataRobot modeling engine and return an integer-valued modelJobId. The GetModelFromJobId function polls the modeling engine until the model has been built or a specified time limit is exceeded, returning an S3 object of class 'dataRobotModel' when the model is available.

Usage

```
GetModelFromJobId(project, modelJobId, maxWait = 60)
```

Arguments

project	Either (1) a character string giving the unique alphanumeric identifier for the project, or (2) a list containing the element projectId with this identifier.
modelJobId	The integer returned by either RequestNewModel or RequestSampleSizeUpdate.
maxWait	Integer, The maximum time (in seconds) to wait for the model job to complete

Details

Motivation for this function is the fact that some models - e.g., very complex machine learning models fit to large datasets - may take a long time to complete. Splitting the model creation request from model retrieval in these cases allows the user to perform other interactive R session tasks between the time the model creation/update request is made and the time the final model is available.

Value

An S3 object of class 'dataRobotModel' summarizing all available information about the model.

GetModelJobs *Retrieve status of Autopilot modeling jobs that are not complete*

Description

This function requests information on DataRobot Autopilot modeling tasks that are not complete, for one of three reasons: the task is running and has not yet completed; the task is queued and has not yet been started; or, the task has terminated due to an error.

Usage

```
GetModelJobs(project, status = NULL)
```

Arguments

`project`	Either (1) a character string giving the unique alphanumeric identifier for the project, or (2) a list containing the element projectId with this identifier.
`status`	The status of the desired jobs: one of JobStatus$Queue, JobStatus$InProgress, or JobStatus$Error. If NULL (default), queued and inprogress jobs are returned.

Details

The jobStatus variable specifies which of the three groups of modeling tasks is of interest. Specifically, if jobStatus has the value 'inprogress', the request returns information about modeling tasks that are running but not yet complete; if jobStatus has the value 'queue', the request returns information about modeling tasks that are scheduled to run but have not yet started; if jobStatus has the value 'error', the request returns information about modeling tasks that have terminated due to an error. By default, jobStatus is NULL, which means jobs with status "inprogress" or "queue" are returned, but not those with status "error".

Value

A list of lists with one element for each modeling task in the group being queried; if there are no tasks in the class being queried, an empty list is returned. If the group is not empty, a list is returned with the following nine elements:

status Prediction job status; one of JobStatus$Queue, JobStatus$InProgress, or JobStatus$Error

processes List of character vectors describing any preprocessing applied, possibly along with the model type

projectId Character string giving the unique identifier for the project

samplePct Numeric: the percentage of the dataset used in constructing the training dataset for model building

modelType Character string specifying the model type

modelCategory Character string: what kind of model this is - 'prime' for DataRobot Prime models, 'blend' for blender models, and 'model' for other models

featurelistId Character string identifying the featurelist used in fitting the model (derived from the modeling dataset)

blueprintId Character string identifying the DataRobot blueprint on which the model is based

modelJobId Character: id of the job

modelId Character string uniquely identifying the model

`GetModelObject` *Retrieve the details of a specified model*

Description

This function returns a DataRobot S3 object of class dataRobotModel for the model defined by project and modelId.

Usage

```
GetModelObject(project, modelId)
```

Arguments

`project`	Either (1) a character string giving the unique alphanumeric identifier for the project, or (2) a list containing the element projectId with this identifier.
`modelId`	Unique alphanumeric identifier for the model of interest.

Details

The S3 object returned by this function is required by the functions DeleteModel, ListModelFeatures, and RequestSampleSizeUpdate.

Value

An S3 object of class 'dataRobotModel', which is a list with the following 13 components:

featurelistId Character string: unique alphanumeric identifier for the featurelist on which the model is based

processes Character vector with components describing preprocessing; may include modelType

featurelistName Character string giving the name of the featurelist on which the model is based

projectId Character string giving the unique alphanumeric identifier for the project

samplePct Numeric: percentage of the dataset used to form the training dataset for model fitting

isFrozen Logical : is model created with frozen tuning parameters

modelType Character string describing the model type

metrics List with one element for each valid metric associated with the model. Each element is a list with elements for each possible evaluation type (holdout, validation, and crossValidation)

modelCategory Character string giving model category (e.g., blend, model)

blueprintId Character string giving the unique DataRobot blueprint identifier on which the model is based

modelId Character string giving the unique alphanumeric model identifier

projectName Character string: optional description of project defined by projectId

projectTarget Character string defining the target variable predicted by all models in the project

projectMetric Character string defining the fitting metric optimized by all project models

`GetPredictions` *Retrieve model predictions from predictJobId*

Description

This function is called with a project descriptor and an integer predictJobId, obtained from an earlier call to RequestPredictions. It returns the predictions generated for the model and data specified in this prior function call.

Usage

```
GetPredictions(project, predictJobId, type = "response", maxWait = 60)
```

Arguments

`project`	Either (1) a character string giving the unique alphanumeric identifier for the project, or (2) a list containing the element projectId with this identifier.
`predictJobId`	Integer, identifying the prediction job created by the call to RequestPredictions.
`type`	Character string, specifying the type of response for binary classifiers; see Details.
`maxWait`	Integer, The maximum time (in seconds) to wait for the prediction job to complete

Details

The contents of the return vector depends on both the modeling task - binary classification or regression - and the value of the type parameter. For regression tasks, the type parameter is ignored and a vector of numerical predictions of the response variable is returned. For binary classification tasks, either a vector of predicted responses is returned if type has the value "response" (the default), or a vector of probabilities for the positive class is returned, if type is "probability".

This function will error if the requested job has errored, or if it isn't complete within maxWait seconds.

Value

Vector of predictions, depending on the modeling task ("Binary" or "Regression") and the value of the type parameter; see Details.

GetPredictJobs *Function to list all prediction jobs in a project*

Description

Function to list all prediction jobs in a project

Usage

```
GetPredictJobs(project, status = NULL)
```

Arguments

project	Either (1) a character string giving the unique alphanumeric identifier for the project, or (2) a list containing the element projectId with this identifier.
status	The status of the desired jobs: one of JobStatus$Queue, JobStatus$InProgress, or JobStatus$Error. If NULL (default), queued and inprogress jobs are returned.

Value

Dataframe with one row for each prediction job in the queue, with the following columns:

status Prediction job status; one of JobStatus$Queue, JobStatus$InProgress, or JobStatus$Error

predictJobId Character string specifying the job id

modelId Character string specifying the model from which predictions have been requested.

projectId Character string specifying the project that contains the model.

GetPrimeEligibility *Check if model can be approximated with DataRobot Prime*

Description

This function returns list with two members: canMakePrime : logical value. TRUE if model can be approximated using DataRobot Prime, FALSE if model can not be approximated message : character. Provides information why model may not be approximated with DataRobot Prime

Usage

```
GetPrimeEligibility(project, modelId)
```

Arguments

project	Either (1) a character string giving the unique alphanumeric identifier for the project, or (2) a list containing the element projectId with this identifier.
modelId	Unique alphanumeric identifier for the model of interest.

