

# Package ‘PatientLevelPrediction’

October 7, 2018

**Type** Package

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

**Version** 3.0.0

**Date** 2018-10-07

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

**Maintainer** Jenna Reps <reps@ohdsi.org>

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

**License** Apache License 2.0

**Depends** R (>= 3.3.0),  
DatabaseConnector (>= 1.11.4),  
FeatureExtraction (>= 2.0.0),  
Cyclops (>= 1.2.2)

**Imports** ggplot2,  
gridExtra,  
PRROC,  
magrittr,  
foreach,  
doParallel,  
dplyr,  
bit,  
ff,  
ffbase (>= 0.12.1),  
plyr,  
survAUC,  
Rcpp (>= 0.11.2),  
SqlRender (>= 1.1.3),  
survival,  
xgboost,

Matrix,  
 AUC,  
 PythonInR,  
 utils,  
 methods,  
 BigKnn,  
 reshape2,  
 officer,  
 diagram,  
 shiny,  
 plotly,  
 DT,  
 htmlwidgets (> 0.8),  
 tidyr,  
 viridisLite,  
 RCurl,  
 RJSONIO,  
 keras,  
 slam,  
 magrittr,  
 OhdsiSharing,  
 OhdsiRTools

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

**LinkingTo** Rcpp

**NeedsCompilation** yes

**RoxygenNote** 6.0.1

## R topics documented:

accuracy . . . . .	5
applyEnsembleModel . . . . .	5
applyModel . . . . .	6
averagePrecision . . . . .	7
brierScore . . . . .	8
bySumFf . . . . .	8
calibrationLine . . . . .	9
checkffFolder . . . . .	9
checkPlpInstallation . . . . .	10
clearffTempDir . . . . .	10
combinePlpModelSettings . . . . .	11
computeAuc . . . . .	11
computeAucFromDataFrames . . . . .	12

createCohort . . . . .	12
createExistingModelSql . . . . .	13
createLearningCurve . . . . .	14
createLearningCurvePar . . . . .	16
createLrSql . . . . .	17
createPlpJournalDocument . . . . .	18
createPlpModelSettings . . . . .	19
createPlpReport . . . . .	19
createStudyPopulation . . . . .	20
createStudyPopulationSettings . . . . .	22
diagnosticOddsRatio . . . . .	23
drawAttritionDiagramPlp . . . . .	24
evaluateExistingModel . . . . .	24
evaluateMultiplePlp . . . . .	26
evaluatePlp . . . . .	27
exportPlpDataToCsv . . . . .	28
exportPlpResult . . . . .	29
externalValidatePlp . . . . .	30
f1Score . . . . .	31
falseDiscoveryRate . . . . .	31
falseNegativeRate . . . . .	32
falseOmissionRate . . . . .	33
falsePositiveRate . . . . .	33
fitGLMModel . . . . .	34
fitPlp . . . . .	34
getAttritionTable . . . . .	35
getCalibration . . . . .	36
getCovariateData . . . . .	36
getModelDetails . . . . .	37
getPlpData . . . . .	38
getPlpTable . . . . .	40
getPredictionDistribution . . . . .	41
getThresholdSummary . . . . .	41
grepCovariateNames . . . . .	42
insertDbPopulation . . . . .	42
interpretInstallCode . . . . .	43
loadEnsemblePlpModel . . . . .	44
loadEnsemblePlpResult . . . . .	44
loadPlpData . . . . .	45
loadPlpModel . . . . .	45
loadPlpResult . . . . .	46
loadPrediction . . . . .	46
loadPredictionAnalysisList . . . . .	47
negativeLikelihoodRatio . . . . .	47
negativePredictiveValue . . . . .	48
packageResults . . . . .	49
PatientLevelPrediction . . . . .	49
personSplitter . . . . .	50
plotDemographicSummary . . . . .	50
plotF1Measure . . . . .	51
plotGeneralizability . . . . .	51
plotLearningCurve . . . . .	52

