Package 'PatientLevelPrediction'

April 3, 2017

Type Package

Title Package for patient level prediction using data in the OMOP Common Data Model

Version 1.2.0

Date 2016-11-17

Author Jenna Reps [aut],
Martijn J. Schuemie [aut, cre],
Marc A. Suchard [aut],
Patrick B. Ryan [aut],
Peter R. Rijnbeek [aut]

Maintainer Jenna Reps <reps@ohdsi.org>

Description A package for creating patient level prediction models. Given a cohort of interest and an outcome of interest, the package can use data in the OMOP Common Data Model to build a large set of features. These features can then be assessed to fit a predictive model using a number of machine learning algorithms. Several performance measures are implemented for model evaluation.

License Apache License 2.0

Depends R (>= 3.2.2),
DatabaseConnector (>= 1.3.0),
Cyclops (>= 1.2.1-2)

Imports ggplot2,
gridExtra,
bit,
ff,
ffbase (>= 0.12.1),
plyr,
survAUC,
Rcpp (>= 0.11.2),
RJDBC,
SqlRender (>= 1.1.3),
survival,
FeatureExtraction,
xgboost,
Matrix,
AUC,
PythonInR,
futile.options,

futile.logger,
utils,
methods,
BigKnn,
reshape2

Suggests testthat,
pROC,
gnm,
knitr,
rmarkdown,
scoring,
Metrics,
SparseM,
ResourceSelection

LinkingTo Rcpp

NeedsCompilation yes

RoxygenNote 6.0.1

R topics documented:

`accuracy` *Calculate the accuracy*

Description

Calculate the accuracy

Usage

```
accuracy(TP, TN, FN, FP)
```

Arguments

TP	Number of true positives
TN	Number of true negatives
FN	Number of false negatives
FP	Number of false positives

Details

Calculate the accuracy

Value

accuracy value

`applyModel` *Apply train model on new data Apply a Patient Level Prediction model on Patient Level Prediction Data and get the predicted risk in [0,1] for each person in the population. If the user inputs a population with an outcomeCount column then the function also returns the evaluation of the prediction (AUC, brier score, calibration)*

Description

Apply train model on new data Apply a Patient Level Prediction model on Patient Level Prediction Data and get the predicted risk in [0,1] for each person in the population. If the user inputs a population with an outcomeCount column then the function also returns the evaluation of the prediction (AUC, brier score, calibration)

Usage

```
applyModel(population, plpData, plpModel, logConnection = NULL,
  databaseOutput = NULL, silent = F)
```

Arguments

population	The population of people who you want to predict the risk for
plpData	The plpData for the population
plpModel	The trained PatientLevelPrediction model
logConnection	A connection to output any logging during the process
databaseOutput	Whether to save the details into the prediction database
silent	Whether to turn off progress reporting

Examples

```
## Not run:
# load the model and data
plpData <- loadPlpData("C:/plpdata")
plpModel <- loadPlpModel("C:/plpmodel")

# use the same population settings as the model:
populationSettings <- plpModel$populationSettings
populationSettings$plpData <- plpData
population <- do.call(createStudyPopulation, populationSettings)

# get the prediction:
prediction <- applyModel(population, plpData, plpModel)$prediction

## End(Not run)
```

---

averagePrecision *Calculate the average precision*

Description

Calculate the average precision

Usage

```
averagePrecision(prediction)
```

Arguments

prediction	A prediction object as generated using the predictProbabilities function.

Details

Calculates the average precision from a predition object

Value

The average precision

brierScore *brierScore*

Description

brierScore

Usage

```
brierScore(prediction)
```

Arguments

prediction	A prediction object as generated using the predictProbabilities function.

Details

Calculates the brierScore from prediction object

Value

A list containing the brier score and the scaled brier score

bySumFf *Compute sum of values binned by a second variable*

Description

Compute sum of values binned by a second variable

Usage

```
bySumFf(values, bins)
```

Arguments

values	An ff object containing the numeric values to be summed
bins	An ff object containing the numeric values to bin by

Examples

```
values <- ff::as.ff(c(1, 1, 2, 2, 1))
bins <- ff::as.ff(c(1, 1, 1, 2, 2))
bySumFf(values, bins)
```

`calibrationLine` *calibrationLine*

Description

calibrationLine

Usage

```
calibrationLine(prediction, numberOfStrata = 10)
```

Arguments

`prediction`	A prediction object as generated using the `predictProbabilities` function.
`numberOfStrata`	The number of groups to split the prediction into

Details

Calculates the calibration from prediction object

`computeAuc` *Compute the area under the ROC curve*

Description

Compute the area under the ROC curve

Usage

```
computeAuc(prediction, confidenceInterval = FALSE)
```

Arguments

`prediction`	A prediction object as generated using the `predict` functions.
`confidenceInterval`	Should 95 percebt confidence intervals be computed?

Details

Computes the area under the ROC curve for the predicted probabilities, given the true observed outcomes.

computeAucFromDataFrames *Compute the area under the ROC curve*

Description

Compute the area under the ROC curve

Usage

```
computeAucFromDataFrames(prediction, status, time = NULL,
  confidenceInterval = FALSE, timePoint, modelType = "logistic")
```

Arguments

prediction	A vector with the predicted hazard rate.
status	A vector with the status of 1 (event) or 0 (no event).
time	Only for survival models: a vector with the time to event or censor (which ever comes first).
confidenceInterval	Should 95 percebt confidence intervals be computed?
timePoint	Only for survival models: time point when the AUC should be evaluated
modelType	Type of model. Currently supported are "logistic" and "survival".

Details

Computes the area under the ROC curve for the predicted probabilities, given the true observed outcomes.

createStudyPopulation *Create a study population*

Description

Create a study population

Usage

```
createStudyPopulation(plpData, population = NULL, outcomeId, binary = T,
  includeAllOutcomes = T, firstExposureOnly = FALSE, washoutPeriod = 0,
  removeSubjectsWithPriorOutcome = TRUE, priorOutcomeLookback = 99999,
  requireTimeAtRisk = T, minTimeAtRisk = 365, riskWindowStart = 0,
  addExposureDaysToStart = FALSE, riskWindowEnd = 365,
  addExposureDaysToEnd = F, verbosity = futile.logger::INFO, ...)
```

Arguments

plpData	An object of type plpData as generated using getDbplpData.
population	If specified, this population will be used as the starting point instead of the cohorts in the plpData object.
outcomeId	The ID of the outcome. If not specified, no outcome-specific transformations will be performed.
binary	Forces the outcomeCount to be 0 or 1 (use for binary prediction problems)
includeAllOutcomes	(binary) indicating whether to include people with outcomes who are not observed for the whole at risk period
firstExposureOnly	Should only the first exposure per subject be included? Note that this is typically done in the createStudyPopulation function,
washoutPeriod	The mininum required continuous observation time prior to index date for a person to be included in the cohort.
removeSubjectsWithPriorOutcome	Remove subjects that have the outcome prior to the risk window start?
priorOutcomeLookback	How many days should we look back when identifying prior outcomes?
requireTimeAtRisk	Should subject without time at risk be removed?
minTimeAtRisk	The minimum number of days at risk required to be included
riskWindowStart	The start of the risk window (in days) relative to the index date (+ days of exposure if the addExposureDaysToStart parameter is specified).
addExposureDaysToStart	Add the length of exposure the start of the risk window?
riskWindowEnd	The end of the risk window (in days) relative to the index data (+ days of exposure if the addExposureDaysToEnd parameter is specified).
addExposureDaysToEnd	Add the length of exposure the risk window?
verbosity	Sets the level of the verbosity. If the log level is at or higher in priority than the logger threshold, a message will print. The levels are: • DEBUGHighest verbosity showing all debug statements • TRACEShowing information about start and end of steps • INFOShow informative information (Default) • WARNShow warning messages • ERRORShow error messages • FATALBe silent except for fatal errors
...	Other inputs

Details

Create a study population by enforcing certain inclusion and exclusion criteria, defining a risk window, and determining which outcomes fall inside the risk window.