Value

list with two members: canMakePrime : logical value. TRUE if model can be approximated using DataRobot Prime, FALSE if model can not be approximated message : character. Provides information why model may not be approximated with DataRobot Prime

---

`GetPrimeFile` *Retrieve a specific Prime file from a DataRobot project*

Description

This function returns information about specified Prime file from a specified project.

Usage

```
GetPrimeFile(project, primeFileId)
```

Arguments

`project`	Either (1) a character string giving the unique alphanumeric identifier for the project, or (2) a list containing the element projectId with this identifier.
`primeFileId`	Unique alphanumeric identifier for the primeFile to be retrieved.

Value

List with following elements:

language Character string. Code programming language

isValid logical flag indicating if code passed validation

rulesetId Integer identifier for the ruleset

parentModelId Unique alphanumeric identifier for the parent model

projectId Unique alphanumeric identifier for the project

id Unique alphanumeric identifier for the Prime file

modelId Unique alphanumeric identifier for the model

`GetPrimeFileFromJobId` *Retrieve a specific Prime file from a DataRobot project for corresponding jobId*

Description

Retrieve a specific Prime file from a DataRobot project for corresponding jobId

Usage

```
GetPrimeFileFromJobId(project, jobId, maxWait = 60)
```

Arguments

`project`	Either (1) a character string giving the unique alphanumeric identifier for the project, or (2) a list containing the element projectId with this identifier.
`jobId`	Unique integer identifier (return for example by RequestPrimeModel)
`maxWait`	maximum time to wait (in sec) before job completed

Value

List with following elements:

language Character string. Code programming language

isValid logical flag indicating if code passed validation

rulesetId Integer identifier for the ruleset

parentModelId Unique alphanumeric identifier for the parent model

projectId Unique alphanumeric identifier for the project

id Unique alphanumeric identifier for the Prime file

modelId Unique alphanumeric identifier for the model

`GetPrimeModel` *Retrieve information about specified DataRobot Prime model*

Description

This function requests the DataRobot Prime model information for the DataRobot project specified by the project argument, and modelId The function returns list containing informatione about specified DataRobot Prime model

Usage

```
GetPrimeModel(project, modelId)
```

Arguments

project	Either (1) a character string giving the unique alphanumeric identifier for the project, or (2) a list containing the element projectId with this identifier.
modelId	Unique alphanumeric identifier for the model of interest.

Value

list containing informatione about specified DataRobot Prime model

---

GetPrimeModelFromJobId

Retrieve information about specified DataRobot Prime model using corresponding jobId

---

Description

Retrieve information about specified DataRobot Prime model using corresponding jobId

Usage

```
GetPrimeModelFromJobId(project, jobId, maxWait = 60)
```

Arguments

project	Either (1) a character string giving the unique alphanumeric identifier for the project, or (2) a list containing the element projectId with this identifier.
jobId	Unique integer identifier (return for example by RequestPrimeModel)
maxWait	maximum time to wait (in sec) before job completed

Value

list containing informatione about specified DataRobot Prime model

---

GetProject

Retrieve dtails about a specified DataRobot modeling project

Description

Returns a list of details about the DataRobot modeling project specified by project.

Usage

```
GetProject(project)
```

Arguments

project	Either (1) a character string giving the unique alphanumeric identifier for the project, or (2) a list containing the element projectId with this identifier.

Value

An S3 object of class 'dataRobotProject', consisting of the following 15 elements:

projectId Character string giving the unique project identifier

projectName Character string giving the name assigned to the project

fileName Character string giving the name of the modeling dataset for the project

stage Character string describing the stage of the DataRobot Autopilot

autopilotMode Numeric: 0 for fully automatic mode; 1 for semi-automatic mode; 2 for manual mode

created Character string representation of the project creation time and date

target Name of the target variable from fileName

metric Character string specifying the metric optimized by all project models

partition A 7-element list describing the data partitioning for model fitting and cross validation

recommender A 3-element list with information specific to recommender models

advancedOptions A 4-element list with advanced option specifications

positiveClass Character string: name of positive class for binary response models

maxTrainPct Maximum training subset percentage for models in this project

holdoutUnlocked A logical flag indicating whether the holdout dataset has been used for model evaluation

targetType Character string specifying the type of modeling problem (e.g., regression or binary classification)

GetProjectList *Retrieve a list of all DataRobot projects*

Description

This function returns an S3 object of class projectSummaryList that describes all DataRobot modeling projects available to the user. This list may be converted into a dataframe with the as.data.frame method for this class of S3 objects.

Usage

```
GetProjectList()
```

Value

An S3 object of class 'projectSummaryList', consisting of the following 15 elements:

projectId List of character strings giving the unique DataRobot identifier for each project

projectName List of character strings giving the user-supplied project names

fileName List of character strings giving the name of the modeling dataset for each project

stage List of character strings specifying each project's Autopilot stage (e.g., 'aim' is necessary to set target)

autopilotMode List of integers specifying the Autopilot mode (0 = fully automatic, 1 = semi-automatic, 2 = manual)

created List of character strings giving the project creation time and date

target List of character strings giving the name of the target variable for each project

metric List of character strings identifying the fitting metric optimized for each project

partition Dataframe with one row for each project and 12 columns specifying partitioning details

recommender Dataframe with one row for each project and 3 columns characterizing recommender projects

advancedOptions Dataframe with one row for each project and 4 columns specifying values for advanced option parameters

positiveClass Character string identifying the positive target class for binary classification projects

maxTrainPct List of integers specifying the maximum training set percentage possible for each project

holdoutUnlocked Logical flag indicating whether holdout subset results have been computed

targetType Character string giving the type of modeling project (e.g., regression or binary classification)

GetProjectStatus *Request Autopilot status for a specified DataRobot project*

Description

This function polls the DataRobot Autopilot for the status of the project specified by the project parameter.

Usage

```
GetProjectStatus(project)
```

Arguments

`project`	Either (1) a character string giving the unique alphanumeric identifier for the project, or (2) a list containing the element projectId with this identifier.

Value

List with the following three components:

autopilotDone Logical flag indicating whether the Autopilot has completed

stage Character string specifying the Autopilot stage

stageDescription Character string interpreting the Autopilot stage value

---

GetRecommendedBlueprints

Retrieve the list of available blueprints for a project

Description

(Deprecated in 2.3, will be removed in 3.0. Use ListBlueprints instead.)

Usage

```
GetRecommendedBlueprints(project)
```

Arguments

project	Either (1) a character string giving the unique alphanumeric identifier for the project, or (2) a list containing the element projectId with this identifier.

