

Package ‘PatientLevelPrediction’

June 3, 2020

Type Package

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

Version 4.0.0

Date 2020-06-04

Maintainer Jenna Reps <jreps@its.jnj.com>

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

URL <https://ohdsi.github.io/PatientLevelPrediction>, <https://github.com/OHDSI/PatientLevelPrediction>

BugReports <https://github.com/OHDSI/PatientLevelPrediction/issues>

Depends R (>= 3.3.0),
DatabaseConnector (>= 3.0.0),
FeatureExtraction (>= 3.0.0),
Cyclops (>= 2.0.0)

Imports ggplot2,
gridExtra,
PRROC,
magrittr,
foreach,
doParallel,
dplyr,
bit,
Andromeda,
plyr,
survAUC,
Rcpp (>= 0.11.2),
SqlRender (>= 1.1.3),
survival,
xgboost,
Matrix,
AUC,

utils,
 methods,
 reshape2,
 officer,
 diagram,
 tidyr,
 viridisLite,
 RCurl,
 RJSONIO,
 keras,
 slam,
 ParallelLogger,
 reticulate (> 1.6),
 tools,
 plotly,
 zeallot,
 htmltools,
 tensorflow

Suggests shiny,
 DT,
 htmlwidgets (> 0.8),
 shinydashboard,
 shinycssloaders,
 cowplot,
 testthat,
 pROC,
 gnm,
 knitr,
 rmarkdown,
 scoring,
 Metrics,
 SparseM,
 ResourceSelection,
 BigKnn (>= 1.0.0),
 aws.s3,
 devtools,
 rms,
 survminer

Remotes ohdsi/FeatureExtraction,
 ohdsi/BigKnn,
 ohdsi/Andromeda

LinkingTo Rcpp

NeedsCompilation yes

RoxygenNote 7.1.0

Encoding UTF-8

R topics documented:

| | |
|------------------------------|---|
| accuracy | 5 |
| applyEnsembleModel | 6 |

| | |
|---|----|
| applyModel | 7 |
| averagePrecision | 8 |
| brierScore | 8 |
| calibrationLine | 9 |
| checkPlpInstallation | 9 |
| combinePlpModelSettings | 10 |
| computeAuc | 10 |
| computeAucFromDataFrames | 11 |
| configurePython | 11 |
| createLearningCurve | 12 |
| createLearningCurvePar | 14 |
| createLrSql | 16 |
| createPlpJournalDocument | 17 |
| createPlpModelSettings | 18 |
| createPlpReport | 19 |
| createStudyPopulation | 20 |
| createStudyPopulationSettings | 22 |
| diagnostic | 23 |
| diagnosticOddsRatio | 26 |
| drawAttritionDiagramPlp | 26 |
| evaluateMultiplePlp | 27 |
| evaluatePlp | 28 |
| externalValidatePlp | 29 |
| f1Score | 30 |
| falseDiscoveryRate | 31 |
| falseNegativeRate | 31 |
| falseOmissionRate | 32 |
| falsePositiveRate | 32 |
| fitGLMMModel | 33 |
| fitPlp | 34 |
| getAttritionTable | 35 |
| getCalibration | 35 |
| getPlpData | 36 |
| getPlpTable | 38 |
| getPredictionDistribution | 39 |
| getThresholdSummary | 39 |
| interpretInstallCode | 40 |
| launchDiagnosticsExplorer | 40 |
| listAppend | 41 |
| loadEnsemblePlpModel | 41 |
| loadEnsemblePlpResult | 42 |
| loadPlpData | 42 |
| loadPlpModel | 43 |
| loadPlpResult | 43 |
| loadPrediction | 44 |
| loadPredictionAnalysisList | 44 |
| negativeLikelihoodRatio | 45 |
| negativePredictiveValue | 46 |
| outcomeSurvivalPlot | 46 |
| PatientLevelPrediction | 47 |
| personSplitter | 47 |
| plotDemographicSummary | 48 |

| | |
|----------------------------|----|
| plotF1Measure | 48 |
| plotGeneralizability | 49 |
| plotLearningCurve | 50 |
| plotPlp | 51 |
| plotPrecisionRecall | 51 |
| plotPredictedPDF | 52 |
| plotPredictionDistribution | 52 |
| plotPreferencePDF | 53 |
| plotRoc | 53 |
| plotSmoothCalibration | 54 |
| plotSparseCalibration | 55 |
| plotSparseCalibration2 | 56 |
| plotSparseRoc | 56 |
| plotVariableScatterplot | 57 |
| plpDataSimulationProfile | 57 |
| positiveLikelihoodRatio | 58 |
| positivePredictiveValue | 58 |
| predictAndromeda | 59 |
| predictPlp | 59 |
| predictProbabilities | 60 |
| randomSplitter | 61 |
| registerParallelBackend | 61 |
| registerSequentialBackend | 62 |
| runEnsembleModel | 62 |
| runPlp | 64 |
| runPlpAnalyses | 67 |
| saveEnsemblePlpModel | 69 |
| saveEnsemblePlpResult | 70 |
| savePlpData | 70 |
| savePlpModel | 71 |
| savePlpResult | 71 |
| savePrediction | 72 |
| savePredictionAnalysisList | 72 |
| sensitivity | 73 |
| setAdaBoost | 74 |
| setCIReNN | 74 |
| setCNNTorch | 76 |
| setCovNN | 77 |
| setCovNN2 | 78 |
| setCoxModel | 79 |
| setDecisionTree | 80 |
| setDeepNN | 80 |
| setGBMSurvival | 81 |
| setGradientBoostingMachine | 83 |
| setKNN | 83 |
| setLassoLogisticRegression | 84 |
| setLRTorch | 84 |
| setMLP | 85 |
| setMLPTorch | 86 |
| setNaiveBayes | 87 |
| setPythonEnvironment | 87 |
| setRandomForest | 88 |

| | |
|--|------------|
| <i>accuracy</i> | 5 |
| setRandomForestQuantileRegressor | 88 |
| setRNNTorch | 90 |
| setSagemakerBinary | 91 |
| similarPlpData | 91 |
| simulatePlpData | 93 |
| specificity | 94 |
| subjectSplitter | 94 |
| timeSplitter | 95 |
| toSparseM | 96 |
| toSparseTorchPython | 97 |
| transferLearning | 98 |
| transportModel | 99 |
| transportPlp | 99 |
| viewMultiplePlp | 100 |
| viewPlp | 101 |
| Index | 102 |

| | |
|-----------------|-------------------------------|
| <i>accuracy</i> | <i>Calculate the accuracy</i> |
|-----------------|-------------------------------|

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

| | |
|--------------------|--|
| applyEnsembleModel | <i>Apply trained ensemble 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)</i> |
|--------------------|--|

Description

Apply trained ensemble 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

```
applyEnsembleModel(
  population,
  dataList,
  ensembleModel,
  analysisId = NULL,
  calculatePerformance = T
)
```

Arguments

| | |
|----------------------|--|
| population | The population of people who you want to predict the risk for |
| dataList | The plpData list for the population |
| ensembleModel | The trained ensemble model returned by running runEnsembleModel |
| analysisId | The analysis ID, which is the ID of running ensemble model training. |
| calculatePerformance | Whether to also calculate the performance metrics [default TRUE] |

Examples

```
## Not run:
# load the model and data
plpData <- loadPlpData("plpdata/")
results <- PatientLevelPrediction::runEnsembleModel(population,
  dataList = list(plpData, plpData),
  modelList = list(model, model),
  testSplit = "person",
  testFraction = 0.2,
  nfold = 3,
  splitSeed = 1000,
  ensembleStrategy = "stacked")

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

```
# get the prediction, please make sure the ensemble strategy for training and apply is the same:
prediction <- applyEnsembleModel(population,
                                dataList = list(plpData, plpData),
                                ensembleModel = results,
                                analysisId = NULL)$prediction

## End(Not run)
```

| | |
|------------|---|
| applyModel | <i>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)</i> |
|------------|---|

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,
  calculatePerformance = T,
  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 |
| calculatePerformance | Whether to also calculate the performance metrics [default TRUE] |
| 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 | <i>Calculate the average precision</i> |
|------------------|--|

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 prediction object

Value

The average precision

| | |
|------------|-------------------|
| brierScore | <i>brierScore</i> |
|------------|-------------------|

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

| | |
|-----------------|------------------------|
| calibrationLine | <i>calibrationLine</i> |
|-----------------|------------------------|

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

| | |
|----------------------|--|
| checkPlpInstallation | <i>Check PatientLevelPrediction and its dependencies are correctly installed</i> |
|----------------------|--|

Description

Check PatientLevelPrediction and its dependencies are correctly installed

Usage

```
checkPlpInstallation(connectionDetails = NULL, python = T)
```

Arguments

connectionDetails An R object of type
 connectionDetails created using the function createConnectionDetails in
 the DatabaseConnector package.
python Whether to test the python models

Details

This function checks whether PatientLevelPrediction and its dependencies are correctly installed. This will check the database connectivity, some models, and large data object handling (ff).

combinePlpModelSettings

combine two objects specifying multiple Plp model settings

Description

combine two objects specifying multiple Plp model settings

Usage

```
combinePlpModelSettings(plpModelSetting1, plpModelSetting2)
```

Arguments

plpModelSetting1

A combination of model, covariate and population settings

plpModelSetting2

A combination of model, covariate and population settings

Details

Takes two output of running createPlpModelSettings() and combined them

Value

A list containing a dataframe settingLookupTable containing all the model, covariate and population combination details, a list models containing all the model settings, a list covariateSettings containing all the covariate settings and a list populationSettings containing all the population settings.

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 percent 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 percent 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.