Value

A data frame specifying the study population. This data frame will have the following columns:

rowId A unique identifier for an exposure

subjectId The person ID of the subject

cohortStartdate The index date

outcomeCount The number of outcomes observed during the risk window

timeAtRisk The number of days in the risk window

survivalTime The number of days until either the outcome or the end of the risk window

---

`diagnosticOddsRatio` *Calculate the diagnostic odds ratio*

Description

Calculate the diagnostic odds ratio

Usage

```
diagnosticOddsRatio(TP, TN, FN, FP)
```

Arguments

`TP`	Number of true positives
`TN`	Number of true negatives
`FN`	Number of false negatives
`FP`	Number of false positives

Details

Calculate the diagnostic odds ratio

Value

diagnosticOddsRatio value

`evaluatePlp` *evaluatePlp*

Description

Evaluates the performance of the patient level prediction model

Usage

```
evaluatePlp(prediction, plpData)
```

Arguments

`prediction`	The patient level prediction model's prediction
`plpData`	The patient level prediction data

Details

The function calculates various metrics to measure the performance of the model

Value

A list containing the performance values

`exportPlpDataToCsv` *Export all data in a plpData object to CSV files*

Description

Export all data in a plpData object to CSV files

Usage

```
exportPlpDataToCsv(plpData, outputFolder)
```

Arguments

`plpData`	An object of type `plpData`.
`outputFolder`	The folder on the file system where the CSV files will be created. If the folder does not yet exist it will be created.

Details

Created a set of CSV files in the output folder with all the data in the plplData object. This function is intended to be used for research into prediction methods. The following files will be created:

cohort.csv Listing all persons and their prediction periods. This file will have these fields: row_id (a unique ID per period), person_id, cohort_start_date, cohort_id, time (number of days in the window).

outcomes.csv Listing all outcomes per period. This file will have these fields: row_id, outcome_id, outcome_count, time_to_event.

exclude.csv Either not exported or a file listing per outcome ID which windows had the outcome prior to the window and should therefore be removed prior to fitting the model. This object will have these fields: rowId, outcomeId.

covariates.csv Listing the baseline covariates per person in the cohorts. This is done using a sparse representation: covariates with a value of 0 are omitted to save space. The covariates file will have three columns: rowId, covariateId, and covariateValue.

covariateRef.csv A file describing the covariates that have been extracted.

metaData Some information on how the plpData object was constructed.

Examples

```
## Not run:
exportPlpDataToCsv(plpData, "s:/temp/exportTest")

## End(Not run)
```

`f1Score` *Calculate the f1Score*

Description

Calculate the f1Score

Usage

```
f1Score(TP, TN, FN, FP)
```

Arguments

`TP`	Number of true positives
`TN`	Number of true negatives
`FN`	Number of false negatives
`FP`	Number of false positives

Details

Calculate the f1Score

Value

f1Score value

falseDiscoveryRate *Calculate the falseDiscoveryRate*

Description

Calculate the falseDiscoveryRate

Usage

```
falseDiscoveryRate(TP, TN, FN, FP)
```

Arguments

TP	Number of true positives
TN	Number of true negatives
FN	Number of false negatives
FP	Number of false positives

Details

Calculate the falseDiscoveryRate

Value

falseDiscoveryRate value

falseNegativeRate *Calculate the falseNegativeRate*

Description

Calculate the falseNegativeRate

Usage

```
falseNegativeRate(TP, TN, FN, FP)
```

Arguments

TP	Number of true positives
TN	Number of true negatives
FN	Number of false negatives
FP	Number of false positives

Details

Calculate the falseNegativeRate

Value

falseNegativeRate value

falseOmissionRate *Calculate the falseOmissionRate*

Description

Calculate the falseOmissionRate

Usage

```
falseOmissionRate(TP, TN, FN, FP)
```

Arguments

TP	Number of true positives
TN	Number of true negatives
FN	Number of false negatives
FP	Number of false positives

Details

Calculate the falseOmissionRate

Value

falseOmissionRate value

falsePositiveRate *Calculate the falsePositiveRate*

Description

Calculate the falsePositiveRate

Usage

```
falsePositiveRate(TP, TN, FN, FP)
```

Arguments

TP	Number of true positives
TN	Number of true negatives
FN	Number of false negatives
FP	Number of false positives

Details

Calculate the falsePositiveRate

Value

falsePositiveRate value

`fitGLMModel` *Fit a predictive model*

Description

Fit a predictive model

Usage

```
fitGLMModel(population, plpData, modelType = "logistic",
  excludeCovariateIds = c(), includeCovariateIds = c(),
  prior = createPrior("laplace", useCrossValidation = TRUE),
  control = createControl(cvType = "auto", fold = 3, startingVariance = 0.01,
  tolerance = 2e-06, cvRepetitions = 1, selectorType = "byPid", noiseLevel =
  "silent", threads = -1, maxIterations = 3000))
```

Arguments

`population`	A population object generated by `createStudyPopulation`, potentially filtered by other functions.
`plpData`	An object of type `plpData` as generated using `getDbPlpData`.
`modelType`	The type of outcome model that will be used. Possible values are "logistic", "poisson", or "cox".
`excludeCovariateIds`	Exclude these covariates from the outcome model.
`includeCovariateIds`	Include only these covariates in the outcome model.
`prior`	The prior used to fit the model. See `createPrior` for details.
`control`	The control object used to control the cross-validation used to determine the hyperparameters of the prior (if applicable). See `createControl` for details.

`fitPlp` *fitPlp*

Description

Train various models using a default parameter gird search or user specified parameters

Usage

```
fitPlp(population, data, modelSettings, cohortId, outcomeId)
```

Arguments

population	The population created using createStudyPopulation() who will have their risks predicted
data	An object of type plpData - the patient level prediction data extracted from the CDM.
modelSettings	An object of class modelSettings created using one of the function: • logisticRegressionModel() A lasso logistic regression model • GBMclassifier() A gradient boosting machine • RFclassifier() A random forest model • GLMclassifier () A generalised linear model • KNNclassifier() A KNN model
cohortId	Id of study cohort
outcomeId	Id of outcome cohort

Details

The user can define the machine learning model to train (regularised logistic regression, random forest, gradient boosting machine, neural network and)

Value

An object of class plpModel containing:

model	The trained prediction model
modelLoc	The path to where the model is saved (if saved)
trainAuc	The AUC obtained on the training set
trainCalibration	The calibration obtained on the training set
modelSettings	A list specifiying the model, preprocessing, outcomeId and cohortId
metaData	The model meta data
trainingTime	The time taken to train the classifier

---

getAttritionTable *Get the attrition table for a population*

Description

Get the attrition table for a population

Usage

```
getAttritionTable(object)
```

Arguments

object	Either an object of type plpData, a population object generated by functions like createStudyPopulation, or an object of type outcomeModel.

Value

A data frame specifying the number of people and exposures in the population after specific steps of filtering.