GetRulesets

List the rulesets approximating a model generated by DataRobot Prime

---

Description

This function wll return list of rulesets that could be used to approximate the specified model. Rulesets are created using the RequestApproximation function. If model hasn't been approximated yet, will return empty list

Usage

```
GetRulesets(project, modelId)
```

Arguments

project	Either (1) a character string giving the unique alphanumeric identifier for the project, or (2) a list containing the element projectId with this identifier.
modelId	Unique alphanumeric identifier for the model of interest.

Value

A list of lists with one element for each ruleset. If there are no rulesets created for a model then an empty list is returned If the group is not empty,a list is returned with the following elements:

projectId Character string giving the unique identifier for the project

rulesetId Integer number giving the identifier for the ruleset

score Score of ruleset (using project leaderboard metric)

parentModelId Character string giving the unique identifier for the parent model

ruleCount integer: number of rules in ruleset

modelId Character string giving the unique identifier for a model using the ruleset. May be NULL if no model using the ruleset has been created yet

---

`GetTransferrableModel` *Retrieve imported model info using import id*

Description

Retrieve imported model info using import id

Usage

```
GetTransferrableModel(importId)
```

Arguments

`importId`	Character string unique id of the import

Value

A list describing upladed transferrable model with the following components:

note Character string Manually added node about this imported model

datasetName Character string Filename of the dataset used to create the project the model belonged to

modelName Character string Model type describing the model generated by DataRobot

displayName Character string Manually specified human-readable name of the imported model

target Character string The target of the project the model belonged to prior to export

projectName Character string Name of the project the model belonged to prior to export

importedByUsername Character string Username of the user who imported the model

importedAt Character string The time the model was imported

version Numeric Project version of the project the model belonged to

projectId Character id of the project the model belonged to prior to export

featurelistName Character string Name of the featurelist used to train the model

createdByUsername Character string Username of the user who created the model prior to export

importedById Character string id of the user who imported the model

id Character string id of the import

createdById Character string id of the user who created the model prior to export

modelId Character string original id of the model prior to export

originUrl Character string URL

`GetValidMetrics` *Retrieve the valid fitting metrics for a specified project and target*

Description

For the response variable defined by the character string target and the project defined by the parameter project, return the vector of metric names that can be specified for fitting models in this project. This function is intended for use after SetupProject has been run but before SetTarget, allowing the user to specify valid non-default values for the metric parameter.

Usage

```
GetValidMetrics(project, target)
```

Arguments

`project`	Either (1) a character string giving the unique alphanumeric identifier for the project, or (2) a list containing the element projectId with this identifier.
`target`	Character string giving the name of the response variable to be predicted by all project models.

Value

Character vector containing the names of the metric values that are valid for a subsequent call to the SetTarget function.

`JobStatus` *Job statuses*

Description

This is a list that contains the valid values for job status when querying the list of jobs mode. If you wish, you can specify job status modes using the list values, e.g. JobStatus$InProgress instead of typing the string 'inprogress'. This way you can benefit from autocomplete and not have to remember the valid optons.

Usage

```
JobStatus
```

Format

An object of class `list` of length 5.

JobType *Job type*

Description

This is a list that contains the valid values for job type when querying the list of jobs.

Usage

```
JobType
```

Format

An object of class `list` of length 6.

ListBlueprints *Retrieve the list of available blueprints for a project*

Description

This function returns the list of available blueprints for a specified modeling project, as an S3 object of class listOfBlueprints; see Value.

Usage

```
ListBlueprints(project)
```

Arguments

`project`	Either (1) a character string giving the unique alphanumeric identifier for the project, or (2) a list containing the element projectId with this identifier.

Value

An S3 object of class 'listOfBlueprints', a list with one element for each recommended blueprint in the associated project. Each element of this list is itself a list with the following four components:

projectId Character string giving the unique DataRobot project identifier

processes List of character strings, identifying any preprocessing steps included in the blueprint

blueprintId Character string giving the unique DataRobot blueprint identifier

modelType Character string, specifying the type of model the blueprint builds

ListFeatureInfo *Details about all features for this project*

Description

Details about all features for this project

Usage

```
ListFeatureInfo(project)
```

Arguments

project	Either (1) a character string giving the unique alphanumeric identifier for the project, or (2) a list containing the element projectId with this identifier.

Value

A list of lists with one element for each feature The named list for each feature contains:

id feature id - note: Throughout the API, features are specified using their names, not this ID.

name feature name

featureType feature type: 'Numeric', 'Categorical', etc.

importance numeric measure of the strength of relationship between the feature and target (independent of any model or other features).

lowInformation whether feature has too few values to be informative

uniqueCount number of unique values

naCount number of missing values

dateFormat format of the feature if it is date-time feature

ListFeaturelists *Retrieve all featurelists associated with a project*

Description

This function returns an S3 object of class listOfFeaturelists that describes all featurelists (i.e., lists of modeling variables) available for the project specified by the project parameter. This list may be converted to a dataframe with the as.data.frame method for objects of class listOfFeaturelists.

Usage

```
ListFeaturelists(project)
```

Arguments

`project`	Either (1) a character string giving the unique alphanumeric identifier for the project, or (2) a list containing the element projectId with this identifier.

Value

An S3 object of class 'listOfFeaturelists', which is a list of dataframes: each element of the list corresponds to one featurelist associated with the project, and each dataframe has one row and the following four columns:

featurelistId Unique alphanumeric identifier for the featurelist

projectId Unique alphanumeric project identifier

features Comma-separated character string listing the variables included in the featurelist

name Character string giving the name of the featurelist

`ListJobs` *Retrieve information about jobs*

Description

This function requests information about the jobs that go through the DataRobot queue.

Usage

```
ListJobs(project, status = NULL)
```

Arguments

`project`	Either (1) a character string giving the unique alphanumeric identifier for the project, or (2) a list containing the element projectId with this identifier.
`status`	The status of the desired jobs: one of JobStatus$Queue, JobStatus$InProgress, or JobStatus$Error. If NULL (default), queued and inprogress jobs are returned.

Value

A list of lists with one element for each job. The named list for each job contains:

status job status ("inprogress", "queue", or "error")

url URL to request more detail about the job (character)

id job id (character).

jobType Job type. See JobType for valid values

projectId the id of the project that contains the model (character).

`ListModelFeatures` *Returns the list of features (i.e., variables) on which a specified model is based*

Description

This function returns the list of features (typically, response variable and raw covariates) used in building the model specified by model, an S3 object of class 'dataRobotModel'.

Usage

```
ListModelFeatures(model)
```

Arguments

`model`	An S3 object of class dataRobotModel like that returned by the function GetModelObject, or each element of the list returned by the function GetAllModels.

Value

A character vector of feature names, with one component for each model feature.

`ListPredictionDatasets` *Retrieve all prediction datasets associated with a project*

Description

This function returns an S3 object of class listDataRobotPredictionDataset that describes all prediction datasets available for the project specified by the project parameter. This list may be converted to a dataframe with the as.data.frame method for objects of class listDataRobotPredictionDataset.