```
configurePython
```

Sets up a virtual environment to use for PLP (can be conda or python)

Description

Sets up a virtual environment to use for PLP (can be conda or python)

Usage

```
configurePython(envname = "PLP", envtype = NULL)
```

Arguments

| | |
|---------|---|
| envname | A string for the name of the virtual environment (default is 'PLP') |
| envtype | An option for specifying the environment as 'conda' or 'python'. If NULL then the default is 'conda' for windows users and 'python' for non-windows users |

Details

This function creates a virtual environment that can be used by PatientLevelPrediction and installs all the required package dependancies. If using python, pip must be set up.

| | |
|---------------------|----------------------------|
| createLearningCurve | <i>createLearningCurve</i> |
|---------------------|----------------------------|

Description

Creates a learning curve object, which can be plotted using the plotLearningCurve() function.

Usage

```
createLearningCurve(
  population,
  plpData,
  modelSettings,
  testSplit = "person",
  testFraction = 0.25,
  trainFractions = c(0.25, 0.5, 0.75),
  splitSeed = NULL,
  nfold = 3,
  indexes = NULL,
  verbosity = "TRACE",
  clearffTemp = FALSE,
  minCovariateFraction = 0.001,
  normalizeData = T,
  saveDirectory = getwd(),
  savePlpData = F,
  savePlpResult = F,
  savePlpPlots = F,
  saveEvaluation = F,
  timeStamp = FALSE,
  analysisId = NULL
)
```

Arguments

| | |
|---------------|--|
| population | The population created using createStudyPopulation() that 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: <ul style="list-style-type: none"> • setLassoLogisticRegression - a lasso logistic regression model |

| | |
|----------------------|--|
| | <ul style="list-style-type: none"> • setGradientBoostingMachine - a gradient boosting machine • setRandomForest - a random forest model • setKNN - a k-nearest neighbour model |
| testSplit | Specifies the type of evaluation used. Can be either 'person' or 'time'. The value 'time' finds the date that splits the population into the testing and training fractions provided. Patients with an index after this date are assigned to the test set and patients with an index prior to this date are assigned to the training set. The value 'person' splits the data randomly into testing and training sets according to fractions provided. The split is stratified by the class label. |
| testFraction | The fraction of the data, which will be used as the testing set in the patient split evaluation. |
| trainFractions | A list of training fractions to create models for. |
| splitSeed | The seed used to split the testing and training set when using a 'person' type split |
| 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). |
| verbosity | <p>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:</p> <ul style="list-style-type: none"> • DEBUG - highest verbosity showing all debug statements • TRACE - showing information about start and end of steps • INFO - show informative messages (default) • WARN - show warning messages • ERROR - show error messages • FATAL - be silent except for fatal errors |
| clearffTemp | Clears the temporary ff-directory after each iteration. This can be useful, if the fitted models are large. |
| minCovariateFraction | Minimum covariate prevalence in population to avoid removal during preprocessing. |
| normalizeData | Whether to normalise the data |
| saveDirectory | Location to save log and results |
| savePlpData | Whether to save the plpData |
| savePlpResult | Whether to save the plpResult |
| savePlpPlots | Whether to save the plp plots |
| saveEvaluation | Whether to save the plp performance csv files |
| timeStamp | Include a timestamp in the log |
| analysisId | The analysis unique identifier |

Value

A learning curve object containing the various performance measures obtained by the model for each training set fraction. It can be plotted using plotLearningCurve.

Examples

```
## Not run:
# define model
modelSettings = PatientLevelPrediction::setLassoLogisticRegression()

# create learning curve
learningCurve <- PatientLevelPrediction::createLearningCurve(population,
                                                             plpData,
                                                             modelSettings)

# plot learning curve
PatientLevelPrediction::plotLearningCurve(learningCurve)

## End(Not run)
```

```
createLearningCurvePar
```

```
    createLearningCurvePar
```

Description

Creates a learning curve in parallel, which can be plotted using the `plotLearningCurve()` function. Currently this functionality is only supported by Lasso Logistic Regression.

Usage

```
createLearningCurvePar(
  population,
  plpData,
  modelSettings,
  testSplit = "stratified",
  testFraction = 0.25,
  trainFractions = c(0.25, 0.5, 0.75),
  splitSeed = NULL,
  nfold = 3,
  indexes = NULL,
  verbosity = "TRACE",
  minCovariateFraction = 0.001,
  normalizeData = T,
  saveDirectory = getwd(),
  savePlpData = F,
  savePlpResult = F,
  savePlpPlots = F,
  saveEvaluation = F,
  timeStamp = FALSE,
  analysisId = "lc-",
  cores = NULL
)
```

Arguments

| | |
|----------------------|---|
| population | The population created using createStudyPopulation() that 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. Currently only one model is supported: <ul style="list-style-type: none"> • setLassoLogisticRegression - a lasso logistic regression model |
| testSplit | Specifies the type of evaluation used. Can be either 'person' or 'time'. The value 'time' finds the date that splits the population into the testing and training fractions provided. Patients with an index after this date are assigned to the test set and patients with an index prior to this date are assigned to the training set. The value 'person' splits the data randomly into testing and training sets according to fractions provided. The split is stratified by the class label. |
| testFraction | The fraction of the data, which will be used as the testing set in the patient split evaluation. |
| trainFractions | A list of training fractions to create models for. |
| splitSeed | The seed used to split the testing and training set when using a 'person' type split |
| 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). |
| 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: <ul style="list-style-type: none"> • DEBUG - highest verbosity showing all debug statements • TRACE - showing information about start and end of steps • INFO - show informative messages (default) • WARN - show warning messages • ERROR - show error messages • FATAL - be silent except for fatal errors |
| minCovariateFraction | Minimum covariate prevalence in population to avoid removal during preprocessing. |
| normalizeData | Whether to normalise the data |
| saveDirectory | Location to save log and results |
| savePlpData | Whether to save the plpData |
| savePlpResult | Whether to save the plpResult |
| savePlpPlots | Whether to save the plp plots |
| saveEvaluation | Whether to save the plp performance csv files |
| timestamp | Include a timestamp in the log |
| analysisId | The analysis unique identifier |
| cores | The number of cores to use |

Value

A learning curve object containing the various performance measures obtained by the model for each training set fraction. It can be plotted using `plotLearningCurve`.

Examples

```
## Not run:
# define model
modelSettings = setLassoLogisticRegression()

# register parallel backend
registerParallelBackend()

# create learning curve
learningCurve <- createLearningCurvePar(population,
                                         plpData,
                                         modelSettings)

# plot learning curve
plotLearningCurve(learningCurve)

## End(Not run)
```

createLrSql

Convert logistic regression model to sql code...

Description

Convert logistic regression model to sql code...

Usage

```
createLrSql(
  models,
  modelNames,
  covariateConstructionName = "prediction",
  modelTable = "#model_table",
  analysisId = 111,
  e = environment(),
  databaseOutput = NULL
)
```

Arguments

| | |
|--|--|
| <code>models</code> | A trianed plp model. |
| <code>modelNames</code> | A name used in the covariate function names (no spaces) |
| <code>covariateConstructionName</code> | the name used for the create covariate function |
| <code>modelTable</code> | The temporary table name storing the model details |
| <code>analysisId</code> | The covariate analysis_id |
| <code>e</code> | The environment to output the covariate setting functions to |
| <code>databaseOutput</code> | If you want to output to go inot a cohort table add the "database.schema.tablename" here |

Details

This function is used to create custom covariates for a logistic regression model (currently only supports, demographics/conditions/drug/procedures/observations and measurement concepts)

```
createPlpJournalDocument
      createPlpJournalDocument
```

Description

Creates a template for a prediction journal paper with the characteristics/results filled in

Usage

```
createPlpJournalDocument(
  plpResult = NULL,
  plpValidation = NULL,
  plpData = NULL,
  targetName = "<target population>",
  outcomeName = "<outcome>",
  table1 = F,
  connectionDetails = NULL,
  includeTrain = FALSE,
  includeTest = TRUE,
  includePredictionPicture = TRUE,
  includeAttritionPlot = TRUE,
  outputLocation = file.path(getwd(), "plp_journal_document.docx"),
  save = T
)
```

Arguments

| | |
|--------------------------|---|
| plpResult | An object of type plpResult returned by running runPlp() |
| plpValidation | An object of type validatePlp returned by running externalValidatePlp() |
| plpData | The plpData |
| targetName | A string with the target description name |
| outcomeName | A string with the outcome description name |
| table1 | Whether to include table1 (characteristics) |
| connectionDetails | The connection required to calculate the characteristics |
| includeTrain | Whether to include the train set performance |
| includeTest | Whether to include the test set performance |
| includePredictionPicture | Whether to include a picture detailing the prediction problem |
| includeAttritionPlot | Whether to include the attrition plot |
| outputLocation | The location to write the document to |
| save | If false this function returns the document and does not save to outputLocation |

Details

The function creates a word document containing the analysis details, data summary and prediction model results.

Value

A work document containing the selected outputs within the user's directory at location specified in outputLocation

`createPlpModelSettings`

create a an object specifying the multiple Plp model settings

Description

create a an object specifying the multiple Plp model settings

Usage

```
createPlpModelSettings(modelList, covariateSettingList, populationSettingList)
```

Arguments

| | |
|------------------------------------|-------------------------------|
| <code>modelList</code> | A list of model settings |
| <code>covariateSettingList</code> | A list of covariate settings |
| <code>populationSettingList</code> | A list of population settings |

Details

Takes a list of models, covariates, population and returns the cartesian product combining all settings.

Value

A list containing a dataframe `settingLookupTable` containing all the model, covariate and population combination details, a list `models` containing all the model settings, a list `covariateSettings` containing all the covariate settings and a list `populationSettings` containing all the population settings.

| | |
|-----------------|------------------------|
| createPlpReport | <i>createPlpReport</i> |
|-----------------|------------------------|

Description

Creates a word document report of the prediction

Usage

```
createPlpReport(
  plpResult = NULL,
  plpValidation = NULL,
  plpData = NULL,
  targetName = "<target population>",
  outcomeName = "<outcome>",
  targetDefinition = NULL,
  outcomeDefinition = NULL,
  outputLocation = file.path(getwd(), "plp_report.docx"),
  save = T
)
```

Arguments

| | |
|-------------------|---|
| plpResult | An object of type plpResult returned by running runPlp() |
| plpValidation | An object of type validatePlp returned by running externalValidatePlp() |
| plpData | The plpData |
| targetName | A string with the target description name |
| outcomeName | A string with the outcome description name |
| targetDefinition | The cohort details |
| outcomeDefinition | The cohort details |
| outputLocation | The location to write the document to |
| save | If false the output of the function of the function is the document rather than creating the document in outputLocation |

Details

The function creates a word document containing the analysis details, data summary and prediction model results.

Value

A work document containing the selected outputs within the user's directory at location specified in outputLocation

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 = F,
  minTimeAtRisk = 365,
  riskWindowStart = 0,
  startAnchor = "cohort start",
  riskWindowEnd = 365,
  endAnchor = "cohort start",
  verbosity = "INFO",
  addExposureDaysToStart,
  addExposureDaysToEnd,
  ...
)
```

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 minimum 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 startAnchor. |
| startAnchor | The anchor point for the start of the risk window. Can be "cohort start" or "cohort end". |
| riskWindowEnd | The end of the risk window (in days) relative to the endAnchor parameter |
| endAnchor | The anchor point for the end of the risk window. Can be "cohort start" or "cohort end". |
| 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: <ul style="list-style-type: none"> • 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 |
| addExposureDaysToStart | DEPRECATED: Add the length of exposure the start of the risk window? Use startAnchor instead. |
| addExposureDaysToEnd | DEPRECATED: Add the length of exposure the risk window? Use endAnchor instead. |
| ... | 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

createStudyPopulationSettings
create the study population settings

Description

create the study population settings

Usage

```
createStudyPopulationSettings(  
  binary = T,  
  includeAllOutcomes = T,  
  firstExposureOnly = FALSE,  
  washoutPeriod = 0,  
  removeSubjectsWithPriorOutcome = TRUE,  
  priorOutcomeLookback = 99999,  
  requireTimeAtRisk = T,  
  minTimeAtRisk = 364,  
  riskWindowStart = 1,  
  startAnchor = "cohort start",  
  addExposureDaysToStart,  
  riskWindowEnd = 365,  
  endAnchor = "cohort start",  
  addExposureDaysToEnd,  
  verbosity = "INFO"  
)
```

Arguments

| | |
|--------------------------------|--|
| 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 minimum 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). |

| | |
|------------------------|---|
| startAnchor | The anchor point for the start of the risk window. Can be "cohort start" or "cohort end". |
| addExposureDaysToStart | DEPRECATED: Add the length of exposure the start of the risk window? Use startAnchor instead. |
| riskWindowEnd | The end of the risk window (in days) relative to the index data (+ days of exposure if the addExposureDaysToEnd parameter is specified). |
| endAnchor | The anchor point for the end of the risk window. Can be "cohort start" or "cohort end". |
| addExposureDaysToEnd | DEPRECATED: Add the length of exposure the risk window? Use endAnchor instead. |
| verbosity | <p>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:</p> <ul style="list-style-type: none"> • 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 |

#'

Details

Takes as input the inputs to create study population

Value

A list containing all the settings required for creating the study population

| | |
|------------|--|
| diagnostic | <i>diagnostic - Investigates the prediction problem settings - use before training a model</i> |
|------------|--|

Description

This function runs a set of prediction diagnoses to help pick a suitable T, O, TAR and determine whether the prediction problem is worth executing.