`getCalibration` *Get a sparse summary of the calibration*

Description

Get a sparse summary of the calibration

Usage

```
getCalibration(prediction, numberOfStrata = 10, truncateFraction = 0.01)
```

Arguments

`prediction`	A prediction object as generated using the `predict` functions.
`numberOfStrata`	The number of strata in the plot.
`truncateFraction`	This fraction of probability values will be ignored when plotting, to avoid the x-axis scale being dominated by a few outliers.

Details

Generates a sparse summary showing the predicted probabilities and the observed fractions. Predictions are stratefied into equally sized bins of predicted probabilities.

Value

A dataframe with the calibration summary

`getModelDetails` *Get the predictive model details*

Description

`getModelDetails` shows the full model, so showing the betas of all variables included in the model, along with the variable names

Usage

```
getModelDetails(predictiveModel, plpData)
```

Arguments

`predictiveModel`	An object of type `predictiveModel` as generated using he `fitPlp` function.
`plpData`	An object of type `plpData` as generated using `getPlpData`.

Details

Shows the coefficients and names of the covariates with non-zero coefficients.

`getPlpData` *Get the patient level prediction data from the server*

Description

This function executes a large set of SQL statements against the database in OMOP CDM format to extract the data needed to perform the analysis.

Usage

```
getPlpData(connectionDetails, cdmDatabaseSchema,
  oracleTempSchema = cdmDatabaseSchema, cohortId, outcomeIds,
  studyStartDate = "", studyEndDate = "",
  cohortDatabaseSchema = cdmDatabaseSchema, cohortTable = "cohort",
  outcomeDatabaseSchema = cdmDatabaseSchema, outcomeTable = "cohort",
  cdmVersion = "5", excludeDrugsFromCovariates = F,
  firstExposureOnly = FALSE, washoutPeriod = 0, covariateSettings)
```

Arguments

`connectionDetails`
: An R object of type `connectionDetails` created using the function `createConnectionDetails` in the `DatabaseConnector` package.

`cdmDatabaseSchema`
: The name of the database schema that contains the OMOP CDM instance. Requires read permissions to this database. On SQL Server, this should specifiy both the database and the schema, so for example 'cdm_instance.dbo'.

`oracleTempSchema`
: For Oracle only: the name of the database schema where you want all temporary tables to be managed. Requires create/insert permissions to this database.

`cohortId`
: A unique identifier to define the at risk cohort. If cohortTable = DRUG_ERA, cohortId is a CONCEPT_ID and all descendant concepts within that CONCEPT_ID will be used to define the cohort. If cohortTable <> DRUG_ERA, cohortId is used to select the cohort_concept_id in the cohort-like table.

`outcomeIds`
: A list of cohort_definition_ids used to define outcomes.

`studyStartDate`
: A calendar date specifying the minimum date that a cohort index date can appear. Date format is 'yyyymmdd'.

`studyEndDate`
: A calendar date specifying the maximum date that a cohort index date can appear. Date format is 'yyyymmdd'. Important: the study end data is also used to truncate risk windows, meaning no outcomes beyond the study end date will be considered.

`cohortDatabaseSchema`
: The name of the database schema that is the location where the cohort data used to define the at risk cohort is available. If cohortTable = DRUG_ERA, cohortDatabaseSchema is not used by assumed to be cdmSchema. Requires read permissions to this database.

`cohortTable` The tablename that contains the at risk cohort. If cohortTable <> DRUG_ERA, then expectation is cohortTable has format of COHORT table: cohort_concept_id, SUBJECT_ID, COHORT_START_DATE, COHORT_END_DATE.

`outcomeDatabaseSchema`
The name of the database schema that is the location where the data used to define the outcome cohorts is available. If cohortTable = CONDITION_ERA, exposureDatabaseSchema is not used by assumed to be cdmSchema. Requires read permissions to this database.

`outcomeTable` The tablename that contains the outcome cohorts. If outcomeTable <> CONDITION_OCCURRENCE, then expectation is outcomeTable has format of COHORT table: COHORT_DEFINITION_ID, SUBJECT_ID, COHORT_START_DATE, COHORT_END_DATE.

`cdmVersion` Define the OMOP CDM version used: currently support "4" and "5".

`excludeDrugsFromCovariates`
Should the target and comparator drugs (and their descendant concepts) be excluded from the covariates? Note that this will work if the drugs are actualy drug concept IDs (and not cohort IDs).

`firstExposureOnly`
Should only the first exposure per subject be included? Note that this is typically done in the `createStudyPopulation` function, but can already be done here for efficiency reasons.

`washoutPeriod` The mininum required continuous observation time prior to index date for a person to be included in the at risk cohort. Note that this is typically done in the `createStudyPopulation` function, but can already be done here for efficiency reasons.

`covariateSettings`
An object of type `covariateSettings` as created using the `createCovariateSettings` function in the `FeatureExtraction` package.

Details

Based on the arguments, the at risk cohort data is retrieved, as well as outcomes occurring in these subjects. The at risk cohort can be identified using the drug_era table, or through user-defined cohorts in a cohort table either inside the CDM instance or in a separate schema. Similarly, outcomes are identified using the condition_era table or through user-defined cohorts in a cohort table either inside the CDM instance or in a separate schema. Covariates are automatically extracted from the appropriate tables within the CDM. Important: The concepts used to define the at risk cohort must not be included in the covariates, including any descendant concepts. If the `cohortId` arguments represent real concept IDs, you can set the `excludeDrugsFromCovariates` argument to TRUE and automatically the drugs and their descendants will be excluded from the covariates. However, if the `cohortId` argument does not represent concept IDs, you will need to manually add the concept_ids and descendants to the `excludedCovariateConceptIds` of the `covariateSettings` argument.

Value

Returns an object of type `plpData`, containing information on the cohorts, their outcomes, and baseline covariates. Information about multiple outcomes can be captured at once for efficiency reasons. This object is a list with the following components:

outcomes A data frame listing the outcomes per person, including the time to event, and the outcome id. Outcomes are not yet filtered based on risk window, since this is done at a later stage.

cohorts A data frame listing the persons in each cohort, listing their exposure status as well as the time to the end of the observation period and time to the end of the cohort (usually the end of the exposure era).

covariates An ffdf object listing the baseline covariates per person in the two cohorts. This is done using a sparse representation: covariates with a value of 0 are omitted to save space.

covariateRef An ffdf object describing the covariates that have been extracted.

metaData A list of objects with information on how the cohortMethodData object was constructed.

The generic `()` and `summary()` functions have been implemented for this object.

`getPredictionDistribution` *Calculates the prediction distribution*

Description

Calculates the prediction distribution

Usage

```
getPredictionDistribution(prediction)
```

Arguments

`prediction`	A prediction object as generated using the `predictProbabilities` function.

Details

Calculates the quantiles from a predition object

Value

The 0.00, 0.1, 0.25, 0.5, 0.75, 0.9, 1.00 quantile pf the prediction, the mean and standard deviation per class

---

`getThresholdSummary` *Calculate all measures for sparse ROC*

Description

Calculate all measures for sparse ROC

Usage

```
getThresholdSummary(prediction)
```

Arguments

`prediction`	A prediction object as generated using the `predictProbabilities` function.

Details

Calculates the TP, FP, TN, FN, TPR, FPR, accuracy, PPF, FOR and Fmeasure from a predition object

Value

A data.frame with all the measures

`grepCovariateNames` *Extract covariate names*

Description

Extracts covariate names using a regular-expression.