Usage

```
ListPredictionDatasets(project)
```

Arguments

`project`	Either (1) a character string giving the unique alphanumeric identifier for the project, or (2) a list containing the element projectId with this identifier.

Value

An S3 object of class 'listDataRobotPredictionDataset', which is a list of dataframes: each element of the list corresponds to one prediction dataset associated with the project, and each dataframe has one row and the following 6 columns:

id Character: string giving the unique alphanumeric identifier for the dataset

numColumns Numeric: number of columns in dataset

name Character: Name of dataset file

created Character: Time of upload

projectId Character: string giving the unique alphanumeric identifier for the project

numRows Numeric: number of rows in dataset

`ListPrimeFiles` *List all downloadable code files from DataRobot Prime for the project*

Description

Training a model using a ruleset is a necessary prerequisite for being able to download the code for a ruleset

Usage

```
ListPrimeFiles(project, parentModelId = NULL, modelId = NULL)
```

Arguments

`project`	Either (1) a character string giving the unique alphanumeric identifier for the project, or (2) a list containing the element projectId with this identifier.
`parentModelId`	(optional) Filter for only those prime files approximating this parent model
`modelId`	(optional) Filter for only those prime files with code for this prime model

Value

List of lists. Each element of the list corresponds to one Prime file avalable to download. The elements of this list have the same format as the return value of GetPrimeFile

`ListPrimeModels` *Retrieve information about all DataRobot Prime models for a DataRobot project*

Description

This function requests the DataRobot Prime models information for the DataRobot project specified by the project argument, described under Arguments. The function returns data.frame containing informatione about each DataRobot Prime model in a project (one row per Prime model)

Usage

```
ListPrimeModels(project)
```

Arguments

`project`	Either (1) a character string giving the unique alphanumeric identifier for the project, or (2) a list containing the element projectId with this identifier.

Value

data.frame containing informatione about each DataRobot Prime model in a project (one row per Prime model)

`ListTransferrableModels` *Retrieve information about all imported models This function returns a data.frame that describes all imported models*

Description

Retrieve information about all imported models This function returns a data.frame that describes all imported models

Usage

```
ListTransferrableModels(limit = NULL, offset = NULL)
```

Arguments

`limit`	integer The number of records to return. The server will use a (possibly finite) default if not specified.
`offset`	integer The number of records to skip.

Value

A data.frame describing upladed transferrable model with the following components:

note Character string Manually added node about this imported model

datasetName Character string Filename of the dataset used to create the project the model belonged to

modelName Character string Model type describing the model generated by DataRobot

displayName Character string Manually specified human-readable name of the imported model

target Character string The target of the project the model belonged to prior to export

projectName Character string Name of the project the model belonged to prior to export

importedByUsername Character string Username of the user who imported the model

importedAt Character string The time the model was imported

version Numeric Project version of the project the model belonged to

projectId Character id of the project the model belonged to prior to export

featurelistName Character string Name of the featurelist used to train the model

createdByUsername Character string Username of the user who created the model prior to export

importedById Character string id of the user who imported the model

id Character string id of the import

createdById Character string id of the user who created the model prior to export

modelId Character string original id of the model prior to export

originUrl Character string URL

PauseQueue *Pause the DataRobot modeling queue*

Description

This function pauses the DataRobot modeling queue for a specified project

Usage

```
PauseQueue(project)
```

Arguments

`project`	Either (1) a character string giving the unique alphanumeric identifier for the project, or (2) a list containing the element projectId with this identifier.

plot.listOfModels *Plot method for DataRobot S3 objects of class listOfModels*

Description

Method for R's generic plot function for DataRobot S3 objects of class listOfModels. This function generates a horizontal barplot as described under Details.

Usage

```
## S3 method for class 'listOfModels'
plot(x, y, metric = NULL, pct = NULL,
  selectRecords = NULL, orderDecreasing = NULL, textSize = 0.8,
  textColor = "black", borderColor = "blue", xpos = NULL, ...)
```

Arguments

x	S3 object of class listOfModels to be plotted.
y	Not used; included for conformance with plot() generic function parameter requirements.
metric	Character string; defines the metric to be used in constructing the barplot. If NULL (the default), the validation set value for the project fitting metric is used; otherwise, this value must name one of the elements of the metrics list associated with each model in x.
pct	Integer; specifies a samplePct value used in selecting models to include in the barplot summary. If NULL (the default), all project models are included. Note, however, that this list of models is intersected with the list of models defined by the selectRecords parameter, so that only those models identified by both selectRecords and pct appear in the plot.
selectRecords	Integer vector; specifies the individual elements of the list x to be included in the barplot summary. If NULL (the default), all models are included. Note, however, that this list of models is intersected with the list of models defined by the pct parameter, so that only those models identified by both selectRecords and pct appear in the plot.
orderDecreasing	Logical; if TRUE, the barplot is built from the bottom up in decreasing order of the metric values; if FALSE, the barplot is built in increasing order of metric values. The default is NULL, which causes the plot to be generated in the order in which the models appear in the list x.
textSize	Numeric; multiplicative scaling factor for the model name labels on the barplot.
textColor	Character or character vector; if character, this parameter specifies the text color used in labelling all models in the barplot; if a character vector, it specifies one color for each model in the plot.
borderColor	Character; specifies the border color for all bars in the barplot, surrounding a transparent background.

xpos	Numeric or numeric vector; defines the horizontal position of the center of all text labels on the plot. The default is NULL, which causes all text to be centered in the plot; if xpos is a single number, all text labels are centered at this position; if xpos is a vector, it specifies one center position for each model in the plot.
...	Additional named parameters to be passed to R's barplot function used in generating the plot

Details

This function generates a horizontal barplot with one bar for each model characterized in the 'listOfModels' object x. The length of each bar is specified by the value of metric; if this parameter is specified as NULL (the default), the project fitting metric is used, as determined by the projectMetric value from the first element of x. Text is added to each bar in the plot, centered at the position specified by the xpos parameter, based on the value of the modelType element of each model in the list x. The size and color of these text labels may be controlled with the textSize and textColor parameters. The order in which these models appear on the plot is controlled by the choice of metric and the value of the orderDecreasing parameter, and subsets of the models appearing in the list x may be selected via the pct and selectRecords parameters.

Value

None. This function is called for its side-effect of generating a plot.

PostgreSQLdrivers *PostgreSQL drivers*

Description

This is a list that contains the valid values for PostgreSQL drivers.

Usage

```
PostgreSQLdrivers
```

Format

An object of class `list` of length 2.

PredictionDatasetFromAsyncUrl *Retrieve prediction dataset info from the dataset creation URL*

Description

If dataset creation times out, the error message includes a URL corresponding to the creation task. That URL can be passed to this function (which will return the completed dataset info details when finished) to resume waiting for creation.

Usage

```
PredictionDatasetFromAsyncUrl(asyncUrl, maxWait = 60)
```

Arguments

asyncUrl	The temporary status URL
maxWait	The maximum time to wait (in seconds) for creation before aborting.