plotPlp . . . . .	53
plotPrecisionRecall . . . . .	53
plotPredictedPDF . . . . .	54
plotPredictionDistribution . . . . .	55
plotPreferencePDF . . . . .	55
plotRoc . . . . .	56
plotSmoothCalibration . . . . .	56
plotSparseCalibration . . . . .	57
plotSparseCalibration2 . . . . .	58
plotSparseRoc . . . . .	58
plotVariableScatterplot . . . . .	59
plpDataSimulationProfile . . . . .	59
positiveLikelihoodRatio . . . . .	60
positivePredictiveValue . . . . .	60
predictFfdf . . . . .	61
predictPlp . . . . .	61
predictProbabilities . . . . .	62
registerParallelBackend . . . . .	63
registerSequentialBackend . . . . .	63
runEnsembleModel . . . . .	64
runPlp . . . . .	65
runPlpAnalyses . . . . .	68
saveEnsemblePlpModel . . . . .	69
saveEnsemblePlpResult . . . . .	70
savePlpData . . . . .	70
savePlpModel . . . . .	71
savePlpResult . . . . .	72
savePrediction . . . . .	72
savePredictionAnalysisList . . . . .	73
sensitivity . . . . .	74
setAdaBoost . . . . .	74
setCIReNN . . . . .	75
setCNNTorch . . . . .	76
setCovNN . . . . .	77
setCovNN2 . . . . .	77
setDecisionTree . . . . .	78
setDeepNN . . . . .	79
setGradientBoostingMachine . . . . .	80
setKNN . . . . .	80
setLassoLogisticRegression . . . . .	81
setLRTorch . . . . .	81
setMLP . . . . .	82
setMLPTorch . . . . .	82
setNaiveBayes . . . . .	83
setRandomForest . . . . .	84
setRNNTorch . . . . .	84
similarPlpData . . . . .	85
simulatePlpData . . . . .	86
specificity . . . . .	87
standardOutput . . . . .	87
submitResults . . . . .	88
timeSplitter . . . . .	88

<i>accuracy</i>	5
toPlpData . . . . .	89
toSparseM . . . . .	90
toSparsePython . . . . .	91
toSparseTorchPython . . . . .	92
transportModel . . . . .	93
transportPlp . . . . .	93
viewPlp . . . . .	94
<b>Index</b>	<b>96</b>

---

<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

---

<code>applyEnsembleModel</code>	<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

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

---

## Description

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

**Usage**

```
applyModel(population, plpData, plpModel, 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 <a href="#">predictProbabilities</a> 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

---

bySumFf	<i>Compute sum of values binned by a second variable</i>
---------	--

---

**Description**

Compute sum of values binned by a second variable

**Usage**

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

**Arguments**

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

**Examples**

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



---

calibrationLine	<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

---

checkffFolder	<i>Check if the fftempdir is writable</i>
---------------	---

---

### Description

Check if the fftempdir is writable

### Usage

```
checkffFolder()
```

### Details

This function checks whether the fftempdir is writable. If not, it will ask the use to specify a writable folder.

---

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).

---

clearffTempDir	<i>clearffTempDir</i>
----------------	-----------------------

---

### Description

Clears the temporary ff directory to free up disk space.

### Usage

```
clearffTempDir()
```

---

 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.

---

```
createCohort
```

*createCohort - Loads all the cohort sql in a network study and creates the cohorts*

---

### Description

This function finds the sql files in a network study package, loads, renders and translates the sql then implements it

### Usage

```
createCohort(cohortDetails, cohortLocation, connectionDetails,
  cdmDatabaseSchema, cohortDatabaseSchema, cohortTable,
  oracleTempSchema = cdmDatabaseSchema, package)
```

**Arguments**

cohortDetails	A dataframe containing two columns: cohortName and cohortId (if missing then the skeleton default is used when available)
cohortLocation	A string specifying the location of the cohort sql files (uses default skeleton location if missing)
connectionDetails	The connection details
cdmDatabaseSchema	A string specifying the CDM database schema e.g., database.dbo
cohortDatabaseSchema	A string specifying the cohort database schema e.g., cohort_database.dbo
cohortTable	A string specifying the cohort table
oracleTempSchema	Temp oracle schema
package	The name of the package

**Details**

This is used by people running network studies using the package skeleton to enable users to create cohorts on their platform

**Value**

A data frame with the cohortName, cohortId, size

---