Usage

```
diagnostic(
  plpData = NULL,
  cdmDatabaseName,
  connectionDetails,
  cdmDatabaseSchema,
  oracleTempSchema = NULL,
  cohortId,
  cohortName = cohortId,
```

```

outcomeIds,
outcomeNames = outcomeIds,
cohortDatabaseSchema,
cohortTable = "cohort",
outcomeDatabaseSchema = cohortDatabaseSchema,
outcomeTable = cohortTable,
cdmVersion = 5,
riskWindowStart = 1,
startAnchor = "cohort start",
riskWindowEnd = 365,
endAnchor = "cohort start",
outputFolder = NULL,
sampleSize = NULL,
minCellCount = 5
)

```

Arguments

| | |
|------------------------------------|--|
| <code>plpData</code> | The data object to do the diagnostic on - if NULL you need to specify the connection settings below |
| <code>cdmDatabaseName</code> | Name of the database |
| <code>connectionDetails</code> | An R object of type <code>connectionDetails</code> created using the function <code>createConnectionDetails</code> in the <code>DatabaseConnector</code> package. |
| <code>cdmDatabaseSchema</code> | The name of the database schema that contains the OMOP CDM instance. Requires read permissions to this database. On SQL Server, this should specify both the database and the schema, so for example <code>'cdm_instance.dbo'</code> . |
| <code>oracleTempSchema</code> | 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. |
| <code>cohortId</code> | A unique identifier to define the at risk cohorts. <code>CohortId</code> is used to select the <code>cohort_concept_id</code> in the cohort-like table. |
| <code>cohortName</code> | A string specifying the name of the target cohort |
| <code>outcomeIds</code> | A vector of <code>cohort_definition_ids</code> used to define outcomes. |
| <code>outcomeNames</code> | A vector of names for each outcome. |
| <code>cohortDatabaseSchema</code> | The name of the database schema that is the location where the cohort data used to define the at risk cohort is available. If <code>cohortTable = DRUG_ERA</code> , <code>cohortDatabaseSchema</code> is not used by assumed to be <code>cdmSchema</code> . Requires read permissions to this database. |
| <code>cohortTable</code> | The tablename that contains the at risk cohort. If <code>cohortTable <> DRUG_ERA</code> , then expectation is <code>cohortTable</code> has format of COHORT table: <code>cohort_concept_id</code> , <code>SUBJECT_ID</code> , <code>COHORT_START_DATE</code> , <code>COHORT_END_DATE</code> . |
| <code>outcomeDatabaseSchema</code> | The name of the database schema that is the location where the data used to define the outcome cohorts is available. If <code>cohortTable = CONDITION_ERA</code> , <code>exposureDatabaseSchema</code> is not used by assumed to be <code>cdmSchema</code> . 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". |
| riskWindowStart | The start of the risk window (in days) relative to the startAnchor. |
| startAnchor | The anchor point for the start of the risk window. Can be "cohort start" or "cohort end". |
| riskWindowEnd | The end of the risk window (in days) relative to the endAnchor parameter |
| endAnchor | The anchor point for the end of the risk window. Can be "cohort start" or "cohort end". |
| outputFolder | Location to save results for shiny app |
| sampleSize | Sample from the target population |
| minCellCount | The minimum count that will be displayed |

Details

Users can define set of Ts, Os, databases and population settings. A list of data.frames containing details such as follow-up time distribution, time-to-event information, characterization details, time from last prior event, observation time distribution.

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.

| | |
|------------------|--|
| distribution | list for each O of a data.frame containing: i) Time to observation end distribution, ii) Time from observation start distribution, iii) Time to event distribution and iv) Time from last prior event to index distribution (only for patients in T who have O before index) |
| incident | list for each O of incidence of O in T during TAR |
| characterization | list for each O of Characterization of T, TnO, Tn~O |

Examples

```
## Not run:
##### EXAMPLE 1 #####

## End(Not run)
```

| | |
|---------------------|--|
| diagnosticOddsRatio | <i>Calculate the diagnostic odds ratio</i> |
|---------------------|--|

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

| | |
|-------------------------|-----------------------------------|
| drawAttritionDiagramPlp | <i>Draw the attrition diagram</i> |
|-------------------------|-----------------------------------|

Description

drawAttritionDiagramPlp draws the attrition diagram, showing how many people were excluded from the study population, and for what reasons.

Usage

```
drawAttritionDiagramPlp(
  attrition,
  targetLabel = "Target Population",
  outcomeLabel = "Outcome Count",
  fileName = NULL
)
```

Arguments

| | |
|--------------|---|
| attrition | The table of attrition details return from the population attr(popualtion, 'meta-Data')\$attrition |
| targetLabel | A label to us for the treated cohort. |
| outcomeLabel | A label to us for the comparator cohort. |
| 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. |

Value

A ggplot object. Use the [ggsave](#) function to save to file in a different format.

| | |
|---------------------|--|
| evaluateMultiplePlp | <i>externally validate the multiple plp models across new datasets</i> |
|---------------------|--|

Description

This function loads all the models in a multiple plp analysis folder and validates the models on new data

Usage

```
evaluateMultiplePlp(
  analysesLocation,
  outputLocation,
  connectionDetails,
  validationSchemaTarget,
  validationSchemaOutcome,
  validationSchemaCdm,
  databaseNames,
  validationTableTarget,
  validationTableOutcome,
  validationIdTarget = NULL,
  validationIdOutcome = NULL,
  oracleTempSchema = NULL,
  verbosity = "INFO",
  keepPrediction = F,
  sampleSize = NULL
)
```

Arguments

| | |
|------------------------|---|
| analysesLocation | The location where the multiple plp analyses are |
| outputLocation | The location to save to validation results |
| connectionDetails | The connection details for extracting the new data |
| validationSchemaTarget | A string or list of strings specifying the database containing the target cohorts |

| | |
|-------------------------|--|
| validationSchemaOutcome | A string or list of strings specifying the database containing the outcome cohorts |
| validationSchemaCdm | A string or list of strings specifying the database containing the cdm |
| databaseNames | A string or list of strings specifying sharing friendly database names corresponding to validationSchemaCdm |
| validationTableTarget | A string or list of strings specifying the table containing the target cohorts |
| validationTableOutcome | A string or list of strings specifying the table containing the outcome cohorts |
| validationIdTarget | An integer or list of integers specifying the cohort id for the target cohorts |
| validationIdOutcome | An integer or list of integers specifying the cohort id for the outcome cohorts |
| oracleTempSchema | The temp oracle schema requires read/write |
| 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: <ul style="list-style-type: none"> • DEBUG Highest verbosity showing all debug statements • TRACE Showing information about start and end of steps • INFO Show informative information (Default) • WARN Show warning messages • ERROR Show error messages • FATAL Be silent except for fatal errors |
| keepPrediction | Whether to keep the predictions for the new data |
| sampleSize | If not NULL, the number of people to sample from the target cohort |

Details

Users need to input a location where the results of the multiple plp analyses are found and the connection and database settings for the new data

| | |
|-------------|--------------------|
| evaluatePlp | <i>evaluatePlp</i> |
|-------------|--------------------|

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

| | |
|---------------------|--|
| externalValidatePlp | <i>externalValidatePlp - Validate a model on new databases</i> |
|---------------------|--|

Description

This function extracts data using a user specified connection and cdm_schema, applied the model and then calculates the performance

Usage

```
externalValidatePlp(
  plpResult,
  connectionDetails,
  validationSchemaTarget,
  validationSchemaOutcome,
  validationSchemaCdm,
  databaseNames,
  validationTableTarget = "cohort",
  validationTableOutcome = "cohort",
  validationIdTarget = NULL,
  validationIdOutcome = NULL,
  oracleTempSchema = NULL,
  verbosity = "INFO",
  keepPrediction = F,
  sampleSize = NULL,
  outputFolder
)
```

Arguments

| | |
|-------------------------|---|
| plpResult | The object returned by runPlp() containing the trained model |
| connectionDetails | The connection details for extracting the new data |
| validationSchemaTarget | A string or vector of strings specifying the database containing the target cohorts |
| validationSchemaOutcome | A string or vector of strings specifying the database containing the outcome cohorts |
| validationSchemaCdm | A string or vector of strings specifying the database containing the cdm |
| databaseNames | A string or vector of strings specifying sharing friendly database names corresponding to validationSchemaCdm |

| | |
|------------------------|--|
| validationTableTarget | A string or vector of strings specifying the table containing the target cohorts |
| validationTableOutcome | A string or vector of strings specifying the table containing the outcome cohorts |
| validationIdTarget | An iteger specifying the cohort id for the target cohort |
| validationIdOutcome | An iteger specifying the cohort id for the outcome cohort |
| oracleTempSchema | The temp oracle schema requires read/write |
| 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: <ul style="list-style-type: none"> • 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 |
| keepPrediction | Whether to keep the predicitions for the new data |
| sampleSize | If not NULL, the number of people to sample from the target cohort |
| outputFolder | If you want to save the results enter the directory to save here |

Details

Users need to input a trained model (the output of runPlp()) and new database connections. The function will return a list of length equal to the number of cdm_schemas input with the performance on the new data

Value

A list containing the performance for each validation_schema

| | |
|---------|------------------------------|
| f1Score | <i>Calculate the f1Score</i> |
|---------|------------------------------|

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 | <i>Calculate the falseDiscoveryRate</i> |
|--------------------|---|

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 | <i>Calculate the falseNegativeRate</i> |
|-------------------|--|

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

| | |
|--------------|-------------------------------|
| fitGLMMModel | <i>Fit a predictive model</i> |
|--------------|-------------------------------|

Description

Fit a predictive model

Usage

```
fitGLMMModel(
  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 grid search or user specified parameters

Usage

```
fitPlp(
  population,
  data,
  modelSettings,
  cohortId,
  outcomeId,
  minCovariateFraction = 0.001,
  normalizeData = T
)
```

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: <ul style="list-style-type: none"> • 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 |
| minCovariateFraction | The minimum fraction of the target population who have a variable for it to be included in the model training |
| normalizeData | Whether to normalise the data before model fitting |

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 specifying the model, preprocessing, outcomeId and cohortId |
| metaData | The model meta data |
| trainingTime | The time taken to train the classifier |

| | |
|-------------------|---|
| getAttritionTable | <i>Get the attrition table for a population</i> |
|-------------------|---|

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 | <i>Get a sparse summary of the calibration</i> |
|----------------|--|

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 stratified into equally sized bins of predicted probabilities.

Value

A dataframe with the calibration summary

| | |
|------------|--|
| getPlpData | <i>Get the patient level prediction data from the server</i> |
|------------|--|

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",
  firstExposureOnly = FALSE,
  washoutPeriod = 0,
  sampleSize = NULL,
  covariateSettings,
  excludeDrugsFromCovariates = FALSE
)
```