Usage

```
grepCovariateNames(pattern, object)
```

Arguments

`pattern`	A regular expression with which to name covariate names
`object`	An R object of type `plpData` or `covariateData`.

Details

This function extracts covariate names that match a regular-expression for a `plpData` or `covariateData` object.

Value

Returns a `data.frame` containing information about covariates that match a regular expression. This `data.frame` has the following columns:

covariateId Numerical identifier for use in model fitting using these covariates

covariateName Text identifier

analysisId Analysis identifier

conceptId OMOP common data model concept identifier, or 0

insertDbPopulation *Insert a population into a database*

Description

Insert a population into a database

Usage

```
insertDbPopulation(population, cohortIds = 1, connectionDetails,
  cohortDatabaseSchema, cohortTable = "cohort", createTable = FALSE,
  dropTableIfExists = TRUE, cdmVersion = "5")
```

Arguments

population	Either an object of type plpData or a population object generated by functions like createStudyPopulation.
cohortIds	The IDs to be used for the treated and comparator cohort, respectively.
connectionDetails	An R object of type connectionDetails created using the function createConnectionDetails in the DatabaseConnector package.
cohortDatabaseSchema	The name of the database schema where the data will be written. Requires write permissions to this database. On SQL Server, this should specifiy both the database and the schema, so for example 'cdm_instance.dbo'.
cohortTable	The name of the table in the database schema where the data will be written.
createTable	Should a new table be created? If not, the data will be inserted into an existing table.
dropTableIfExists	If createTable = TRUE and the table already exists it will be overwritten.
cdmVersion	Define the OMOP CDM version used: currently support "4" and "5".

Details

Inserts a population table into a database. The table in the database will have the same structure as the 'cohort' table in the Common Data Model.

loadPlpData *Load the cohort data from a folder*

Description

loadPlpData loads an object of type plpData from a folder in the file system.

Usage

```
loadPlpData(file, readOnly = TRUE)
```

Arguments

file	The name of the folder containing the data.
readOnly	If true, the data is opened read only.

Details

The data will be written to a set of files in the folder specified by the user.

Value

An object of class plpData.

Examples

```
# todo
```

---

loadPlpModel *loads the plp model*

Description

loads the plp model

Usage

```
loadPlpModel(dirPath)
```

Arguments

dirPath	The location of the model

Details

Loads a plp model that was saved using `savePlpModel()`

---

loadPlpResult *Loads the evalaution dataframe*

Description

Loads the evalaution dataframe

Usage

```
loadPlpResult(dirPath)
```

Arguments

dirPath	The directory where the evaluation was saved

Details

Loads the evaluation

`loadPrediction` *Loads the prediciton dataframe to csv*

Description

Loads the prediciton dataframe to csv

Usage

```
loadPrediction(dirPath)
```

Arguments

`dirPath`	The directory to saved the csv

Details

Loads the prediciton csv file

`negativeLikelihoodRatio` *Calculate the negativeLikelihoodRatio*

Description

Calculate the negativeLikelihoodRatio

Usage

```
negativeLikelihoodRatio(TP, TN, FN, FP)
```

Arguments

`TP`	Number of true positives
`TN`	Number of true negatives
`FN`	Number of false negatives
`FP`	Number of false positives

Details

Calculate the negativeLikelihoodRatio

Value

negativeLikelihoodRatio value

`negativePredictiveValue` *Calculate the negativePredictiveValue*

Description

Calculate the negativePredictiveValue

Usage

```
negativePredictiveValue(TP, TN, FN, FP)
```

Arguments

TP	Number of true positives
TN	Number of true negatives
FN	Number of false negatives
FP	Number of false positives

Details

Calculate the negativePredictiveValue

Value

negativePredictiveValue value

`PatientLevelPrediction` *PatientLevelPrediction*

Description

PatientLevelPrediction

personSplitter *Split data into random subsets stratified by class*

Description

Split data into random subsets stratified by class

Usage

```
personSplitter(population, test = 0.3, nfold = 3, seed = NULL)
```

Arguments

population	An object created using createStudyPopulation().
test	A real number between 0 and 1 indicating the test set fraction of the data
nfold	An integer >= 1 specifying the number of folds used in cross validation
seed	If set a fixed seed is used, otherwise a random split is performed

Details

Returns a dataframe of rowIds and indexes with a -1 index indicating the rowId belongs to the test set and a positive integer index value indicating the rowId's cross valiation fold within the train set.

Value

A dataframe containing the columns: rowId and index

plotDemographicSummary *Plot the Observed vs. expected incidence, by age and gender*

Description

Plot the Observed vs. expected incidence, by age and gender

Usage

```
plotDemographicSummary(evaluation, type = "train", fileName = NULL)
```

Arguments

evaluation	A prediction object as generated using the runPlp function.
type	options: 'train' or test'
fileName	Name of the file where the plot should be saved, for example 'plot.png'. See the function ggsave in the ggplot2 package for supported file formats.

Details

Create a plot showing the Observed vs. expected incidence, by age and gender #'

Value

A ggplot object. Use the ggsave function to save to file in a different format.

`plotF1Measure` *Plot the F1 measure efficiency frontier using the sparse thresholdSummary data frame*

Description

Plot the F1 measure efficiency frontier using the sparse thresholdSummary data frame

Usage

```
plotF1Measure(evaluation, type = "train", fileName = NULL)
```

Arguments

`evaluation`	A prediction object as generated using the runPlp function.
`type`	options: 'train' or test'
`fileName`	Name of the file where the plot should be saved, for example 'plot.png'. See the function ggsave in the ggplot2 package for supported file formats.

Details

Create a plot showing the F1 measure efficiency frontier using the sparse thresholdSummary data frame

Value

A ggplot object. Use the ggsave function to save to file in a different format.

`plotGeneralizability` *Plot the train/test generalizability diagnostic*

Description

Plot the train/test generalizability diagnostic

Usage

```
plotGeneralizability(covariateSummary, fileName = NULL)
```

Arguments

`covariateSummary`	A prediction object as generated using the runPlp function.
`fileName`	Name of the file where the plot should be saved, for example 'plot.png'. See the function ggsave in the ggplot2 package for supported file formats.

Details

Create a plot showing the train/test generalizability diagnostic #'

Value

A ggplot object. Use the ggsave function to save to file in a different format.

---

plotPlp *Plot all the PatientLevelPrediction plots*

Description

Plot all the PatientLevelPrediction plots

Usage

```
plotPlp(result, filename)
```

Arguments

result	Object returned by the runPlp() function
filename	Name of the file where the plot should be saved, for example 'plot.png'. See the function ggsave in the ggplot2 package for supported file formats.

Details

Create a directory with all the plots

Value

TRUE if it ran

---

plotPrecisionRecall *Plot the precision-recall curve using the sparse thresholdSummary data frame*

Description

Plot the precision-recall curve using the sparse thresholdSummary data frame

Usage

```
plotPrecisionRecall(evaluation, type = "train", fileName = NULL)
```

Arguments

evaluation	A prediction object as generated using the runPlp function.
type	options: 'train' or test'
fileName	Name of the file where the plot should be saved, for example 'plot.png'. See the function ggsave in the ggplot2 package for supported file formats.

Details

Create a plot showing the precision-recall curve using the sparse thresholdSummary data frame

Value

A ggplot object. Use the ggsave function to save to file in a different format.