PrimeLanguage *Prime Language*

Description

This is a list that contains the valid values for downloadable code programming languages.

Usage

```
PrimeLanguage
```

Format

An object of class list of length 2.

ProjectFromAsyncUrl *Retrieve a project from the project-creation URL*

Description

If project creation times out, the error message includes a URL corresponding to the project creation task. That URL can be passed to this function (which will return the completed project details when finished) to resume waiting for project creation.

Usage

```
ProjectFromAsyncUrl(asyncUrl, maxWait = 60)
```

Arguments

asyncUrl	The temporary status URL
maxWait	The maximum time to wait (in seconds) for project creation before aborting.

RequestApproximation *Request an approximation of a model using DataRobot Prime*

Description

This function wll create several rulesets that approximate the specified model. The code used in the approximation can be downloaded to be run locally. Currenly only Python and Java downloadable code is avalable

Usage

```
RequestApproximation(project, modelId)
```

Arguments

project	Either (1) a character string giving the unique alphanumeric identifier for the project, or (2) a list containing the element projectId with this identifier.
modelId	Unique alphanumeric identifier for the model of interest.

Details

General workflow of creating and downloading Prime code may look like following: RequestApproximation - create several rulestes that approximate the specified model GetRulesets - list all rulesests created for the parent model RequestPrimeModel - create Prime model for specified ruleset (use one of rulesets return by GetRulests) GetPrimeModelFromJobId - get PrimeModelId using JobId returned by RequestPrimeModel CreatePrimeCode - create code for one of avalable Prime models GetPrimeFileFromJobId - get PrimeFilelId using JobId returned by CreatePrimeCode DownloadPrimeCode - download specified Prime code file

Value

job Id

`RequestFeatureImpact` *Request Feature Impact to be computed.*

Description

This adds a Feature Impact job to the project queue.

Usage

```
RequestFeatureImpact(model)
```

Arguments

`model`	The model for which you want to compute Feature Impact, e.g. from the list of models returned by GetAllModels(project)

Value

A job ID (character)

Examples

```
## Not run:
model <- GetAllModels(project)[[1]]
featureImpactJobId <- RequestFeatureImpact(model)
featureImpact <- GetFeatureImpactForJobId(project, featureImpactJobId)

## End(Not run)
```

`RequestNewModel` *Adds a new model of type specified by blueprint to a DataRobot project*

Description

This function requests the creation of a new model in the DataRobot modeling project defined by the project parameter. The function also allows the user to specify alternatives to the project default for featurelist, samplePct, and scoringType. This function returns an integer modelJobId value, which can be used by the GetModelFromJobId function to return the full model object.

Usage

```
RequestNewModel(project, blueprint, featurelist = NULL, samplePct = NULL,
  scoringType = NULL)
```

Arguments

project	Either (1) a character string giving the unique alphanumeric identifier for the project, or (2) a list containing the element projectId with this identifier.
blueprint	A list with at least the following two elements: blueprintId and projectId. Note that the individual elements of the list returned by GetRecommendedBlueprints are admissible values for this parameter.
featurelist	A list that contains the element featurelistId that specifies the featurelist to be used in building the model; if not specified (i.e., for the default value NULL), the project default (Informative Features) is used.
samplePct	Numeric, specifying the percentage of the training dataset to be used in building the new model; if not specified (i.e., for the default value NULL), the maxTrainPct value for the project is used.
scoringType	Character string specifying the scoring type; default is validation set scoring, but cross-validation averaging is also possible.

Details

Motivation for this function is the fact that some models - e.g., very complex machine learning models fit to large datasets - may take a long time to complete. Splitting the model creation request from model retrieval in these cases allows the user to perform other interactive R session tasks between the time the model creation/update request is made and the time the final model is available.

Value

An integer value that can be used as the modelJobId parameter in subsequent calls to the GetModelFromJobId function.

RequestPredictions *Request predictions for model from newdata*

Description

This function uploads the data source defined by newdata and requests the predictions generated from this data source by model. The data source is specified in the same way as for the function SetupProject and can be either a CSV file or a dataframe containing the features on which model was built. Note that this function only requests the predictions be generated and returns a prediction job identifier to be used by the GetPredictions function to retrieve the predictions once they have been generated.

Usage

```
RequestPredictions(model, newdata)
```

Arguments

model	An S3 object of class dataRobotModel like that returned by the function GetModelObject, or each element of the list returned by the function GetAllModels.
newdata	Either (a) the name of a CSV file or (b) a dataframe; in either case, this parameter identifies the source of the data from which all model predictions will be generated. See Details.

Details

The DataRobot modeling engine requires a CSV file containing the data to be used in generating predictions, and this has been implemented here in two ways. The first and simpler is to specify dataSource as the name of this CSV file, but for the convenience of those who wish to work with dataframes, this function also provides the option of specifying a dataframe, which is then written to a CSV file and uploaded to the DataRobot server. In this case, the file name is either specified directly by the user through the saveFile parameter, or indirectly from the name of the dataSource dataframe if saveFile = NULL (the default). In this second case, the file name consists of the name of the dataSource dataframe with the string csvExtension appended.

Value

Integer predictJobId to be used by GetPredictions function to retrieve the model predictions.

RequestPredictionsForDataset

Request predictions against a previously uploaded dataset

Description

Request predictions against a previously uploaded dataset

Usage

```
RequestPredictionsForDataset(project, modelId, datasetId)
```

Arguments

project	Either (1) a character string giving the unique alphanumeric identifier for the project, or (2) a list containing the element projectId with this identifier.
modelId	The ID of the model to use to make predictions
datasetId	The ID of the dataset to make predictions against (as uploaded from UploadPredictionDataset)

Value

predictJobId to be used by GetPredictions function to retrieve the model predictions.

Examples

```
## Not run:
dataset <- UploadPredictionDataset(project, diamonds_small)
model <- GetAllModels(project)[[1]]
modelId <- model$modelId
predictJobId <- RequestPredictionsForDataset(project, modelId, ds$id)
predictions <- GetPredictions(project, predictJobId)

## End(Not run)
```

`RequestPrimeModel` *Request training for a DataRobot Prime model using a specified ruleset*

Description

Training a model using a ruleset is a necessary prerequisite for being able to download the code for a ruleset

Usage

```
RequestPrimeModel(project, ruleset)
```

Arguments

`project`	Either (1) a character string giving the unique alphanumeric identifier for the project, or (2) a list containing the element projectId with this identifier.
`ruleset`	A list specifying rulest parameters (see GetRulesets)

Value

job Id

`RequestSampleSizeUpdate` *Refits an existing model to a different fraction of the training dataset*

Description

This function requests a refit of the model defined by the model parameter to the same training dataset used in building it originally, but with a different fraction of the data, specified by the samplePct parameter. The function returns an integer value that may be used with the function GetModelFromJobId to retrieve the model after fitting is complete.