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 specify 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. 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 (-999 mean no outcome gets downloaded). |
| 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. Requires read permissions to this database. |
| cohortTable | The tablename that contains the at risk cohort. 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. Requires read permissions to this database. |
| outcomeTable | The tablename that contains the outcome cohorts. 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", "5" and "6". |
| 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 minimum 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. |
| sampleSize | If not NULL, only this number of people will be sampled from the target population (Default NULL) |
| covariateSettings | An object of type covariateSettings as created using the createCovariateSettings function in the FeatureExtraction package. |
| excludeDrugsFromCovariates | A redundant option |

Details

Based on the arguments, the at risk cohort data is retrieved, as well as outcomes occurring in these subjects. The at risk cohort is identified through user-defined cohorts in a cohort table either inside the CDM instance or in a separate schema. Similarly, outcomes are identified 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. If you wish to exclude concepts from covariates 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.

| | |
|--------------------------|--|
| <code>getPlpTable</code> | <i>Create a dataframe with the summary details of the population cohort for publications</i> |
|--------------------------|--|

Description

Create a dataframe with the summary details of the population cohort for publications

Usage

```
getPlpTable(
  cdmDatabaseSchema,
  oracleTempSchema,
  covariateSettings,
  longTermStartDays = -365,
  population,
  connectionDetails,
  cohortTable = "#temp_person"
)
```

Arguments

| | |
|--------------------------------|--|
| <code>cdmDatabaseSchema</code> | The schema containing the OMOP CDM data |
| <code>oracleTempSchema</code> | The oracle schema if needed |
| <code>covariateSettings</code> | The covariateSettings if different from default |
| <code>longTermStartDays</code> | How far to look back when looking for the variables in the data |
| <code>population</code> | The population you want the summary table for |
| <code>connectionDetails</code> | The connection details used to connect to the CDM database |
| <code>cohortTable</code> | The name of the temp table that will store the population cohort |

Details

This function is used to create a summary table for population to be inserted into publications

Examples

```
## Not run:
getTable1 (plpData, population, connectionDetails)

## End(Not run)
```

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 prediction 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 prediction object

Value

A data.frame with all the measures

| | |
|----------------------|------------------------------------|
| interpretInstallCode | <i>Tells you the package issue</i> |
|----------------------|------------------------------------|

Description

Tells you the package issue

Usage

```
interpretInstallCode(response)
```

Arguments

| | |
|----------|---|
| response | The response code from checkPlpInstallation() |
|----------|---|

Details

This function prints any issues found during the checkPlpInstallation() call

| | |
|---------------------------|--|
| launchDiagnosticsExplorer | <i>Launch the Diagnostics Explorer Shiny app</i> |
|---------------------------|--|

Description

Launch the Diagnostics Explorer Shiny app

Usage

```
launchDiagnosticsExplorer(dataFolder, launch.browser = FALSE)
```

Arguments

| | |
|----------------|---|
| dataFolder | A folder where the exported zip files with the results are stored. Zip files containing results from multiple databases can be placed in the same folder. |
| launch.browser | Should the app be launched in your default browser, or in a Shiny window. Note: copying to clipboard will not work in a Shiny window. |

Details

Launches a Shiny app that allows the user to explore the diagnostics

| | |
|------------|-----------------------|
| listAppend | <i>join two lists</i> |
|------------|-----------------------|

Description

join two lists

Usage

```
listAppend(a, b)
```

Arguments

| | |
|---|--------------|
| a | A list |
| b | Another list |

Details

This function joins two lists

| | |
|----------------------|---|
| loadEnsemblePlpModel | <i>loads the Ensmeble plp model and return a model list</i> |
|----------------------|---|

Description

loads the Ensmeble plp model and return a model list

Usage

```
loadEnsemblePlpModel(dirPath)
```

Arguments

| | |
|---------|---------------------------|
| dirPath | The location of the model |
|---------|---------------------------|

Details

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

| | |
|-----------------------|---------------------------------------|
| loadEnsemblePlpResult | <i>loads the Ensemble plp results</i> |
|-----------------------|---------------------------------------|

Description

loads the Ensemble plp results

Usage

```
loadEnsemblePlpResult(dirPath)
```

Arguments

| | |
|---------|---------------------------|
| dirPath | The location of the model |
|---------|---------------------------|

Details

Loads a plp model list that was saved using saveEnsemblePlpResults()

| | |
|-------------|---|
| loadPlpData | <i>Load the cohort data from a folder</i> |
|-------------|---|

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 | <i>loads the plp model</i> |
|--------------|----------------------------|

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 | <i>Loads the evalaution dataframe</i> |
|---------------|---------------------------------------|

Description

Loads the evalaution dataframe

Usage

```
loadPlpResult(dirPath)
```

Arguments

| | |
|---------|--|
| dirPath | The directory where the evaluation was saved |
|---------|--|

Details

Loads the evaluation

| | |
|----------------|---|
| loadPrediction | <i>Loads the predicton dataframe to csv</i> |
|----------------|---|

Description

Loads the predicton dataframe to csv

Usage

```
loadPrediction(fileLocation)
```

Arguments

fileLocation The location with the saved prediction

Details

Loads the predicton RDS file

| | |
|----------------------------|---|
| loadPredictionAnalysisList | <i>Load the multiple prediction json settings from a file</i> |
|----------------------------|---|

Description

Load the multiple prediction json settings from a file

Usage

```
loadPredictionAnalysisList(predictionAnalysisListFile)
```

Arguments

predictionAnalysisListFile
 The predicton specification json extracted from atlas.

Details

This function interprets a json with the multiple prediction settings and creates a list that can be combined with connection settings to run a multiple prediction study

Examples

```
## Not run:
predictionAnalysisList <- loadPredictionAnalysisList('./predictionStudyAnalyses.json')
predictionAnalysisList$connectionDetails = connectionDetails
predictionAnalysisList$cdmDatabaseSchema = cdmDatabaseSchema
predictionAnalysisList$cdmDatabaseName = cdmDatabaseName
predictionAnalysisList$oracleTempSchema = oracleTempSchema
predictionAnalysisList$cohortDatabaseSchema = cohortDatabaseSchema
predictionAnalysisList$cohortTable = cohortTable
predictionAnalysisList$outcomeDatabaseSchema = outcomeDatabaseSchema
predictionAnalysisList$outcomeTable = outcomeTable
predictionAnalysisList$cdmVersion = cdmVersion
predictionAnalysisList$outputFolder = outputFolder
result <- do.call(runPlpAnalyses, predictionAnalysisList)

## End(Not run)
```

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

outcomeSurvivalPlot *Plot the outcome incidence over time*

Description

Plot the outcome incidence over time

Usage

```
outcomeSurvivalPlot(
  plpData,
  outcomeId,
  removeSubjectsWithPriorOutcome = T,
  riskWindowStart = 1,
  riskWindowEnd = 3650,
  riskTable = T,
  confInt = T,
  yLabel = "Fraction of those who are outcome free in target population"
)
```

Arguments

| | |
|--------------------------------|--|
| plpData | The plpData object returned by running getPlpData() |
| outcomeId | The cohort id corresponding to the outcome |
| removeSubjectsWithPriorOutcome | Remove patients who have had the outcome before their target cohort index date from the plot |
| riskWindowStart | (integer) The time-at-risk starts at target cohort index date plus this value |
| riskWindowEnd | (integer) The time-at-risk ends at target cohort index date plus this value |
| riskTable | (binary) Whether to include a table at the bottom of the plot showing the number of people at risk over time |
| confInt | (binary) Whether to include a confidence interval |
| yLabel | (string) The label for the y-axis |

Details

This creates a survival plot that can be used to pick a suitable time-at-risk period

Value

TRUE if it ran

| | |
|------------------------|-------------------------------|
| PatientLevelPrediction | <i>PatientLevelPrediction</i> |
|------------------------|-------------------------------|

| | |
|----------------|---|
| personSplitter | <i>Split data into random subsets stratified by class</i> |
|----------------|---|

Description

Split data into random subsets stratified by class

Usage

```
personSplitter(population, test = 0.3, train = NULL, 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 |
| train | A real number between 0 and 1 indicating the train set fraction of the data. If not set train is equal to 1 - test |
| 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 validation 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 = "test", 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 = "test", 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.

| | |
|-------------------|--------------------------|
| plotLearningCurve | <i>plotLearningCurve</i> |
|-------------------|--------------------------|

Description

Create a plot of the learning curve using the object returned from `createLearningCurve`.

Usage

```
plotLearningCurve(
  learningCurve,
  metric = "AUROC",
  abscissa = "observations",
  plotTitle = "Learning Curve",
  plotSubtitle = NULL,
  fileName = NULL
)
```

Arguments

| | |
|----------------------------|--|
| <code>learningCurve</code> | An object returned by <code>createLearningCurve</code> function. |
| <code>metric</code> | Specifies the metric to be plotted: <ul style="list-style-type: none"> 'AUROC' - use the area under the Receiver Operating Characteristic curve 'AUPRC' - use the area under the Precision-Recall curve 'sBrier' - use the scaled Brier score |
| <code>abscissa</code> | Specify the abscissa metric to be plotted: <ul style="list-style-type: none"> 'observations' - use number of observations 'outcomes' - use number of positive outcomes |
| <code>plotTitle</code> | Title of the learning curve plot. |
| <code>plotSubtitle</code> | Subtitle of the learning curve plot. |
| <code>fileName</code> | Filename of plot to be saved, for example 'plot.png'. See the function <code>ggsave</code> in the <code>ggplot2</code> package for supported file formats. |

Value

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

Examples

```
## Not run:
# create learning curve object
learningCurve <- createLearningCurve(population,
                                     plpData,
                                     modelSettings)

# plot the learning curve
plotLearningCurve(learningCurve)

## End(Not run)
```

| | |
|---------|--|
| plotPlp | <i>Plot all the PatientLevelPrediction plots</i> |
|---------|--|

Description

Plot all the PatientLevelPrediction plots

Usage

```
plotPlp(result, filename, type = "test")
```

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. |
| type | Evaluation data type either 'test', 'val' or 'train' |

Details

Create a directory with all the plots

Value

TRUE if it ran

| | |
|---------------------|---|
| plotPrecisionRecall | <i>Plot the precision-recall curve using the sparse thresholdSummary data frame</i> |
|---------------------|---|

Description

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

Usage

```
plotPrecisionRecall(evaluation, type = "test", 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 | <i>Plot the Predicted probability density function, showing prediction overlap between true and false cases</i> |
|------------------|---|

Description

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

Usage

```
plotPredictedPDF(evaluation, type = "test", 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 | <i>Plot the side-by-side boxplots of prediction distribution, by class#'</i> |
|----------------------------|--|

Description

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

Usage

```
plotPredictionDistribution(evaluation, type = "test", 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 | <i>Plot the preference score probability density function, showing prediction overlap between true and false cases #'</i> |
|-------------------|---|