```
createExistingModelSql
```

*Apply an existing logistic regression prediction model*

---

**Description**

Apply an existing logistic regression prediction model

**Usage**

```
createExistingModelSql(modelTable, modelNames, interceptTable, covariateTable,
  type = "logistic", analysisId = 112, covariateSettings, asFunctions = F,
  customCovariates = NULL, e = environment())
```

**Arguments**

modelTable	A dataframe or list of dataframes with columns: modelId, modelCovariateId, coefficientValue all doubles
modelNames	A name used in the covariate function names (no spaces)
interceptTable	A dataframe or list of dataframes with the columns: modelId, interceptValue
covariateTable	A dataframe or list of dataframes with columns: modelCovariateId, covariateId (the mapping of covariate_id to standard covariates)
type	The type of model: logistic or linear/score

analysisId	The covariate analysis_id (default 112)
covariateSettings	The settings for the standard covariates (needs for temporal settings)
asFunctions	If T then return two functions
customCovariates	enables custome SQL to be used to create custom covariates
e	The environment to output the covariate setting functions to

## Details

This function is used to create custom covariates corresponding to existing models

---

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, saveDir = NULL,
  saveModel = TRUE, verbosity = "TRACE", clearffTemp = FALSE,
  minCovariateFraction = 0.001)
```

## 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"> <li>• setLassoLogisticRegression - a lasso logistic regression model</li> <li>• setGradientBoostingMachine - a gradient boosting machine</li> <li>• setRandomForest - a random forest model</li> <li>• setKNN - a k-nearest neighbour model</li> </ul>
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).
saveDir	The path to the directory where the models will be saved (if NULL, uses working directory).
saveModel	Logical indicating whether to save the model once it has been trained (default is TRUE).
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"> <li>• DEBUG - highest verbosity showing all debug statements</li> <li>• TRACE - showing information about start and end of steps</li> <li>• INFO - show informative messages (default)</li> <li>• WARN - show warning messages</li> <li>• ERROR - show error messages</li> <li>• FATAL - be silent except for fatal errors</li> </ul>
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.

## 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 = "person", testFraction = 0.25, trainFractions = c(0.25, 0.5,
  0.75), splitSeed = NULL, nfold = 3, indexes = NULL,
  minCovariateFraction = 0.001)
```

### Arguments

population	The population created using <code>createStudyPopulation()</code> that will be used to develop the model.
plpData	An object of type <code>plpData</code> - the patient level prediction data extracted from the CDM.
modelSettings	An object of class <code>modelSettings</code> created using one of the function. Currently only one model is supported: <ul style="list-style-type: none"> <li>• <code>setLassoLogisticRegression</code> - a lasso logistic regression model</li> </ul>
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 <code>rowId</code> and <code>index</code> column where the index value of -1 means in the test set, and positive integer represents the cross validation fold (default is NULL).
minCovariateFraction	Minimum covariate prevalence in population to avoid removal during preprocessing.

### 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**

models	A trianed plp model.
modelNames	A name used in the covariate function names (no spaces)
covariateConstructionName	the name used for the create covariate function
modelTable	The temporary table name storing the model details
analysisId	The covariate analysis_id
e	The environment to output the covariate setting functions to
databaseOutput	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"))
```

## 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

## 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

modelList	A list of model settings
covariateSettingList	A list of covariate settings
populationSettingList	A list of population settings
modelSettingNames	A vector of names for each model in modelList
covariateSettingNames	A vector of names for each covariate setting in covariateSettingList
populationSettingNames	A vector of names for each population setting in populationSettingList