Usage

```
RequestSampleSizeUpdate(model, samplePct)
```

Arguments

model	An S3 object of class dataRobotModel like that returned by the function GetModelObject, or each element of the list returned by the function GetAllModels.
samplePct	Numeric, specifying the percentage of the training dataset to be used in building the new model.

Details

Motivation for this function is the fact that some models - e.g., very complex machine learning models fit to large datasets - may take a long time to complete. Splitting the model creation request from model retrieval in these cases allows the user to perform other interactive R session tasks between the time the model creation/update request is made and the time the final model is available.

Value

Integer, value to be used as the modelJobId parameter in calling the function GetModelFromJobId to retrieve the updated model.

RequestTransferrableModel

Request generation of an transferrable model file for use in an on-premise DataRobot standalone prediction environment. This function can only be used if model export is enabled, and will only be useful if you have an on-premise environment in which to import it. This function does not download the exported file. Use DownloadTransferrableModel for that. Example: jobId<-RequestTransferrrableModel(projectObject, modelId) WaitForJobToComplete(projectObject, jobId) DownloadTransferrableModel(projectObject, modelId, file)

Description

Request generation of an transferrable model file for use in an on-premise DataRobot standalone prediction environment. This function can only be used if model export is enabled, and will only be useful if you have an on-premise environment in which to import it. This function does not download the exported file. Use DownloadTransferrableModel for that. Example: jobId<-RequestTransferrrableModel(projectObject, modelId) WaitForJobToComplete(projectObject, jobId) DownloadTransferrableModel(projectObject, modelId, file)

Usage

```
RequestTransferrableModel(project, modelId)
```

Arguments

project	Either (1) a character string giving the unique alphanumeric identifier for the project, or (2) a list containing the element projectId with this identifier.
modelId	Unique alphanumeric identifier for the model of interest.

Value

jobId

SetTarget *Set the target variable (and by default, start the DataRobot Autopilot)*

Description

This function sets the target variable for the project defined by project, starting the process of building models to predict the response variable target. Both of these parameters - project and target - are required and they are sufficient to start a modeling project with DataRobot default specifications for the other 10 optional parameters.

Usage

```
SetTarget(project, target, metric = NULL, weights = NULL,
  partition = NULL, mode = NULL, seed = NULL, positiveClass = NULL,
  blueprintThreshold = NULL, responseCap = NULL, recommenderUserId = NULL,
  recommenderItemId = NULL, quickrun = NULL, featurelistId = NULL,
  maxWait = 60)
```

Arguments

project	Either (1) a character string giving the unique alphanumeric identifier for the project, or (2) a list containing the element projectId with this identifier.
target	Character string giving the name of the response variable to be predicted by all project models.
metric	Optional character string specifying the model fitting metric to be optimized; a list of valid options for this parameter, which depends on both project and target, may be obtained with the function GetValidMetrics.
weights	Optional character string specifying the name of the column from the modeling dataset to be used as weights in model fitting.
partition	Optional S3 object of class 'partition' whose elements specify a valid partitioning scheme. See help for functions CreateGroupPartition, CreateRandomPartition, CreateStratifiedPartition, and CreateUserPartition.
mode	Optional, specifies the autopilot mode used to start the modeling project; valid options are 'auto' (fully automatic, the current DataRobot default, obtained when mode = NULL), 'semi' (semi is deprecated in 2.3, will be removed in 3.0), 'manual' and 'quick'

seed	Optional integer seed for the random number generator used in creating random partitions for model fitting.
positiveClass	Optional target variable value corresponding to a positive response in binary classification problems.
blueprintThreshold	Optional integer specifying the maximum time (in hours) that any modeling blueprint is allowed to run before being terminated.
responseCap	Optional floating point value, between 0.5 and 1.0, specifying a capping limit for the response variable. The default value NULL corresponds to an uncapped response, equivalent to responseCap = 1.0.
recommenderUserId	Optional character string, giving the name of the data column containing user ID's (for recommender models only).
recommenderItemId	Optional character string, giving the name of the data column containing item ID's (for recommender models only).
quickrun	Optional logcial variable; if TRUE then DR will perform a quickrun, limiting the number of models evaluated during autopilot. (quickrun flag is deprecated in 2.4, will be removed in 3.0)
featurelistId	Specifies which feature list to use. If NULL (default), a default featurelist is used.
maxWait	Specifies how many seconds to wait for the server to finish analyzing the target and begin the modeling process. If the process takes longer than this parameter specifies, execution will stop (but the server will continue to process the request).

`SetupProject` *Function to set up a new DataRobot project*

Description

This function uploads a modeling dataset defined by the dataSource parameter and allows specification of the optional project name projectName. The dataSource parameter can be either the name of a CSV file or a dataframe; in the latter case, it is saved as a CSV file whose name is described in the Details section. This function returns the projectName specified in the calling sequence, the unique alphanumeric identifier projectId for the new project, the name of the modeling dataset uploaded to create this project, and the project creation time and date.

Usage

```
SetupProject(dataSource, projectName = NULL, maxWait = 60 * 60)
```

Arguments

dataSource	Either (a) the name of a CSV file or (b) a dataframe (c) url to publicly available file; in each case, this parameter identifies the source of the data from which all project models will be built. See Details.
projectName	Optional character string specifying a project name.
maxWait	The maximum time to wait for each of two steps: (1) The initial project creation request, and (2) data processing that occurs after receiving the response to this initial request.

Details

The DataRobot modeling engine requires a CSV file containing the data to be used in fitting models, and this has been implemented here in two ways. The first and simpler is to specify dataSource as the name of this CSV file, but for the convenience of those who wish to work with dataframes, this function also provides the option of specifying a dataframe, which is then written to a CSV file and uploaded to the DataRobot server. In this case, the file name is either specified directly by the user through the saveFile parameter, or indirectly from the name of the dataSource dataframe if saveFile = NULL (the default). In this second case, the file name consists of the name of the dataSource dataframe with the string csvExtension appended.

Value

This function returns a list with the following four components:

projectName The name assigned to the DataRobot project

projectId The unique alphanumeric project identifier for this DataRobot project

fileName The name of the CSV modeling file uploaded for this project

created Character string containing the time and date of project creation

---

SetupProjectFromHDFS *Function to set up a new DataRobot project using datasource on a WebHDFS server*

Description

This function returns the projectName specified in the calling sequence, the unique alphanumeric identifier projectId for the new project, the name of the modeling dataset uploaded to create this project, and the project creation time and date.

Usage

```
SetupProjectFromHDFS(url, port = NULL, projectName = NULL, maxWait = 60 *
  60)
```

Arguments

url	Character string. The location of the WebHDFS file, both server and full path. Per the DataRobot specification, must begin with 'hdfs://'
port	Optional int. The port to use. If not specified, will default to the server default (50070)
projectName	Optional character string specifying a project name.
maxWait	The maximum time to wait for each of two steps: (1) The initial project creation request, and (2) data processing that occurs after receiving the response to this initial request.