Description

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

Usage

```
plotPreferencePDF(evaluation, type = "test", 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 | <i>Plot the ROC curve</i> |
|---------|---------------------------|

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.

plotSmoothCalibration *Plot the smooth calibration as detailed in Calster et al. "A calibration heirarchy for risk models was defined: from utopia to empirical data" (2016)*

Description

Plot the smooth calibration as detailed in Calster et al. "A calibration heirarchy for risk models was defined: from utopia to empirical data" (2016)

Usage

```
plotSmoothCalibration(
  result,
  smooth = c("loess", "rcs"),
  span = 1,
  nKnots = 5,
  scatter = F,
  type = "test",
  bins = 20,
  zoom = c("none", "deciles", "data"),
  fileName = NULL
)
```

Arguments

| | |
|---------|--|
| result | The result of running runPlp function. 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. |
| smooth | options: 'loess' or 'rcs' |
| span | This specifies the width of span used for loess. This will allow for faster computing and lower memory usage. |
| nKnots | The number of knots to be used by the rcs evaluation. Default is 5 |
| scatter | plot the decile calibrations as points on the graph. Default is False |
| type | Whether to use train or test data, default is test. |

| | |
|----------|---|
| bins | The number of bins for the histogram. Default is 20. |
| zoom | Zoom in on the region containing the deciles or on the data. If not specified shows the entire space. |
| 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 smoothed calibration #'

Value

A cowplot object. Use the cowplot::save_plot function to save to file in a different format.

plotSparseCalibration *Plot the calibration*

Description

Plot the calibration

Usage

```
plotSparseCalibration(evaluation, type = "test", 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 = "test", 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 = "test", 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)
```

Format

A data frame containing the following elements:

covariatePrevalence prevalence of all covariates

outcomeModels regression model parameters to simulate outcomes

metaData settings used to simulate the profile

covariateRef covariateIds and covariateNames

timePrevalence time window

exclusionPrevalence prevalence of exclusion of covariates

| |
|--|
| positiveLikelihoodRatio |
| <i>Calculate the positiveLikelihoodRatio</i> |

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 |
| <i>Calculate the positivePredictiveValue</i> |

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

| | |
|------------------|--|
| predictAndromeda | <i>Generated predictions from a regression model</i> |
|------------------|--|

Description

Generated predictions from a regression model

Usage

```
predictAndromeda(
  coefficients,
  population,
  covariateData,
  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 |
| covariateData | An andromeda object containing the covariateData with predefined columns (see below). |
| modelType | Current supported types are "logistic", "poisson", "cox" 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 | <i>predictPlp</i> |
|------------|-------------------|

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, covariateData)
```

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 outcome, survival: hazard rate (per day) of the outcome.

| | |
|----------------|---|
| randomSplitter | <i>Split data into random subsets stratified by class</i> |
|----------------|---|

Description

Split data into random subsets stratified by class

Usage

```
randomSplitter(population, test = 0.3, train = NULL, 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 |
| train | A real number between 0 and 1 indicating the train set fraction of the data. If not set train is equal to 1 - test |
| 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 validation fold within the train set.

Value

A dataframe containing the columns: rowId and index

| | |
|-------------------------|--------------------------------|
| registerParallelBackend | <i>registerParallelBackend</i> |
|-------------------------|--------------------------------|

Description

Registers a parallel backend for multi core processing. The number of cores will be detected automatically, unless specified otherwise.

Usage

```
registerParallelBackend(cores = NULL, logical = TRUE)
```

Arguments

| | |
|---------|--|
| cores | the number of cores to use for multi core processing |
| logical | whether to consider logical or physical cores |

Examples

```
## Not run:
# detect logical cores automatically
registerParallelBackend()

# use four physical cores
numCores <- 4
registerParallelBackend(numCores, logical = FALSE)

## End(Not run)
```

| | |
|---------------------------|----------------------------------|
| registerSequentialBackend | |
| | <i>registerSequentialBackend</i> |

Description

registerSequentialBackend registers a sequential backend for single core processing.

Usage

```
registerSequentialBackend()
```

Examples

```
## Not run:
# register a sequential backend
registerSequentialBackend()

## End(Not run)
```

| | |
|------------------|---|
| runEnsembleModel | <i>ensemble - Create an ensembling model using different models</i> |
|------------------|---|

Description

```
#'
```

Usage

```
runEnsembleModel(
  population,
  dataList,
  modelList,
  testSplit = "time",
  testFraction = 0.2,
  splitSeed = NULL,
  nfold = 3,
  saveDirectory = NULL,
  saveEnsemble = F,
  savePlpData = F,
  savePlpResult = F,
  savePlpPlots = F,
  saveEvaluation = F,
  analysisId = NULL,
  verbosity = "INFO",
  ensembleStrategy = "mean"
)
```

Arguments

| | |
|----------------|--|
| population | The population created using createStudyPopulation() who will be used to develop the model |
| dataList | An list of object of type plpData - the patient level prediction data extracted from the CDM. |
| modelList | An list of type of base model created using one of the function in final ensemble model, the base model can be any model implemented in this package. |
| 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) |
| saveDirectory | The path to the directory where the results will be saved (if NULL uses working directory) |
| saveEnsemble | Binary indicating whether to save the ensemble |
| savePlpData | Binary indicating whether to save the plpData object (default is F) |
| savePlpResult | Binary indicating whether to save the object returned by runPlp (default is F) |
| savePlpPlots | Binary indicating whether to save the performance plots as pdf files (default is F) |
| saveEvaluation | Binary indicating whether to save the oerformance as csv files (default is T) |
| analysisId | The analysis ID |
| 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: <ul style="list-style-type: none"> • 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

ensembleStrategy

The strategy used for ensembling the outputs from different models, it can be 'mean', 'product', 'weighted' and 'stacked' 'mean' the average probability from different models 'product' the product rule 'weighted' the weighted average probability from different models using train AUC as weights. 'stacked' the stacked ensemble trains a logistics regression on different models.

Details

This function applied a list of models and combines them into an ensemble model

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 splitting but checks there are no overlaps in patients within training set and validation set - any overlaps are removed from validation set)

Usage

```
runPlp(
  population,
  plpData,
  minCovariateFraction = 0.001,
  normalizeData = T,
  modelSettings,
  testSplit = "stratified",
  testFraction = 0.25,
  trainFraction = NULL,
  splitSeed = NULL,
  nfold = 3,
  indexes = NULL,
  saveDirectory = NULL,
  savePlpData = T,
  savePlpResult = T,
  savePlpPlots = T,
  saveEvaluation = T,
  verbosity = "INFO",
  timeStamp = FALSE,
```



```

    analysisId = NULL,
    save = 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. |
| minCovariateFraction | The minimum fraction of target population who must have a covariate for it to be included in the model training |
| normalizeData | Whether to normalise the covariates before training (Default: TRUE) |
| modelSettings | An object of class modelSettings created using one of the function: <ul style="list-style-type: none"> • setLassoLogisticRegression() A lasso logistic regression model • setGradientBoostingMachine() A gradient boosting machine • setAdaBoost() An ada boost model • setRandomForest() A random forest model • setDecisionTree() A decision tree model • setCovNN() A convolutional neural network model • setCIReNN() A recurrent neural network model • setMLP() A neural network model • setDeepNN() A deep neural network model • setKNN() A KNN model |
| testSplit | Either 'stratified', 'subject' 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 'stratified' 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. 'subject' split is useful when a subject is in the data multiple times and you want all rows for the same subject in either the test or the train set but not in both. |
| testFraction | The fraction of the data to be used as the test set in the patient split evaluation. |
| trainFraction | A real number between 0 and 1 indicating the train set fraction of the data. If not set trainFraction is equal to 1 - test |
| 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) |
| saveDirectory | The path to the directory where the results will be saved (if NULL uses working directory) |
| savePlpData | Binary indicating whether to save the plpData object (default is T) |
| savePlpResult | Binary indicating whether to save the object returned by runPlp (default is T) |
| savePlpPlots | Binary indicating whether to save the performance plots as pdf files (default is T) |

| | |
|----------------|--|
| saveEvaluation | Binary indicating whether to save the oerformance as csv files (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: <ul style="list-style-type: none"> • 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. |
| save | Old input - please now use saveDirectory |

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, minCovariateFraction = 0.001,
                 modelSettings = model.lr ,
                 testSplit = 'time', testFraction=0.3,
                 nfold=3, indexes=NULL,
                 saveDirectory =file.path('C:', 'User', 'myPredictionName'),
                 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,
                 saveDirectory =file.path('C:', 'User', 'myPredictionName2'))

## End(Not run)

```

runPlpAnalyses

*Run a list of predictions***Description**

Run a list of predictions

Usage

```

runPlpAnalyses(
  connectionDetails,
  cdmDatabaseSchema,
  cdmDatabaseName,
  oracleTempSchema = cdmDatabaseSchema,
  cohortDatabaseSchema = cdmDatabaseSchema,

```

```

cohortTable = "cohort",
outcomeDatabaseSchema = cdmDatabaseSchema,
outcomeTable = "cohort",
cdmVersion = 5,
outputFolder = "../PlpOutput",
modelAnalysisList,
cohortIds,
cohortNames,
outcomeIds,
outcomeNames,
washoutPeriod = 0,
maxSampleSize = NULL,
minCovariateFraction = 0,
normalizeData = T,
testSplit = "person",
testFraction = 0.25,
splitSeed = NULL,
nfold = 3,
verbosity = "INFO"
)

```

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 specify both the database and the schema, so for example `'cdm_instance.dbo'`.

cdmDatabaseName

A string with a shareable name of the database (this will be shown to OHDSI researchers if the results get transported)

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 target cohorts are available. Requires read permissions to this database.

cohortTable

The tablename that contains the target cohorts. Expectation is cohortTable has format of COHORT table: `COHORT_DEFINITION_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. Requires read permissions to this database.

outcomeTable

The tablename that contains the outcome cohorts. 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".

outputFolder

Name of the folder where all the outputs will written to.

| | |
|----------------------|---|
| modelAnalysisList | A list of objects of type modelSettings as created using the createPlpModelSettings function. |
| cohortIds | A vector of cohortIds that specify all the target cohorts |
| cohortNames | A vector of cohortNames corresponding to the cohortIds |
| outcomeIds | A vector of outcomeIds that specify all the outcome cohorts |
| outcomeNames | A vector of outcomeNames corresponding to the outcomeIds |
| washoutPeriod | Minimum number of prior observation days |
| maxSampleSize | Max number of target people to sample from to develop models |
| minCovariateFraction | Any covariate with an incidence less than this value if ignored |
| normalizeData | Whether to normalize the covariates |
| testSplit | How to split into test/train (time or person) |
| testFraction | Fraction of data to use as test set |
| splitSeed | The seed used for the randomization into test/train |
| nfold | Number of folds used to do cross validation |
| verbosity | The logging level |

Details

Run a list of predictions for the target cohorts and outcomes of interest. This function will run all specified predictions, meaning that the total number of outcome models is $\text{length}(\text{cohortIds}) * \text{length}(\text{outcomeIds}) * \text{length}(\text{modelAnalysisList})$.