### 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 popualtion 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

*createPlpReport*


---

### 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"))
```

**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

**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 = T, minTimeAtRisk = 365, riskWindowStart = 0,
  addExposureDaysToStart = FALSE, riskWindowEnd = 365,
  addExposureDaysToEnd = F, verbosity = "INFO", ...)
```

**Arguments**

plpData	An object of type plpData as generated using getDbplpData.
population	If specified, this population will be used as the starting point instead of the cohorts in the plpData object.
outcomeId	The ID of the outcome. If not specified, no outcome-specific transformations will be performed.
binary	Forces the outcomeCount to be 0 or 1 (use for binary prediction problems)
includeAllOutcomes	(binary) indicating whether to include people with outcomes who are not observed for the whole at risk period

firstExposureOnly	Should only the first exposure per subject be included? Note that this is typically done in the createStudyPopulation function,
washoutPeriod	The 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).
addExposureDaysToStart	Add the length of exposure the start of the risk window?
riskWindowEnd	The end of the risk window (in days) relative to the index data (+ days of exposure if the addExposureDaysToEnd parameter is specified).
addExposureDaysToEnd	Add the length of exposure the risk window?
verbosity	Sets the level of the verbosity. If the log level is at or higher in priority than the logger threshold, a message will print. The levels are: <ul style="list-style-type: none"> <li>• DEBUGHighest verbosity showing all debug statements</li> <li>• TRACEShowing information about start and end of steps</li> <li>• INFOShow informative information (Default)</li> <li>• WARNShow warning messages</li> <li>• ERRORShow error messages</li> <li>• FATALBe silent except for fatal errors</li> </ul>
...	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,
  addExposureDaysToStart = FALSE, riskWindowEnd = 365,
  addExposureDaysToEnd = F, 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).
addExposureDaysToStart	Add the length of exposure the start of the risk window?
riskWindowEnd	The end of the risk window (in days) relative to the index data (+ days of exposure if the addExposureDaysToEnd parameter is specified).
addExposureDaysToEnd	Add the length of exposure the risk window?
verbosity	Sets the level of the verbosity. If the log level is at or higher in priority than the logger threshold, a message will print. The levels are: <ul style="list-style-type: none"> <li>• DEBUG Highest verbosity showing all debug statements</li> </ul>

- 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

---

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
```

*Draw the attrition diagram*

---

### 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

<code>attrition</code>	The table of attrition details return from the population <code>attr(population, 'meta-Data')\$attrition</code>
<code>targetLabel</code>	A label to us for the treated cohort.
<code>outcomeLabel</code>	A label to us for the comparator cohort.
<code>fileName</code>	Name of the file where the plot should 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.

---

```
evaluateExistingModel evaluateExistingModel
```

---

### Description

This function implements an existing model

### Usage