`plotPredictedPDF` *Plot the Predicted probability density function, showing prediction overlap between true and false cases*

Description

Plot the Predicted probability density function, showing prediction overlap between true and false cases

Usage

```
plotPredictedPDF(evaluation, type = "train", fileName = NULL)
```

Arguments

`evaluation`	A prediction object as generated using the runPlp function.
`type`	options: 'train' or test'
`fileName`	Name of the file where the plot should be saved, for example 'plot.png'. See the function ggsave in the ggplot2 package for supported file formats.

Details

Create a plot showing the predicted probability density function, showing prediction overlap between true and false cases

Value

A ggplot object. Use the ggsave function to save to file in a different format.

`plotPredictionDistribution` *Plot the side-by-side boxplots of prediction distribution, by class#'*

Description

Plot the side-by-side boxplots of prediction distribution, by class#'

Usage

```
plotPredictionDistribution(evaluation, type = "train", fileName = NULL)
```

Arguments

evaluation	A prediction object as generated using the runPlp function.
type	options: 'train' or test'
fileName	Name of the file where the plot should be saved, for example 'plot.png'. See the function ggsave in the ggplot2 package for supported file formats.

Details

Create a plot showing the side-by-side boxplots of prediction distribution, by class #'

Value

A ggplot object. Use the ggsave function to save to file in a different format.

---

plotPreferencePDF *Plot the preference score probability density function, showing prediction overlap between true and false cases #'*

Description

Plot the preference score probability density function, showing prediction overlap between true and false cases #'

Usage

```
plotPreferencePDF(evaluation, type = "train", fileName = NULL)
```

Arguments

evaluation	A prediction object as generated using the runPlp function.
type	options: 'train' or test'
fileName	Name of the file where the plot should be saved, for example 'plot.png'. See the function ggsave in the ggplot2 package for supported file formats.

Details

Create a plot showing the preference score probability density function, showing prediction overlap between true and false cases #'

Value

A ggplot object. Use the ggsave function to save to file in a different format.

plotRoc *Plot the ROC curve*

Description

Plot the ROC curve

Usage

```
plotRoc(prediction, fileName = NULL)
```

Arguments

prediction	A prediction object as generated using the predictProbabilities function.
fileName	Name of the file where the plot should be saved, for example 'plot.png'. See the function ggsave in the ggplot2 package for supported file formats.

Details

Create a plot showing the Receiver Operator Characteristics (ROC) curve.

Value

A ggplot object. Use the ggsave function to save to file in a different format.

plotSparseCalibration *Plot the calibration*

Description

Plot the calibration

Usage

```
plotSparseCalibration(evaluation, type = "train", fileName = NULL)
```

Arguments

evaluation	A prediction object as generated using the runPlp function.
type	options: 'train' or test'
fileName	Name of the file where the plot should be saved, for example 'plot.png'. See the function ggsave in the ggplot2 package for supported file formats.

Details

Create a plot showing the calibration #'

Value

A ggplot object. Use the ggsave function to save to file in a different format.

plotSparseCalibration2 *Plot the conventional calibration*

Description

Plot the conventional calibration

Usage

```
plotSparseCalibration2(evaluation, type = "train", fileName = NULL)
```

Arguments

evaluation	A prediction object as generated using the runPlp function.
type	options: 'train' or test'
fileName	Name of the file where the plot should be saved, for example 'plot.png'. See the function ggsave in the ggplot2 package for supported file formats.

Details

Create a plot showing the calibration #'

Value

A ggplot object. Use the ggsave function to save to file in a different format.

plotSparseRoc *Plot the ROC curve using the sparse thresholdSummary data frame*

Description

Plot the ROC curve using the sparse thresholdSummary data frame

Usage

```
plotSparseRoc(evaluation, type = "train", fileName = NULL)
```

Arguments

evaluation	A prediction object as generated using the runPlp function.
type	options: 'train' or test'
fileName	Name of the file where the plot should be saved, for example 'plot.png'. See the function ggsave in the ggplot2 package for supported file formats.

Details

Create a plot showing the Receiver Operator Characteristics (ROC) curve.

Value

A ggplot object. Use the ggsave function to save to file in a different format.

plotVariableScatterplot *Plot the variable importance scatterplot*

Description

Plot the variable importance scatterplot

Usage

```
plotVariableScatterplot(covariateSummary, fileName = NULL)
```

Arguments

covariateSummary	A prediction object as generated using the runPlp function.
fileName	Name of the file where the plot should be saved, for example 'plot.png'. See the function ggsave in the ggplot2 package for supported file formats.

Details

Create a plot showing the variable importance scatterplot #'

Value

A ggplot object. Use the ggsave function to save to file in a different format.

plpDataSimulationProfile *A simulation profile*

Description

A simulation profile

Usage

```
data(plpDataSimulationProfile)
```

`positiveLikelihoodRatio` *Calculate the positiveLikelihoodRatio*

Description

Calculate the positiveLikelihoodRatio

Usage

```
positiveLikelihoodRatio(TP, TN, FN, FP)
```

Arguments

TP	Number of true positives
TN	Number of true negatives
FN	Number of false negatives
FP	Number of false positives

Details

Calculate the positiveLikelihoodRatio

Value

positiveLikelihoodRatio value

`positivePredictiveValue` *Calculate the positivePredictiveValue*

Description

Calculate the positivePredictiveValue

Usage

```
positivePredictiveValue(TP, TN, FN, FP)
```

Arguments

TP	Number of true positives
TN	Number of true negatives
FN	Number of false negatives
FP	Number of false positives

Details

Calculate the positivePredictiveValue

Value

positivePredictiveValue value

`predictFfdf` *Generated predictions from a regression model*

Description

Generated predictions from a regression model

Usage

```
predictFfdf(coefficients, population, covariates, modelType = "logistic")
```

Arguments

`coefficients`	A names numeric vector where the names are the covariateIds, except for the first value which is expected to be the intercept.
`population`	A data frame containing the population to do the prediction for
`covariates`	A data frame or ffdf object containing the covariates with predefined columns (see below).
`modelType`	Current supported types are "logistic", "poisson", or "survival".

Details

These columns are expected in the outcome object:

rowId	(integer)	Row ID is used to link multiple covariates (x) to a single outcome (y)
time	(real)	For models that use time (e.g. Poisson or Cox regression) this contains time (e.g. number of days)

These columns are expected in the covariates object:

rowId	(integer)	Row ID is used to link multiple covariates (x) to a single outcome (y)
covariateId	(integer)	A numeric identifier of a covariate
covariateValue	(real)	The value of the specified covariate

`predictPlp` *predictPlp*

Description

Predict the risk of the outcome using the input plpModel for the input plpData

Usage

```
predictPlp(plpModel, population, plpData, index = NULL)
```

Arguments

plpModel	An object of type plpModel - a patient level prediction model
population	The population created using createStudyPopulation() who will have their risks predicted
plpData	An object of type plpData - the patient level prediction data extracted from the CDM.
index	A data frame containing rowId: a vector of rowids and index: a vector of doubles the same length as the rowIds. If used, only the rowIds with a negative index value are used to calculate the prediction.

Details

The function applied the trained model on the plpData to make predictions

Value

A dataframe containing the prediction for each person in the population with an attribute metaData containing prediction details.

---

predictProbabilities *Create predictive probabilities*

Description

Create predictive probabilities

Usage

```
predictProbabilities(predictiveModel, population, covariates)
```

Arguments

predictiveModel	An object of type predictiveModel as generated using fitPlp.
population	The population to calculate the prediction for
covariates	The covariate part of PlpData containing the covariates for the population

Details

Generates predictions for the population specified in plpData given the model.