Value

A data frame with the following columns:

| | |
|------------------|---|
| analysisId | The unique identifier for a set of analysis choices. |
| cohortId | The ID of the target cohort populations. |
| outcomeId | The ID of the outcomeId. |
| plpDataFolder | The location where the plpData was saved |
| studyPopFile | The name of the file containing the study population |
| evaluationFolder | The name of file containing the evaluation saved as a csv |
| modelFolder | The name of the file containing the developed model. |

saveEnsemblePlpModel *saves the Ensemble plp model*

Description

saves the Ensemble plp model

Usage

```
saveEnsemblePlpModel(ensembleModel, dirPath)
```

Arguments

ensembleModel The ensemble model to save
 dirPath The location to save the model

Details

Saves a plp ensemble model

saveEnsemblePlpResult *saves the Ensemble plp results*

Description

saves the Ensemble plp results

Usage

```
saveEnsemblePlpResult(ensembleResult, dirPath)
```

Arguments

ensembleResult The ensemble result
 dirPath The location to save the ensemble results

Details

Saves a plp ensemble results

savePlpData *Save the cohort data to folder*

Description

savePlpData saves an object of type plpData to folder.

Usage

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

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
 overwrite Whether to force overwrite an existing file

Details

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

Examples

```
# todo
```

| | |
|--------------|----------------------------|
| savePlpModel | <i>Saves the plp model</i> |
|--------------|----------------------------|

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 specified folder

| | |
|---------------|---|
| savePlpResult | <i>Saves the result from runPlp into the location directory</i> |
|---------------|---|

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 | <i>Saves the prediction dataframe to RDS</i> |
|----------------|--|

Description

Saves the prediction dataframe to RDS

Usage

```
savePrediction(prediction, dirPath, fileName = "prediction.rds")
```

Arguments

| | |
|------------|--|
| prediction | The prediciton data.frame |
| dirPath | The directory to save the prediction RDS |
| fileName | The name of the RDS file that will be saved in dirPath |

Details

Saves the prediction data frame returned by predict.R to an RDS file and returns the fileLocation where the prediction is saved

| | |
|----------------------------|--|
| savePredictionAnalysisList | <i>Saves a json prediction settings given R settings</i> |
|----------------------------|--|

Description

Saves a json prediction settings given R settings

Usage

```
savePredictionAnalysisList(
  workFolder = "inst/settings",
  cohortIds,
  outcomeIds,
  cohortSettingCsv = file.path(workFolder, "CohortsToCreate.csv"),
  covariateSettingList,
  populationSettingList,
  modelSettingList,
  maxSampleSize = NULL,
  washoutPeriod = 0,
  minCovariateFraction = 0,
  normalizeData = T,
  testSplit = "person",
  testFraction = 0.25,
  splitSeed = 1,
  nfold = 3
)
```


Arguments

| | |
|-----------------------|---|
| workFolder | Location to save json specification |
| cohortIds | Vector of target population cohort ids |
| outcomeIds | Vector of outcome cohort ids |
| cohortSettingCsv | The location to the csv containing the cohort details |
| covariateSettingList | A list of covariate settings |
| populationSettingList | A list of population settings |
| modelSettingList | A list of model settings |
| maxSampleSize | If not NULL then max number of target population to sample for model training |
| washoutPeriod | Minimum prior observation for each person in target pop to be included |
| minCovariateFraction | Minimum covariate fraction to include |
| normalizeData | Whether to normalise data |
| testSplit | Split by person or time |
| testFraction | Fraction of data to use for test set |
| splitSeed | Seed used in test split |
| nfold | Number of folds used when training model |

Details

This function interprets a json with the multiple prediction settings and creates a list that can be combined with connection settings to run a multiple prediction study

| | |
|-------------|----------------------------------|
| sensitivity | <i>Calculate the sensitivity</i> |
|-------------|----------------------------------|

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

| | |
|-------------|--|
| setAdaBoost | <i>Create setting for AdaBoost with python</i> |
|-------------|--|

Description

Create setting for AdaBoost with python

Usage

```
setAdaBoost(nEstimators = 50, learningRate = 1, seed = NULL)
```

Arguments

| | |
|--------------|--|
| nEstimators | The maximum number of estimators at which boosting is terminated |
| learningRate | Learning rate shrinks the contribution of each classifier by learningRate. There is a trade-off between learningRate and nEstimators . |
| seed | A seed for the model |

Examples

```
## Not run:
model.adaBoost <- setAdaBoost(size = 4, alpha = 1e-05, seed = NULL)

## End(Not run)
```

| | |
|-----------|--|
| setCIReNN | <i>Create setting for CIReNN model</i> |
|-----------|--|

Description

Create setting for CIReNN model

Usage

```

setCIReNN(
  numberOfRNNLayr = c(1),
  units = c(128, 64),
  recurrentDropout = c(0.2),
  layerDropout = c(0.2),
  lr = c(1e-04),
  decay = c(1e-05),
  outcomeWeight = c(0),
  batchSize = c(100),
  epochs = c(100),
  earlyStoppingMinDelta = c(1e-04),
  earlyStoppingPatience = c(10),
  bayes = T,
  useDeepEnsemble = F,
  numberOfEnsembleNetwork = 5,
  useVae = T,
  vaeDataSamplingProportion = 0.1,
  vaeValidationSplit = 0.2,
  vaeBatchSize = 100L,
  vaeLatentDim = 10L,
  vaeIntermediateDim = 256L,
  vaeEpoch = 100L,
  vaeEpsilonStd = 1,
  useGPU = FALSE,
  maxGPUs = 2,
  seed = 1234
)

```

Arguments

| | |
|-----------------------|--|
| numberOfRNNLayr | The number of RNN layer, only 1, 2, or 3 layers available now. eg. 1, c(1,2), c(1,2,3) |
| units | The number of units of RNN layer - as a list of vectors |
| recurrentDropout | The reccurent dropout rate (regularisation) |
| layerDropout | The layer dropout rate (regularisation) |
| lr | Learning rate |
| decay | Learning rate decay over each update. |
| outcomeWeight | The weight of the outcome class in the loss function. Default is 0, which will be replaced by balanced weight. |
| batchSize | The number of data points to use per training batch |
| epochs | Number of times to iterate over dataset |
| earlyStoppingMinDelta | minimum change in the monitored quantity to qualify as an improvement for early stopping, i.e. an absolute change of less than min_delta in loss of validation data, will count as no improvement. |
| earlyStoppingPatience | Number of epochs with no improvement after which training will be stopped. |

| | |
|---------------------------|---|
| bayes | logical (either TRUE or FALSE) value for using Bayesian Drop Out Layer to measure uncertainty. If it is TRUE, both Epistemic and Aleatoric uncertainty will be measured through Bayesian Drop Out layer |
| useDeepEnsemble | logical (either TRUE or FALSE) value for using Deep Ensemble (Lakshminarayanan et al., 2017) to measure uncertainty. It cannot be used together with Bayesian deep learning. |
| numberOfEnsembleNetwork | Integer. Number of network used for Deep Ensemble (Lakshminarayanan et al recommended 5). |
| useVae | logical (either TRUE or FALSE) value for using Variational AutoEncoder before RNN |
| vaeDataSamplingProportion | Data sampling proportion for VAE |
| vaeValidationSplit | Validation split proportion for VAE |
| vaeBatchSize | batch size for VAE |
| vaeLatentDim | Number of latent dimesion for VAE |
| vaeIntermediateDim | Number of intermediate dimesion for VAE |
| vaeEpoch | Number of times to interate over dataset for VAE |
| vaeEpsilonStd | Epsilon |
| useGPU | logical (either TRUE or FALSE) value. If you have GPUs in your machine, and want to use multiple GPU for deep learning, set this value as TRUE |
| maxGPUs | Integer, If you will use GPU, how many GPUs will be used for deep learning in VAE? GPU parallelisation for deep learning will be activated only when parallel vae is true. Integer >= 2 or list of integers, number of GPUs or list of GPU IDs on which to create model replicas. |
| seed | Random seed used by deep learning model |

Examples

```
## Not run:
model.CIReNN <- setCIReNN()

## End(Not run)
```

setCNNTorch

Create setting for CNN model with python

Description

Create setting for CNN model with python

Usage

```
setCNTorch(
  nbfilters = c(16, 32),
  epochs = c(20, 50),
  seed = 0,
  class_weight = 0,
  type = "CNN"
)
```

Arguments

| | |
|--------------|---|
| nbfilters | The number of filters |
| epochs | The number of epochs |
| seed | A seed for the model |
| class_weight | The class weight used for imbalanced data: 0: Inverse ratio between positives and negatives -1: Focal loss |
| type | It can be normal 'CNN', 'CNN_LSTM', 'CNN_MLF' with multiple kernels with different kernel size, 'CNN_MIX', 'ResNet' and 'CNN_MULTI' |

Examples

```
## Not run:
model.cnnTorch <- setCNTorch()

## End(Not run)
```

setCovNN

Create setting for multi-resolution CovNN model (stucture based on <https://arxiv.org/pdf/1608.00647.pdf> CNN1)

Description

Create setting for multi-resolution CovNN model (stucture based on <https://arxiv.org/pdf/1608.00647.pdf> CNN1)

Usage

```
setCovNN(
  batchSize = 1000,
  outcomeWeight = 1,
  lr = 1e-05,
  decay = 1e-06,
  dropout = 0,
  epochs = 10,
  filters = 3,
  kernelSize = 10,
  loss = "binary_crossentropy",
  seed = NULL
)
```

Arguments

| | |
|---------------|--|
| batchSize | The number of samples to used in each batch during model training |
| outcomeWeight | The weight assined to the outcome (make greater than 1 to reduce unballanced label issue) |
| lr | The learning rate |
| decay | The decay of the learning rate |
| dropout | [currently not used] the dropout rate for regularisation |
| epochs | The number of times data is used to train the model (e.g., epoches=1 means data only used once to train) |
| filters | The number of columns output by each convolution |
| kernelSize | The number of time dimensions used for each convolution |
| loss | The loss function implemented |
| seed | The random seed |

Examples

```
## Not run:
model.CovNN <- setCovNN()

## End(Not run)
```

| | |
|-----------|--|
| setCovNN2 | <i>Create setting for CovNN2 model - convolution across input and time - https://arxiv.org/pdf/1608.00647.pdf</i> |
|-----------|--|

Description

Create setting for CovNN2 model - convolution across input and time - <https://arxiv.org/pdf/1608.00647.pdf>

Usage

```
setCovNN2(
  batchSize = 1000,
  outcomeWeight = 1,
  lr = 1e-05,
  decay = 1e-06,
  dropout = 0,
  epochs = 10,
  filters = 3,
  kernelSize = 10,
  loss = "binary_crossentropy",
  seed = NULL
)
```

Arguments

| | |
|---------------|--|
| batchSize | The number of samples to used in each batch during model training |
| outcomeWeight | The weight assined to the outcome (make greater than 1 to reduce unballanced label issue) |
| lr | The learning rate |
| decay | The decay of the learning rate |
| dropout | [currently not used] the dropout rate for regularisation |
| epochs | The number of times data is used to train the model (e.g., epoches=1 means data only used once to train) |
| filters | The number of columns output by each convolution |
| kernelSize | The number of time dimensions used for each convolution |
| loss | The loss function implemented |
| seed | The random seed |

Examples

```
## Not run:
model.CovNN <- setCovNN()

## End(Not run)
```

setCoxModel

Create setting for lasso Cox model

Description

Create setting for lasso Cox model

Usage

```
setCoxModel(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 <- setCoxModel()
```

| | |
|-----------------|--|
| setDecisionTree | <i>Create setting for DecisionTree with python</i> |
|-----------------|--|