Value

This function returns a list with the following four components:

projectName The name assigned to the DataRobot project

projectId The unique alphanumeric project identifier for this DataRobot project

fileName The name of the CSV modeling file uploaded for this project

created Character string containing the time and date of project creation

---

SetupProjectFromMySQL *Function to set up a new DataRobot project using data from MySQL table*

Description

This function returns the projectName specified in the calling sequence, the unique alphanumeric identifier projectId for the new project, the name of the modeling dataset uploaded to create this project, and the project creation time and date.

Usage

```
SetupProjectFromMySQL(server, database, table, user, port = NULL,
  prefetch = NULL, projectName = NULL, password = NULL,
  encryptedPassword = NULL, maxWait = 60 * 60)
```

Arguments

server	Character string. The address of the MySQL server
database	Character string. The name of the database to use
table	Character string. The name of the table to fetch
user	Character string. The username to use to access the database
port	Optional integer. The port to reach the MySQL server. If not specified, will use the default specified by DataRobot (3306).

`prefetch` Optional integer. If specified, specifies the number of rows to stream at a time from the database. If not specified, fetches all results at once. This is an optimization for reading from the database

`projectName` Optional character string specifying a project name.

`password` Optional character string. The plaintext password to be used to access MySQL database. Will be first encrypted with DataRobot. Only use this or 'encryptedPassword', not both.

`encryptedPassword` Optional character string. The encrypted password to be used to access MySQL database. Only use this or 'password', not both.

`maxWait` The maximum time to wait for each of two steps: (1) The initial project creation request, and (2) data processing that occurs after receiving the response to this initial request.

Value

This function returns a list with the following four components:

projectName The name assigned to the DataRobot project

projectId The unique alphanumeric project identifier for this DataRobot project

fileName The name of the CSV modeling file uploaded for this project

created Character string containing the time and date of project creation

---

`SetupProjectFromOracle` *Function to set up a new DataRobot project using data from Oracle table*

Description

This function returns the projectName specified in the calling sequence, the unique alphanumeric identifier projectId for the new project, the name of the modeling dataset uploaded to create this project, and the project creation time and date.

Usage

```
SetupProjectFromOracle(dbq, table, username, fetchBufferSize = NULL,
  projectName = NULL, password = NULL, encryptedPassword = NULL,
  maxWait = 60 * 60)
```

Arguments

dbq	Character string. tnsnames.ora entry in host:port/sid format
table	Character character string. The name of the table to fetch
username	Character character string. The username to use to access the database
fetchBufferSize	Optional integer. If specified, specifies the size of buffer that will be used to stream data from the database. Otherwise will use DataRobot default value.
projectName	Optional character string specifying a project name.
password	Optional character string. The plaintext password to be used to access MySQL database. Will be first encrypted with DataRobot. Only use this or 'encryptedPassword', not both.
encryptedPassword	Optional character string. The encrypted password to be used to access MySQL database. Only use this or 'password', not both.
maxWait	The maximum time to wait for each of two steps: (1) The initial project creation request, and (2) data processing that occurs after receiving the response to this initial request.

Value

This function returns a list with the following four components:

projectName The name assigned to the DataRobot project

projectId The unique alphanumeric project identifier for this DataRobot project

fileName The name of the CSV modeling file uploaded for this project

created Character string containing the time and date of project creation

---

SetupProjectFromPostgreSQL

Function to set up a new DataRobot project using data from PostgreSQL table

---

Description

This function returns the projectName specified in the calling sequence, the unique alphanumeric identifier projectId for the new project, the name of the modeling dataset uploaded to create this project, and the project creation time and date.

Usage

```
SetupProjectFromPostgreSQL(server, database, table, username, port = NULL,
  driver = NULL, fetch = NULL, useDeclareFetch = NULL,
  projectName = NULL, password = NULL, encryptedPassword = NULL,
  maxWait = 60 * 60)
```

Arguments

server	Character string. The address of the MySQL server
database	Character string. The name of the database to use
table	Character string. The name of the table to fetch
username	Character string. The username to use to access the database
port	Optional integer. The port to reach the PostgreSQL server. If not specified, will use the default specified by DataRobot (5432).
driver	Optional character string. Specify ODBC driver to use. If not specified - use DataRobot default. See the values within datarobot.enums.POSTGRESQL_DRIVER
fetch	Optional integer. If specified, specifies the number of rows to stream at a time from the database. If not specified, fetches all results at once. This is an optimization for reading from the database
useDeclareFetch	Optional bool. On True, server will fetch result as available using DB cursor. On False it will try to retrieve entire result set - not recommended for big tables. If not specified - use the default specified by DataRobot.
projectName	Optional character string specifying a project name.
password	Optional character string. The plaintext password to be used to access MySQL database. Will be first encrypted with DataRobot. Only use this or 'encryptedPassword', not both.
encryptedPassword	Optional character string. The encrypted password to be used to access MySQL database. Only use this or 'password', not both.
maxWait	The maximum time to wait for each of two steps: (1) The initial project creation request, and (2) data processing that occurs after receiving the response to this initial request.

Value

This function returns a list with the following four components:

projectName The name assigned to the DataRobot project

projectId The unique alphanumeric project identifier for this DataRobot project

fileName The name of the CSV modeling file uploaded for this project

created Character string containing the time and date of project creation

`StartNewAutoPilot` *Starts autopilot on provided featurelist. Only one autopilot can be running at the time. That's why any ongoing autopilot on different featurelist will be halted - modelling jobs in queue would not be affected but new jobs would not be added to queue by halted autopilot.*

Description

There is an error if autopilot is currently running on or has already finished running on the provided featurelist and also if project's target was not selected (via SetTarget).

Usage

```
StartNewAutoPilot(project, featurelistId, mode = AutopilotMode$FullAuto)
```

Arguments

`project`	Either (1) a character string giving the unique alphanumeric identifier for the project, or (2) a list containing the element projectId with this identifier.
`featurelistId`	Specifies which feature list to use. If NULL (default), a default featurelist is used.
`mode`	The desired autopilot mode: either AutopilotMode$FullAuto (default) or AutopilotMode$SemiAuto (SemiAuto is deprecated in 2.3, will be removed in 3.0)

`summary.dataRobotModel` *DataRobot S3 object methods for R's generic summary function*

Description

These functions extend R's generic summary function to the DataRobot S3 object classes dataRobotModel, dataRobotProject, listOfBlueprints, listOfFeaturelists, listOfModels, and projectSummaryList.

Usage

```
## S3 method for class 'dataRobotModel'
summary(object, ...)

## S3 method for class 'dataRobotProject'
summary(object, ...)

## S3 method for class 'listOfBlueprints'
summary(object, nList = 6, ...)
```

```
## S3 method for class 'listOfFeaturelists'
summary(object, nList = 6, ...)

## S3 method for class 'listOfModels'
summary(object, nList = 6, ...)

## S3 method for class 'projectSummaryList'
summary(object, nList = 6, ...)
```

Arguments

object	The S3 object to be summarized.
nList	Integer: for the 'listOf' class objects, the first nList elements of the list are summarized in the dataframe in the second element of the list returned by the function.
...	Not currently used.