Value

The value column in the result data.frame is: logistic: probabilities of the outcome, poisson: Poisson rate (per day) of the outome, survival: hazard rate (per day) of the outcome.

runPlp *runPlp - Train and evaluate the model*

Description

This provides a general framework for training patient level prediction models. The user can select various default feature selection methods or incorporate their own, The user can also select from a range of default classifiers or incorporate their own. There are three types of evaluations for the model patient (randomly splits people into train/validation sets) or year (randomly splits data into train/validation sets based on index year - older in training, newer in validation) or both (same as year spliting but checks there are no overlaps in patients within training set and validaiton set - any overlaps are removed from validation set)

Usage

```
runPlp(population, plpData, modelSettings, testSplit = "time",
  testFraction = 0.25, splitSeed = NULL, nfold = 3, indexes = NULL,
  save = NULL, saveModel = T, verbosity = futile.logger::INFO,
  timeStamp = FALSE, analysisId = NULL)
```

Arguments

`population`	The population created using createStudyPopulation() who will be used to develop the model
`plpData`	An object of type `plpData` - the patient level prediction data extracted from the CDM.
`modelSettings`	An object of class `modelSettings` created using one of the function: • logisticRegressionModel() A lasso logistic regression model • GBMclassifier() A gradient boosting machine • RFclassifier() A random forest model • GLMclassifier () A generalised linear model • KNNclassifier() A KNN model
`testSplit`	Either 'person' or 'time' specifying the type of evaluation used. 'time' find the date where testFraction of patients had an index after the date and assigns patients with an index prior to this date into the training set and post the date into the test set 'person' splits the data into test (1-testFraction of the data) and train (validationFraction of the data) sets. The split is stratified by the class label.
`testFraction`	The fraction of the data to be used as the test set in the patient split evaluation.
`splitSeed`	The seed used to split the test/train set when using a person type testSplit
`nfold`	The number of folds used in the cross validation (default 3)
`indexes`	A dataframe containing a rowId and index column where the index value of -1 means in the test set, and positive integer represents the cross validation fold (default is NULL)
`save`	The path to the directory where the models will be saved (if NULL uses working directory)
`saveModel`	Binary indicating whether to save the model once it is trained (default is T)

`verbosity` Sets the level of the verbosity. If the log level is at or higher in priority than the logger threshold, a message will print. The levels are:

- DEBUGHighest verbosity showing all debug statements
- TRACEShowing information about start and end of steps
- INFOShow informative information (Default)
- WARNShow warning messages
- ERRORShow error messages
- FATALBe silent except for fatal errors

`timeStamp` If TRUE a timestamp will be added to each logging statement. Automatically switched on for TRACE level.

`analysisId` Identifier for the analysis. It is used to create, e.g., the result folder. Default is a timestamp.

Details

Users can define a risk period of interest for the prediction of the outcome relative to index or use the cohprt dates. The user can then specify whether they wish to exclude patients who are not observed during the whole risk period, cohort period or experienced the outcome prior to the risk period.

Value

An object containing the model or location where the model is save, the data selection settings, the preprocessing and training settings as well as various performance measures obtained by the model.

`predict` A function that can be applied to new data to apply the trained model and make predictions

`model` A list of class `plpModel` containing the model, training metrics and model meta-data

`prediction` A dataframe containing the prediction for each person in the test set

`evalType` The type of evaluation that was performed ('person' or 'time')

`performanceTest`
A list detailing the size of the test sets

`performanceTrain`
A list detailing the size of the train sets

`time` The complete time taken to do the model framework

Examples

```
## Not run:
#******** EXAMPLE 1 *********
#load plpData:
plpData <- loadPlpData(file.path('C:','User','home','data'))

#create study population to develop model on
#require minimum of 365 days observation prior to at risk start
#no prior outcome and person must be observed for 365 after index (minTimeAtRisk)
#with risk window from 0 to 365 days after index
population <- createStudyPopulation(plpData,outcomeId=2042,
                                    firstExposureOnly = FALSE,
                                    washoutPeriod = 365,
                                    removeSubjectsWithPriorOutcome = TRUE,
```

```
                                    priorOutcomeLookback = 99999,
                                    requireTimeAtRisk = TRUE,
                                    minTimeAtRisk=365,
                                    riskWindowStart = 0,
                                    addExposureDaysToStart = FALSE,
                                    riskWindowEnd = 365,
                                    addExposureDaysToEnd = FALSE)

#lasso logistic regression predicting outcome 200 in cohorts 10
#using no feature selection with a time split evaluation with 30% in test set
#70% in train set where the model hyper-parameters are selected using 3-fold cross validation:
#and results are saved to file.path('C:','User','home')
model.lr <- lassoLogisticRegression.set()
mod.lr <- runPlp(population=population,
                        plpData= plpData,
                        modelSettings = model.lr ,
                        testSplit = 'time', testFraction=0.3,
                        nfold=3, indexes=NULL,
                        save=file.path('C:','User','home'),
                        verbosity='INFO')

#******** EXAMPLE 2 *********
# Gradient boosting machine with a grid search to select hyper parameters
# using the test/train/folds created for the lasso logistic regression above
model.gbm <- gradientBoostingMachine.set(rsampRate=c(0.5,0.9,1),csampRate=1,
                           ntrees=c(10,100), bal=c(F,T),
                           max_depth=c(4,5), learn_rate=c(0.1,0.01))
mod.gbm <- runPlp(population=population,
                        plpData= plpData,
                        modelSettings = model.gbm,
                        testSplit = 'time', testFraction=0.3,
                        nfold=3, indexes=mod.lr$indexes,
                        save=file.path('C:','User','home'))

## End(Not run)
```

`runPlpAnalyses` *Develop patient-level predcition models for multiple outcomes, target popuations and settings*

Description

Develop patient-level predcition models for multiple outcomes, target popuations and settings

Usage

```
runPlpAnalyses(outputFolder = getwd(), connectionDetails = NULL,
  cdmDatabaseSchema = NULL, oracleTempSchema = cdmDatabaseSchema,
  cohortDatabaseSchema = cdmDatabaseSchema, cohortTable = "cohort",
  outcomeDatabaseSchema = cdmDatabaseSchema, outcomeTable = "cohort",
  cdmVersion = "5", studyStartDate = "", studyEndDate = "",
  atRiskCohortIds = 1, outcomeIds = 2,
  covariateSettings = list(FeatureExtraction::createCovariateSettings(useCovariateDemographics
  = T, useCovariateDemographicsGender = T, useCovariateDemographicsRace = T,
```

```
  useCovariateDemographicsAge = T, useCovariateDemographicsYear = F,
  useCovariateDemographicsMonth = T, useCovariateConditionOccurrence = T,
  useCovariateConditionOccurrence365d = T),
  FeatureExtraction::createCovariateSettings(useCovariateDemographics = T,
  useCovariateDemographicsGender = T, useCovariateDemographicsRace = T,
  useCovariateDemographicsAge = T,      useCovariateDemographicsYear = F,
  useCovariateDemographicsMonth = T, useCovariateDrugExposure = T,
  useCovariateDrugExposure365d = T)),
  timeAtRisks = list(setTimeAtRisks(riskWindowEnd = 365),
  setTimeAtRisks(riskWindowEnd = 365 * 2)), modelSettings = NULL,
  internalValidation = "time", testFraction = 0.25, nfold = 3,
  splitSeed = NULL, indexes = NULL, verbosity = futile.logger::INFO)
```