Description

Create setting for DecisionTree with python

Usage

```
setDecisionTree(
  maxDepth = 10,
  minSamplesSplit = 2,
  minSamplesLeaf = 10,
  minImpurityDecrease = 10^-7,
  seed = NULL,
  classWeight = "None",
  plot = F
)
```

Arguments

| | |
|---------------------|--|
| maxDepth | The maximum depth of the tree |
| minSamplesSplit | The minimum samples per split |
| minSamplesLeaf | The minimum number of samples per leaf |
| minImpurityDecrease | Threshold for early stopping in tree growth. A node will split if its impurity is above the threshold, otherwise it is a leaf. |
| seed | The random state seed |
| classWeight | Balance or None |
| plot | Boolean whether to plot the tree (requires python pydotplus module) |

Examples

```
## Not run:
model.decisionTree <- setDecisionTree(maxDepth=10,minSamplesLeaf=10, seed=NULL )

## End(Not run)
```

| | |
|-----------|--|
| setDeepNN | <i>Create setting for DeepNN model</i> |
|-----------|--|

Description

Create setting for DeepNN model

Usage

```
setDeepNN(
  units = list(c(128, 64), 128),
  layer_dropout = c(0.2),
  lr = c(1e-04),
  decay = c(1e-05),
  outcome_weight = c(1),
  batch_size = c(100),
  epochs = c(100),
  seed = NULL
)
```

Arguments

| | |
|----------------|--|
| units | The number of units of the deep network - as a list of vectors |
| layer_dropout | The layer dropout rate (regularisation) |
| lr | Learning rate |
| decay | Learning rate decay over each update. |
| outcome_weight | The weight of the outcome class in the loss function |
| batch_size | The number of data points to use per training batch |
| epochs | Number of times to iterate over dataset |
| seed | Random seed used by deep learning model |

Examples

```
## Not run:
model <- setDeepNN()

## End(Not run)
```

| | |
|----------------|--|
| setGBMSurvival | <i>Create setting for GBM Survival with python #' @description This creates a setting for fitting GBM survival model. You need sksurv python install. To install this open your command line and type: conda install -c sebp scikit-survival</i> |
|----------------|--|

Description

Create setting for GBM Survival with python #' @description This creates a setting for fitting GBM survival model. You need sksurv python install. To install this open your command line and type: conda install -c sebp scikit-survival

Usage

```
setGBMSurvival(
  loss = "coxph",
  learningRate = 0.1,
  nEstimators = c(100),
  criterion = "friedman_mse",
```

```

minSamplesSplit = 2,
minSamplesLeaf = 1,
minWeightFractionLeaf = 0,
maxDepth = c(3, 10, 17),
minImpuritySplit = NULL,
minImpurityDecrease = 0,
maxFeatures = NULL,
maxLeafNodes = NULL,
presort = NULL,
subsample = 1,
dropoutRate = 0,
seed = NULL,
quiet = F
)

```

Arguments

| | |
|-----------------------|---|
| loss | A string specifying the loss function to minimise (default: 'coxph') |
| learningRate | A double specifying the learning rate (controls convergence speed) |
| nEstimators | An integer specifying how many trees to build |
| criterion | Default: 'friedman_mse' |
| minSamplesSplit | An integer specifying min samples per tree split (complexity) |
| minSamplesLeaf | An integer specifying min samples per leaf (complexity) |
| minWeightFractionLeaf | Lookup |
| maxDepth | An integer specifying the max depth of trees (complexity) |
| minImpuritySplit | A double or NULL specifying the minimum impurity split |
| minImpurityDecrease | will add |
| maxFeatures | will add |
| maxLeafNodes | will add |
| presort | will add |
| subsample | will add |
| dropoutRate | will add |
| seed | will add |
| quiet | will add |

Details

Pick the hyper-parameters you want to do a grid search for

Examples

```

## Not run:
gbmSurv <- setGBMSurvival(learningRate=c(0.1,0.01), nEstimators =c(10,50,100),
  maxDepth=c(4,10,17), seed = 2)

## End(Not run)

```

```
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(100, 1000),
  nthread = 20,
  earlyStopRound = 25,
  maxDepth = c(4, 6, 17),
  minRows = 2,
  learnRate = c(0.005, 0.01, 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?) |
| earlyStopRound | If the performance does not increase over earlyStopRound number of interactions then training stops (this prevents overfitting) |
| maxDepth | Maximum number of interactions - a large value will lead to slow model training |
| minRows | The minimum number of rows required at each end node of the tree |
| learnRate | 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,
                                         maxDepth=c(4,6), learnRate=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"), threads = 1)
```

Arguments

| | |
|-------------|---|
| k | The number of neighbors to consider |
| indexFolder | The directory where the results and intermediate steps are output |
| threads | The number of threads to use when applying big knn |

Examples

```
## Not run:
model.knn <- setKNN(k=10000)

## End(Not run)
```

```
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()
```

```
setLRTorch
```

Create setting for logistics regression model with python

Description

Create setting for logistics regression model with python

Usage

```
setLRTorch(
  w_decay = c(5e-04, 0.005),
  epochs = c(20, 50, 100),
  seed = NULL,
  class_weight = 0,
  autoencoder = FALSE,
  vae = FALSE
)
```

Arguments

| | |
|--------------|--|
| w_decay | The l2 regularisation |
| epochs | The number of epochs |
| seed | A seed for the model |
| class_weight | The class weight used for imbalanced data: 0: Inverse ratio between positives and negatives -1: Focal loss |
| autoencoder | First learn stacked autoencoder for input features, then train LR on the encoded features. |
| vae | First learn stacked variational autoencoder for input features, then train LR on the encoded features. |

Examples

```
## Not run:
model.lrTorch <- setLRTorch()

## End(Not run)
```

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)
```

setMLPTorch

*Create setting for neural network model with python***Description**

Create setting for neural network model with python

Usage

```
setMLPTorch(
  size = c(500, 1000),
  w_decay = c(5e-04, 0.005),
  epochs = c(20, 50),
  seed = 0,
  class_weight = 0,
  mlp_type = "MLP",
  autoencoder = FALSE,
  vae = FALSE
)
```

Arguments

| | |
|--------------|--|
| size | The number of hidden nodes |
| w_decay | The l2 regularisation |
| epochs | The number of epochs |
| seed | A seed for the model |
| class_weight | The class weight used for imbalanced data: 0: Inverse ratio between positives and negatives -1: Focal loss |
| mlp_type | The type of multiple layer network, including MLP and SNN (self-normalizing neural network) |
| autoencoder | First learn stacked autoencoder for input features, then train MLP on the encoded features. |
| vae | First learn stacked variational autoencoder for input features, then train MLP on the encoded features. |

Examples

```
## Not run:
model.mlpTorch <- setMLPTorch()

## End(Not run)
```

| | |
|---------------|---|
| setNaiveBayes | <i>Create setting for naive bayes model with python</i> |
|---------------|---|

Description

Create setting for naive bayes model with python

Usage

```
setNaiveBayes(variableNumber = 2000)
```

Arguments

`variableNumber` The number of variables selected by feature selection prior to training the model (this is required due to Naive Bayes requiring a non sparse matrix)

Examples

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

## End(Not run)
```

| | |
|----------------------|--|
| setPythonEnvironment | <i>Use the virtual environment created using configurePython()</i> |
|----------------------|--|

Description

Use the virtual environment created using configurePython()

Usage

```
setPythonEnvironment(envname = "PLP", envtype = NULL)
```

Arguments

`envname` A string for the name of the virtual environment (default is 'PLP')

`envtype` An option for specifying the environment as 'conda' or 'python'. If NULL then the default is 'conda' for windows users and 'python' for non-windows users

Details

This function sets PatientLevelPrediction to use a virtual environment

| | |
|-----------------|--|
| 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 = 500,
  maxDepth = c(4, 10, 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 |
| maxDepth | 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),
                           maxDepth=c(5,20))

## End(Not run)
```

setRandomForestQuantileRegressor

Create setting for RandomForestQuantileRegressor with python scikit-garden (skgarden.quantile.RandomForestQuantileRegressor) #' @description This creates a setting for fitting a RandomForestQuantileRegressor model. You need skgarden python install. To install this open your command line and type: conda install -c conda-forge scikit-garden

Description

Create setting for RandomForestQuantileRegressor with python scikit-garden (skgarden.quantile.RandomForestQuantileRegressor) #' @description This creates a setting for fitting a RandomForestQuantileRegressor model. You need skgarden python install. To install this open your command line and type: conda install -c conda-forge scikit-garden

Usage

```
setRandomForestQuantileRegressor(
  nEstimators = c(100),
  criterion = "mse",
  maxFeatures = -1,
  maxDepth = 4,
  minSamplesSplit = 2,
  minSamplesLeaf = 1,
  minWeightFractionLeaf = 0,
  maxLeafNodes = NULL,
  bootstrap = TRUE,
  oobScore = FALSE,
  warmStart = FALSE,
  seed = NULL,
  quiet = F
)
```

Arguments

| | |
|-----------------------|---|
| nEstimators | (int default:100) The number of trees in the forest. |
| criterion | (string default="mse") The function to measure the quality of a split. Supported criteria are "mse" for the mean squared error, which is equal to variance reduction as feature selection criterion, and "mae" for the mean absolute error. |
| maxFeatures | (int default: -1) The number of features to consider when looking for the best split. If -1 then use sqrt of total number of features. |
| maxDepth | (int default:4) The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than minSamplesSplit samples. |
| minSamplesSplit | An integer specifying min samples per tree split (complexity) |
| minSamplesLeaf | An integer specifying min samples per leaf (complexity) |
| minWeightFractionLeaf | Lookup |
| maxLeafNodes | (int) Grow trees with maxLeafNodes in best-first fashion. Best nodes are defined as relative reduction in impurity. If None then unlimited number of leaf nodes. |
| bootstrap | (boolean default:TRUE) Whether bootstrap samples are used when building trees. |
| oobScore | (boolean default:FALSE) Whether to use out-of-bag samples to estimate the R^2 on unseen data. |
| warmStart | (boolean default:FALSE) When set to True, reuse the solution of the previous call to fit and add more estimators to the ensemble, otherwise, just fit a whole new forest. |
| seed | will add |
| quiet | will add |

Details

Pick the hyper-parameters you want to do a grid search for

Examples

```
## Not run:
rfQR <- setRandomForestQuantileRegressor(nEstimators =c(10,50,100),
  maxDepth=c(4,10,17), seed = 2)

## End(Not run)
```

setRNNTorch

Create setting for RNN model with python

Description

Create setting for RNN model with python

Usage

```
setRNNTorch(
  hidden_size = c(50, 100),
  epochs = c(20, 50),
  seed = 0,
  class_weight = 0,
  type = "RNN"
)
```

Arguments

| | |
|--------------|--|
| hidden_size | The hidden size |
| epochs | The number of epochs |
| seed | A seed for the model |
| class_weight | The class weight used for imbalanced data: 0: Inverse ratio between positives and negatives -1: Focal loss |
| type | It can be normal 'RNN', 'BiRNN' (bidirectional RNN) and 'GRU' |

Examples

```
## Not run:
model.rnnTorch <- setRNNTorch()