Value

An object-specific summary: for objects of class dataRobotModel and dataRobotProject, this summary is a character vector giving key characteristics of the model or project, respectively; for the other object classes, the value is a two-element list where the first element is a brief summary character string and the second element is a more detailed dataframe with nList elements.

UnpauseQueue *Re-start the DataRobot modeling queue*

Description

This function re-starts the modeling queue for a specified DataRobot project.

Usage

```
UnpauseQueue(project)
```

Arguments

project	Either (1) a character string giving the unique alphanumeric identifier for the project, or (2) a list containing the element projectId with this identifier.

UpdateProject *Update parameters for an existing project*

Description

This function updates parameters for the project defined by project.

Usage

```
UpdateProject(project, newProjectName = NULL, workerCount = NULL,
  holdoutUnlocked = NULL)
```

Arguments

project	Either (1) a character string giving the unique alphanumeric identifier for the project, or (2) a list containing the element projectId with this identifier.
newProjectName	Updated value for the projectName parameter associated with the project.
workerCount	Integer; sets the number of workers requested for the associated project.
holdoutUnlocked	Either NULL (the default) or logical TRUE; if TRUE, this function requests the DataRobot Autopilot to unlock the holdout data subset.

UpdateTransferrableModel *Update the display name or note for an imported model.*

Description

Update the display name or note for an imported model.

Usage

```
UpdateTransferrableModel(importId, displayName = NULL, note = NULL)
```

Arguments

importId	Character string unique id of the import
displayName	Character string The new display name
note	Character string The new note

Value

A list describing upladed transferrable model with the following components:

note Character string Manually added node about this imported model

datasetName Character string Filename of the dataset used to create the project the model belonged to

modelName Character string Model type describing the model generated by DataRobot

displayName Character string Manually specified human-readable name of the imported model

target Character string The target of the project the model belonged to prior to export

projectName Character string Name of the project the model belonged to prior to export

importedByUsername Character string Username of the user who imported the model

importedAt Character string The time the model was imported

version Numeric Project version of the project the model belonged to

projectId Character id of the project the model belonged to prior to export

featurelistName Character string Name of the featurelist used to train the model

createdByUsername Character string Username of the user who created the model prior to export

importedById Character string id of the user who imported the model

id Character string id of the import

createdById Character string id of the user who created the model prior to export

modelId Character string original id of the model prior to export

originUrl Character string URL

UploadPredictionDataset

Function to upload new data to a DataRobot project for predictions

---

Description

The DataRobot prediction engine requires a CSV file containing the data to be used in prediction, and this has been implemented here in two ways. The first and simpler is to specify dataSource as the name of this CSV file, but for the convenience of those who wish to work with dataframes, this function also provides the option of specifying a dataframe, which is then written to a CSV file and uploaded to the DataRobot server.

Usage

```
UploadPredictionDataset(project, dataSource, maxWait = 60)
```

Arguments

`project`	Either (1) a character string giving the unique alphanumeric identifier for the project, or (2) a list containing the element projectId with this identifier.
`dataSource`	Either (a) the name of a CSV file (b) a dataframe or (c) url to publicly available file; in each case, this parameter identifies the source of the data for which predictions will be calculated.
`maxWait`	The maximum time (in seconds) to wait for each of two steps: (1) The initial dataset upload request, and (2) data processing that occurs after receiving the response to this initial request.

Value

list with the following 6 components:

id Character: string giving the unique alphanumeric identifier for the dataset

numColumns Numeric: number of columns in dataset

name Character: Name of dataset file

created Character: Time of upload

projectId Character: string giving the unique alphanumeric identifier for the project

numRows Numeric: number of rows in dataset

`UploadTransferrableModel` *Import a previously exported model for predictions.*

Description

Import a previously exported model for predictions.

Usage

```
UploadTransferrableModel(modelFile, maxWait = 60)
```

Arguments

`modelFile`	Path to binary transeffable model file
`maxWait`	Specifies how many seconds to wait for uplaod to finish

Value

A list describing upladed transferrable model with the following components:

note Character string Manually added node about this imported model

datasetName Character string Filename of the dataset used to create the project the model belonged to

modelName Character string Model type describing the model generated by DataRobot

displayName Character string Manually specified human-readable name of the imported model

target Character string The target of the project the model belonged to prior to export

projectName Character string Name of the project the model belonged to prior to export

importedByUsername Character string Username of the user who imported the model

importedAt Character string The time the model was imported

version Numeric Project version of the project the model belonged to

projectId Character id of the project the model belonged to prior to export

featurelistName Character string Name of the featurelist used to train the model

createdByUsername Character string Username of the user who created the model prior to export

importedById Character string id of the user who imported the model

id Character string id of the import

createdById Character string id of the user who created the model prior to export

modelId Character string original id of the model prior to export

originUrl Character string URL

`ViewWebModel` *Retrieve a DataRobot web page that displays detailed model information*

Description

This function brings up a web page that displays detailed model information like that available from the standard DataRobot user interface (e.g., graphical representations of model structures).

Usage

```
ViewWebModel(model)
```

Arguments

`model`	An S3 object of class dataRobotModel like that returned by the function GetModelObject, or each element of the list returned by the function GetAllModels.

`ViewWebProject` *Retrieve a DataRobot web page that displays detailed project information*

Description

This function brings up a web page that displays detailed project information like that available from the standard DataRobot user interface.

Usage

```
ViewWebProject(project)
```

Arguments

`project`	Either (1) a character string giving the unique alphanumeric identifier for the project, or (2) a list containing the element projectId with this identifier.

`WaitForAutopilot` *This function periodically checks whether Autopilot is finished and returns only after it is.*

Description

This function periodically checks whether Autopilot is finished and returns only after it is.

Usage

```
WaitForAutopilot(project, checkInterval = 20, timeout = NULL,
  verbosity = 1)
```

Arguments

`project`	The project for which you want to wait until autopilot is finished
`checkInterval`	Maximum wait (in seconds) between checks that Autopilot is finished
`timeout`	Time (in seconds) after which to give up (NULL implies no timeout). There is an error if Autopilot is not finished before timing out.
`verbosity`	Verbosity level: 0 is silent, 1 or more displays information about progress

`WaitForJobToComplete` *Wait for specified job to complete*

Description

Wait for specified job to complete

Usage

```
WaitForJobToComplete(project, jobId, maxWait = 60)
```

Arguments

`project`	Either (1) a character string giving the unique alphanumeric identifier for the project, or (2) a list containing the element projectId with this identifier.
`jobId`	integer identifier (returned for example by RequestPrimeModel)
`maxWait`	maximum time to wait (in seconds) for the job to complete

Index