Arguments

outputFolder	The directory to save the results and data to - needs read/write privileges
connectionDetails	An R object of type connectionDetails created using the function createConnectionDetails in the DatabaseConnector package.
cdmDatabaseSchema	The name of the database schema that contains the OMOP CDM instance. Requires read permissions to this database. On SQL Server, this should specifiy both the database and the schema, so for example 'cdm_instance.dbo'.
oracleTempSchema	For Oracle only: the name of the database schema where you want all temporary tables to be managed. Requires create/insert permissions to this database.
cohortDatabaseSchema	The name of the database schema that is the location where the cohort data used to define the at risk cohort is available. If cohortTable = DRUG_ERA, cohortDatabaseSchema is not used by assumed to be cdmSchema. Requires read permissions to this database.
cohortTable	The tablename that contains the at risk cohort. If cohortTable <> DRUG_ERA, then expectation is cohortTable has format of COHORT table: cohort_concept_id, SUBJECT_ID, COHORT_START_DATE, COHORT_END_DATE.
outcomeDatabaseSchema	The name of the database schema that is the location where the data used to define the outcome cohorts is available. If cohortTable = CONDITION_ERA, exposureDatabaseSchema is not used by assumed to be cdmSchema. Requires read permissions to this database.
outcomeTable	The tablename that contains the outcome cohorts. If outcomeTable <> CONDITION_OCCURRENCE, then expectation is outcomeTable has format of COHORT table: COHORT_DEFINITION_ID, SUBJECT_ID, COHORT_START_DATE, COHORT_END_DATE.
cdmVersion	Define the OMOP CDM version used: currently support "4" and "5".
studyStartDate	A calendar date specifying the minimum date that a cohort index date can appear. Date format is 'yyyymmdd'.
studyEndDate	A calendar date specifying the maximum date that a cohort index date can appear. Date format is 'yyyymmdd'. Important: the study end data is also used to truncate risk windows, meaning no outcomes beyond the study end date will be considered.

atRiskCohortIds
: A vector containing the unique identifiers to define the at risk cohorts. Each at risk cohortId is used to select the cohort_concept_id in the cohort-like table.

outcomeIds
: A list of cohort_definition_ids used to define outcomes.

covariateSettings
: An object of type covariateSettings as created using the createCovariateSettings function in the FeatureExtraction package. This can be a list of multiple settings.

timeAtRisks
: A list detailing the time at risk intervals that willbe used to create the prediciton models (the period of time we wish to predict the outcome occurence within) created using the function setTimeAtRisks.

modelSettings
: A list of model settings created using the setGradientBoostingMachine, setRandomForest, setLassoLogisticRegression, setNaiveBayes or setKNN.

internalValidation
: The type of internal validation for the model. Either 'person' which stratifies by outcome to partion into test/train sets or 'time' which picks a set date and all people with an at risk cohort start date prior to this join the train set and people after join the test set.

testFraction
: The fracion of the target population to include into the test set

nfold
: The number of cross validation folds to apply when finding the optimal hyperparameters

splitSeed
: (default NULL) The seed used to do the random split for internalValidation='person'

indexes
: The nfold validation indexes

verbosity
: Sets the level of the verbosity. If the log level is at or higher in priority than the logger threshold, a message will print. The levels are:
  - DEBUGHighest verbosity showing all debug statements
  - TRACEShowing information about start and end of steps
  - INFOShow informative information (Default)
  - WARNShow warning messages
  - ERRORShow error messages
  - FATALBe silent except for fatal errors

---

savePlpData *Save the cohort data to folder*

Description

savePlpData saves an object of type plpData to folder.

Usage

```
savePlpData(plpData, file, envir = NULL)
```

Arguments

plpData
: An object of type plpData as generated using getDbPlpData.

file
: The name of the folder where the data will be written. The folder should not yet exist.

envir
: The environment for to evaluate variables when saving

Details

The data will be written to a set of files in the folder specified by the user.

Examples

```
# todo
```

---

savePlpModel *Saves the plp model*

Description

Saves the plp model

Usage

```
savePlpModel(plpModel, dirPath)
```

Arguments

plpModel	A trained classifier returned by running runPlp()$model
dirPath	A location to save the model to

Details

Saves the plp model to a user specificed folder

---

savePlpResult *Saves the result from runPlp into the location directory*

Description

Saves the result from runPlp into the location directory

Usage

```
savePlpResult(result, dirPath)
```

Arguments

result	The result of running runPlp()
dirPath	The directory to save the csv

Details

Saves the result from runPlp into the location directory

savePrediction *Saves the prediction dataframe to csv*

Description

Saves the prediction dataframe to csv

Usage

```
savePrediction(prediction, dirPath)
```

Arguments

prediction	The prediciton data.frame
dirPath	The directory to save the csv

Details

Saves the prediction data frame returned by predict.R to a csv file

sensitivity *Calculate the sensitivity*

Description

Calculate the sensitivity

Usage

```
sensitivity(TP, TN, FN, FP)
```

Arguments

TP	Number of true positives
TN	Number of true negatives
FN	Number of false negatives
FP	Number of false positives

Details

Calculate the sensitivity

Value

sensitivity value

setGradientBoostingMachine

Create setting for gradient boosting machine model using gbm_xgboost implementation

Description

Create setting for gradient boosting machine model using gbm_xgboost implementation

Usage

```
setGradientBoostingMachine(ntrees = c(10, 100), nthread = 20,
  max_depth = 6, min_rows = 20, learn_rate = 0.1, seed = NULL)
```

Arguments

ntrees	The number of trees to build
nthread	The number of computer threads to (how many cores do you have?)
max_depth	Maximum number of interactions - a large value will lead to slow model training
min_rows	The minimum number of rows required at each end node of the tree
learn_rate	The boosting learn rate
seed	An option to add a seed when training the final model

Examples

```
model.gbm <- setGradientBoostingMachine(ntrees=c(10,100), nthread=20,
                          max_depth=c(4,6), learn_rate=c(0.1,0.3))
```

setKNN

Create setting for knn model

Description

Create setting for knn model

Usage

```
setKNN(k = 1000, indexFolder = file.path(getwd(), "knn"))
```

Arguments

k	The number of neighbors to consider
indexFolder	The directory where the results and intermediate steps are output

Examples

```
model.knn <- setKNN(k=10000)
```

setLassoLogisticRegression *Create setting for lasso logistic regression*

Description

Create setting for lasso logistic regression

Usage

```
setLassoLogisticRegression(variance = 0.01, seed = NULL)
```

Arguments

variance	a single value used as the starting value for the automatic lambda search
seed	An option to add a seed when training the model

Examples

```
model.lr <- setLassoLogisticRegression()
```

setMLP *Create setting for neural network model with python*

Description

Create setting for neural network model with python

Usage

```
setMLP(size = 4, alpha = 1e-05, seed = NULL)
```

Arguments

size	The number of hidden nodes
alpha	The l2 regularisation
seed	A seed for the model

Examples

```
## Not run:
model.mlp <- setMLP(size=4, alpha=0.00001, seed=NULL)

## End(Not run)
```

setNaiveBayes *Create setting for naive bayes model with python*

Description

Create setting for naive bayes model with python

Usage

```
setNaiveBayes()
```

Examples

```
## Not run:
model.nb <- setNaiveBayes()

## End(Not run)
```

setRandomForest *Create setting for random forest model with python (very fast)*

Description

Create setting for random forest model with python (very fast)

Usage

```
setRandomForest(mtries = -1, ntrees = c(10, 500), max_depth = 17,
  varImp = T, seed = NULL)
```