## End(Not run)
```

| | |
|--------------------|------------------------------------|
| setSagemakerBinary | Create setting for sagemaker model |
|--------------------|------------------------------------|

Description

Create setting for sagemaker model

Usage

```
setSagemakerBinary(
  classifier = "xgboost",
  bucket,
  prefix = "data",
  roleArn,
  otherparams = NULL,
  seed = NULL
)
```

Arguments

| | |
|-------------|--|
| classifier | The name of the sagemaker binary classifier to use (pick from: knn, xgboost or linear-learner) |
| bucket | The s3 bucket string to save data for model training |
| prefix | The s3 subdirectory for the data |
| roleArn | The amazon roleArn |
| otherparams | Other parameters for training (currently not working) |
| seed | The seed for the training |

Examples

```
## Not run:
model.sm <- setSagemakerBinary(classifier='xgboost', bucket='ohdsi3')

## End(Not run)
```

| | |
|----------------|---|
| similarPlpData | Extract new plpData using plpModel settings use metadata in plp-Model 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,
  newCdmDatabaseSchema = NULL,
  newCohortDatabaseSchema = NULL,
  newCohortTable = NULL,
  newCohortId = NULL,
  newOutcomeDatabaseSchema = NULL,
  newOutcomeTable = NULL,
  newOutcomeId = NULL,
  newOracleTempSchema = newCdmDatabaseSchema,
  sample = NULL,
  createPopulation = T,
  createCohorts = T
)
```

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 |
| newOracleTempSchema | The temp coracle schema |
| sample | The number of people to sample (default is NULL meaning use all data) |
| createPopulation | Whether to create the study population as well |
| createCohorts | No longer used |

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 | <i>Generate simulated data</i> |
|-----------------|--------------------------------|

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 | <i>Calculate the specificity</i> |
|-------------|----------------------------------|

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

| | |
|-----------------|--|
| subjectSplitter | <i>Split data when patients are in the data multiple times such that the same patient is always either in the train set or the test set (the same patient cannot be in both the test and train set at different times)</i> |
|-----------------|--|

Description

Split data when patients are in the data multiple times such that the same patient is always either in the train set or the test set (the same patient cannot be in both the test and train set at different times)

Usage

```
subjectSplitter(population, test = 0.3, train = NULL, 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 |
| train | A real number between 0 and 1 indicating the train set fraction of the data. If not set train is equal to 1 - test |
| 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 validation fold within the train set.

Value

A dataframe containing the columns: rowId and index

| | |
|--------------|---|
| timeSplitter | <i>Split test/train data by time and then partitions training set into random folds stratified by class</i> |
|--------------|---|

Description

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

Usage

```
timeSplitter(population, test = 0.3, train = NULL, 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 |
| train | A real number between 0 and 1 indicating the training 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 validation 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, temporal = F)
```

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) |
| temporal | Whether you want to convert temporal data |

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
```

| | |
|---------------------|---|
| toSparseTorchPython | <i>Convert the plpData in COO format into a sparse python matrix using torch.sparse</i> |
|---------------------|---|

Description

Converts the standard plpData to a sparse matrix directly into python

Usage

```
toSparseTorchPython(
    plpData,
    population,
    map = NULL,
    temporal = F,
    pythonExePath = 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) |
| temporal | Whether to include timeId into tensor |
| pythonExePath | Location of python exe you want to use |

Details

This function converts the covariate file from ffd 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
```

| | |
|------------------|--|
| transferLearning | <i>[Under development] Transfer learning</i> |
|------------------|--|

Description

[Under development] Transfer learning

Usage

```
transferLearning(
  plpResult,
  plpData,
  population,
  fixLayers = T,
  includeTop = F,
  addLayers = c(100, 10),
  layerDropout = c(T, T),
  layerActivation = c("relu", "softmax"),
  outcomeWeight = 1,
  batchSize = 10000,
  epochs = 20
)
```

Arguments

| | |
|-----------------|--|
| plpResult | The plp result when training a kersa deep learning model on big data |
| plpData | The new data to fine tune the model on |
| population | The population for the new data |
| fixLayers | boolean specifying whether to fix weights in model being transferred |
| includeTop | If TRUE the final layer of the model being transferred is removed |
| addLayers | vector specifying nodes in each layer to add e.g. c(100,10) will add another layer with 100 nodes and then a final layer with 10 |
| layerDropout | Add dropout to each new layer (binary vector length of addLayers) |
| layerActivation | Activation function for each new layer (string vector length of addLayers) |
| outcomeWeight | The weight to assign the class 1 when training the model |
| batchSize | Size of each batch for updating layers |
| epochs | Number of epoches to run |

Examples

```
## Not run:
modelSet <- setDeepNN()
plpResult <- runPlp(plpData, population, modelSettings = modelSet, ...)

transferLearning(...)

## End(Not run)
```

| | |
|----------------|---|
| transportModel | <i>Transports a plpModel to a new location and removes sensitive data</i> |
|----------------|---|

Description

Transports a plpModel to a new location and removes sensitive data

Usage

```
transportModel(plpModel, outputFolder)
```

Arguments

| | |
|--------------|---|
| plpModel | A trianed model. |
| 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

This function is used to

Examples

```
## Not run:
transportModel(plpModel, "s:/temp/exportTest")

## End(Not run)
```

| | |
|--------------|--|
| transportPlp | <i>Transports a plpResult to a new location and removed sensitive data</i> |
|--------------|--|

Description

Transports a plpResult to a new location and removed sensitive data

Usage

```
transportPlp(
  plpResult,
  modelName = NULL,
  dataName = NULL,
  outputFolder,
  n = NULL,
  includeEvaluationStatistics = T,
  includeThresholdSummary = T,
  includeDemographicSummary = T,
  includeCalibrationSummary = T,
  includePredictionDistribution = T,
  includeCovariateSummary = T,
  save = T
)
```

Arguments

| | |
|-------------------------------|---|
| plpResult | An object returned by running runPlp. |
| modelName | A string of the name of the model |
| dataName | A string of the name of the data |
| 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. |
| n | The minimum number of people required for each result summary to be included |
| includeEvaluationStatistics | Whether to include the evaluationStatistics |
| includeThresholdSummary | Whether to include the thresholdSummary |
| includeDemographicSummary | Whether to include the demographicSummary |
| includeCalibrationSummary | Whether to include the calibrationSummary |
| includePredictionDistribution | Whether to include the predictionDistribution |
| includeCovariateSummary | Whether to include the covariateSummary |
| save | Whether to save the result or just return the transportable object |

Details

This function is used to

Examples

```
## Not run:
transportPlp(plpResult, "s:/temp/exportTest", n=10)

## End(Not run)
```

| | |
|-----------------|---|
| viewMultiplePlp | <i>open a local shiny app for viewing the result of a multiple PLP analyses</i> |
|-----------------|---|

Description

open a local shiny app for viewing the result of a multiple PLP analyses

Usage

```
viewMultiplePlp(analysesLocation)
```

Arguments

| | |
|------------------|--|
| analysesLocation | The directory containing the results (with the analysis_x folders) |
|------------------|--|

Details

Opens a shiny app for viewing the results of the models from various T,O, Tar and settings settings.

viewPlp*viewPlp - Interactively view the performance and model settings*

Description

This is a shiny app for viewing interactive plots of the performance and the settings

Usage

```
viewPlp(runPlp, validatePlp = NULL)
```

Arguments

runPlp The output of runPlp() (an object of class 'runPlp')

validatePlp The output of externalValidatePlp (on object of class 'validatePlp')

Details

Either the result of runPlp and view the plots

Value

Opens a shiny app for interactively viewing the results

Index

*Topic **datasets**

plpDataSimulationProfile, 57

accuracy, 5

applyEnsembleModel, 6

applyModel, 7

averagePrecision, 8

brierScore, 8

calibrationLine, 9

checkPlpInstallation, 9

combinePlpModelSettings, 10

computeAuc, 10

computeAucFromDataFrames, 11

configurePython, 11

createControl, 33

createLearningCurve, 12, 50

createLearningCurvePar, 14

createLrSql, 16

createPlpJournalDocument, 17

createPlpModelSettings, 18, 69

createPlpReport, 19

createPrior, 33

createStudyPopulation, 20

createStudyPopulationSettings, 22

diagnostic, 23

diagnosticOddsRatio, 26

drawAttritionDiagramPlp, 26

evaluateMultiplePlp, 27

evaluatePlp, 28

externalValidatePlp, 29

f1Score, 30

falseDiscoveryRate, 31

falseNegativeRate, 31

falseOmissionRate, 32

falsePositiveRate, 32

fitGLMModel, 33

fitPlp, 34, 60

getAttritionTable, 35

getCalibration, 35

getPlpData, 36

getPlpTable, 38

getPredictionDistribution, 39

getThresholdSummary, 39

ggsave, 27, 48–57

interpretInstallCode, 40

launchDiagnosticsExplorer, 40

listAppend, 41

loadEnsemblePlpModel, 41

loadEnsemblePlpResult, 42

loadPlpData, 42

loadPlpModel, 43

loadPlpResult, 43

loadPrediction, 44

loadPredictionAnalysisList, 44

negativeLikelihoodRatio, 45

negativePredictiveValue, 46

outcomeSurvivalPlot, 46

PatientLevelPrediction, 47

personSplitter, 47

plotDemographicSummary, 48

plotF1Measure, 48

plotGeneralizability, 49

plotLearningCurve, 50

plotPlp, 51

plotPrecisionRecall, 51

plotPredictedPDF, 52

plotPredictionDistribution, 52

plotPreferencePDF, 53

plotRoc, 53

plotSmoothCalibration, 54

plotSparseCalibration, 55

plotSparseCalibration2, 56

plotSparseRoc, 56

plotVariableScatterplot, 57

plpDataSimulationProfile, 57

positiveLikelihoodRatio, 58

positivePredictiveValue, 58

predict, 10, 35

predictAndromeda, 59

predictPlp, 59
predictProbabilities, 8, 9, 39, 54, 60

randomSplitter, 61
registerParallelBackend, 61
registerSequentialBackend, 62
runEnsembleModel, 62
runPlp, 48, 49, 51–57, 64
runPlpAnalyses, 67

saveEnsemblePlpModel, 69
saveEnsemblePlpResult, 70
savePlpData, 70
savePlpModel, 71
savePlpResult, 71
savePrediction, 72
savePredictionAnalysisList, 72
sensitivity, 73
setAdaBoost, 74
setCIReNN, 74
setCNTorch, 76
setCovNN, 77
setCovNN2, 78
setCoxModel, 79
setDecisionTree, 80
setDeepNN, 80
setGBMSurvival, 81
setGradientBoostingMachine, 83
setKNN, 83
setLassoLogisticRegression, 84
setLRTorch, 84
setMLP, 85
setMLPTorch, 86
setNaiveBayes, 87
setPythonEnvironment, 87
setRandomForest, 88
setRandomForestQuantileRegressor, 88
setRNNTorch, 90
setSagemakerBinary, 91
similarPlpData, 91
simulatePlpData, 93
specificity, 94
subjectSplitter, 94

timeSplitter, 95
toSparseM, 96
toSparseTorchPython, 97
transferLearning, 98
transportModel, 99
transportPlp, 99

viewMultiplePlp, 100
viewPlp, 101