Arguments

mtries	The number of features to include in each tree (-1 defaults to square root of total features)
ntrees	The number of trees to build
max_depth	Maximum number of interactions - a large value will lead to slow model training
varImp	Perform an initial variable selection prior to fitting the model to select the useful variables
seed	An option to add a seed when training the final model

Examples

```
## Not run:
model.rf <- setRandomForest(mtries=c(-1,5,20),  ntrees=c(10,100),
                            max_depth=c(5,20))

## End(Not run)
```

setTimeAtRisk *setTimeAtRisk*

Description

create the timeAtRisks for the multiple analysis studies

Usage

```
setTimeAtRisk(includeAllOutcomes = T, firstExposureOnly = F,
  washoutPeriod = 0, removeSubjectsWithPriorOutcome = T,
  priorOutcomeLookback = 99999, riskWindowStart = 1,
  addExposureDaysToStart = F, riskWindowEnd = 365,
  addExposureDaysToEnd = F, requireTimeAtRisk = T,
  minTimeAtRisk = riskWindowEnd - riskWindowStart)
```

Arguments

includeAllOutcomes	Do you want to include people who have the outcome but are not observed for the whole at risk period?
firstExposureOnly	Only consider the first time occurence of the outcome?
washoutPeriod	The minimum prior observation in days a person required to be included
removeSubjectsWithPriorOutcome	Remove people who have the outcome some period before the time at risk?
priorOutcomeLookback	The number of days prior to investigate for the variable removeSubjectsWithPriorOutcome
riskWindowStart	The number of days after the at risk popualtion subject's index date to start the time at risk period
addExposureDaysToStart	Should the risk window start be relative to the index end date instead?
riskWindowEnd	The number of days after the at risk popualtion subject's index date to end the time at risk period
addExposureDaysToEnd	Should the risk window end be relative to the index end date instead?
requireTimeAtRisk	Should you only include people with a minimum time at risk period?
minTimeAtRisk	If requireTimeAtRisk is TRUE, then this is the minimum number of days a person must be at risk

`similarPlpData` *Extract new plpData using plpModel settings use metadata in plpModel to extract similar data and population for new databases:*

Description

Extract new plpData using plpModel settings use metadata in plpModel to extract similar data and population for new databases:

Usage

```
similarPlpData(plpModel = NULL, newConnectionDetails = NULL,
  newCdmDatabaseSchema = NULL, newCohortDatabaseSchema = NULL,
  newCohortTable = NULL, newCohortId = NULL,
  newOutcomeDatabaseSchema = NULL, newOutcomeTable = NULL,
  newOutcomeId = NULL)
```

Arguments

`plpModel`	The trained PatientLevelPrediction model or object returned by runPlp()
`newConnectionDetails`	The connectionDetails for the new database
`newCdmDatabaseSchema`	The database schema for the new CDM database
`newCohortDatabaseSchema`	The database schema where the cohort table is stored
`newCohortTable`	The table name of the cohort table
`newCohortId`	The cohort_definition_id for the cohort of at risk people
`newOutcomeDatabaseSchema`	The database schema where the outcome table is stored
`newOutcomeTable`	The table name of the outcome table
`newOutcomeId`	The cohort_definition_id for the outcome

Examples

```
## Not run:
# set the connection
connectionDetails <- DatabaseConnector::createConnectionDetails()

# load the model and data
plpModel <- loadPlpModel("C:/plpmodel")

# extract the new data in the 'newData.dbo' schema using the model settings
newDataList <- similarPlpData(plpModel=plpModel,
                              newConnectionDetails = connectionDetails,
                              newCdmDatabaseSchema = 'newData.dbo',
                              newCohortDatabaseSchema = 'newData.dbo',
                              newCohortTable = 'cohort',
                              newCohortId = 1,
                              newOutcomeDatabaseSchema = 'newData.dbo',
```

```
                                newOutcomeTable = 'outcome',
                                newOutcomeId = 2)

# get the prediction:
prediction <- applyModel(newDataList$population, newDataList$plpData, plpModel)$prediction

## End(Not run)
```

`simulatePlpData` *Generate simulated data*

Description

`simulateplpData` creates a plpData object with simulated data.

Usage

```
simulatePlpData(plpDataSimulationProfile, n = 10000)
```

Arguments

`plpDataSimulationProfile`	An object of type `plpDataSimulationProfile` as generated using the `createplpDataSimulationProfile` function.
`n`	The size of the population to be generated.

Details

This function generates simulated data that is in many ways similar to the original data on which the simulation profile is based. The contains same outcome, comparator, and outcome concept IDs, and the covariates and their 1st order statistics should be comparable.

Value

An object of type `plpData`.

`specificity` *Calculate the specificity*

Description

Calculate the specificity

Usage

```
specificity(TP, TN, FN, FP)
```

Arguments

TP	Number of true positives
TN	Number of true negatives
FN	Number of false negatives
FP	Number of false positives

Details

Calculate the specificity

Value

specificity value

---

timeSplitter *Split test/train data by time and then partitions training set into random folds stratified by class*

Description

Split test/train data by time and then partitions training set into random folds stratified by class

Usage

```
timeSplitter(population, test = 0.3, nfold = 3, seed = NULL)
```

Arguments

population	An object created using createStudyPopulation().
test	A real number between 0 and 1 indicating the test set fraction of the data
nfold	An integer >= 1 specifying the number of folds used in cross validation
seed	If set a fixed seed is used, otherwise a random split is performed

Details

Returns a dataframe of rowIds and indexes with a -1 index indicating the rowId belongs to the test set and a positive integer index value indicating the rowId's cross valiation fold within the train set.

Value

A dataframe containing the columns: rowId and index

`toSparseM` *Convert the plpData in COO format into a sparse R matrix*

Description

Converts the standard plpData to a sparse matrix

Usage

```
toSparseM(plpData, population, map = NULL)
```

Arguments

`plpData`	An object of type `plpData` with covariate in coo format - the patient level prediction data extracted from the CDM.
`population`	The population to include in the matrix
`map`	A covariate map (telling us the column number for covariates)

Details

This function converts the covariate file from ffdf in COO format into a sparse matrix from the package Matrix

Value

Returns a list, containing the data as a sparse matrix, the plpData covariateRef and a data.frame named map that tells us what covariate corresponds to each column This object is a list with the following components:

data A sparse matrix with the rows corresponding to each person in the plpData and the columns corresponding to the covariates.

covariateRef The plpData covariateRef.

map A data.frame containing the data column ids and the corresponding covariateId from covariateRef.

Examples

```
#TODO
```

`toSparsePython` *Convert the plpData in COO format into a sparse python matrix*

Description

Converts the standard plpData to a sparse matrix firectly into python

Usage

```
toSparsePython(plpData, population, map = NULL)
```

Arguments

`plpData`	An object of type `plpData` with covariate in coo format - the patient level prediction data extracted from the CDM.
`population`	The population to include in the matrix
`map`	A covariate map (telling us the column number for covariates)

Details

This function converts the covariate file from ffdf in COO format into a sparse matrix from the package Matrix

Value

Returns a list, containing the python object name of the sparse matrix, the plpData covariateRef and a data.frame named map that tells us what covariate corresponds to each column This object is a list with the following components:

data The python object name containing a sparse matrix with the rows corresponding to each person in the plpData and the columns corresponding to the covariates.

covariateRef The plpData covariateRef.

map A data.frame containing the data column ids and the corresponding covariateId from covariateRef.

Examples

```
#TODO
```

Index