```
evaluateExistingModel(modelTable, covariateTable, interceptTable = NULL,
  type = "score", covariateSettings, customCovariates = NULL,
  riskWindowStart = 1, addExposureDaysToEnd = F, riskWindowEnd = 365,
  requireTimeAtRisk = T, minTimeAtRisk = 364, includeAllOutcomes = T,
  removeSubjectsWithPriorOutcome = T, connectionDetails, cdmDatabaseSchema,
  cohortDatabaseSchema, cohortTable, cohortId, outcomeDatabaseSchema,
  outcomeTable, outcomeId, oracleTempSchema = cdmDatabaseSchema,
  modelName = "existingModel", calibrationPopulation = NULL)
```



**Arguments**

modelTable	The model covariates and scores
covariateTable	The mapping from model covariates to standard covariates
interceptTable	The model intercepts
type	Model type (score or logistic)
covariateSettings	The standard covariate settings (specify covariate lookback time)
customCovariates	A table of covariateId, sql (sql creates the custom covariate)
riskWindowStart	The day after index to start predicting the outcome
addExposureDaysToEnd	riskWindowEnd relative to the cohort end date instead of the cohort start date?
riskWindowEnd	The day after index to stop predicting the outcome
requireTimeAtRisk	Do you want to ignore people who leave the database some point between the riskWindowStart and riskWindowEnd
minTimeAtRisk	If requireTimeAtRisk is TRUE, how many days must they be observed before leaving to get included (default recommendation is all risk period: riskWindowEnd-riskWindowStart)
includeAllOutcomes	Setting this to TRUE means people with the outcome who leave the data during the risk period are still included, so only non-outcome people who leave during the risk period are removed
removeSubjectsWithPriorOutcome	Remove people from the target population if they have the outcome prior to target cohort start date
connectionDetails	The details to connect to the CDM
cdmDatabaseSchema	A string specifying the database containing the cdm
cohortDatabaseSchema	A string specifying the database containing the target cohorts
cohortTable	A string specifying the table containing the target cohorts
cohortId	An integer specifying the cohort id for the target cohorts
outcomeDatabaseSchema	A string specifying the database containing the outcome cohorts
outcomeTable	A string specifying the table containing the outcome cohorts
outcomeId	An integer specifying the cohort id for the outcome cohorts
oracleTempSchema	The temp oracle schema
modelName	The name of the model
calibrationPopulation	A data.frame of subjectId, cohortStartDate, indexes used to recalibrate the model on new data

**Details**

Implements an existing model and evaluates its performance

**Value**

The performance of the existing model and prediction

---

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

**Arguments**

analysisLocation	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"><li>• <b>DEBUG</b>Highest verbosity showing all debug statements</li><li>• <b>TRACE</b>Showing information about start and end of steps</li><li>• <b>INFO</b>Show informative information (Default)</li><li>• <b>WARN</b>Show warning messages</li><li>• <b>ERROR</b>Show error messages</li><li>• <b>FATAL</b>Be silent except for fatal errors</li></ul>
keepPrediction	Whether to keep the predicitions 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

*evaluatePlp*

---

### Description

Evaluates the performance of the patient level prediction model

### Usage

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

### Arguments

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

### Details

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

### Value

A list containing the performance values

---

exportPlpDataToCsv	<i>Export all data in a plpData object to CSV files</i>
--------------------	---

---

## Description

Export all data in a plpData object to CSV files

## Usage

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

## Arguments

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

## Details

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

**cohort.csv** Listing all persons and their prediction periods. This file will have these fields: row\_id (a unique ID per period), person\_id, cohort\_start\_date, cohort\_id, time (number of days in the window).

**outcomes.csv** Listing all outcomes per period. This file will have these fields: row\_id, outcome\_id, outcome\_count, time\_to\_event.

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

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

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

**metaData** Some information on how the plpData object was constructed.

## Examples

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

## End(Not run)
```

---

exportPlpResult	<i>exportPlpResult exports an object returned by runPlp into a network study package while removing sensitive information from the object</i>
-----------------	---

---

## Description

This function should be used to export a prediciton model and performance information into a network study for others to implement on new data for external validation

## Usage

```
exportPlpResult(plpResult, modelName, packageName, gitHubLocation, n = NULL,
  includeEvaluationStatistics = T, includeThresholdSummary = T,
  includeDemographicSummary = T, includeCalibrationSummary = T,
  includePredictionDistribution = T, includeCovariateSummary = F)
```

## Arguments

plpResult	The model to be saved into the package
modelName	The name of the model
packageName	The network study package name
gitHubLocation	The github directory
n	If not null, the minimum number of people required for a row to be included
includeEvaluationStatistics	Whether to include includeEvaluationStatistics evaluation
includeThresholdSummary	Whether to include thresholdSummary evaluation
includeDemographicSummary	Whether to include demographicSummary evaluation
includeCalibrationSummary	Whether to include calibrationSummary evaluation
includePredictionDistribution	Whether to include predictionDistribution evaluation
includeCovariateSummary	Whether to include covariateSummary evaluation

## Details

This is a helper function to perform external validation

## Value

The location of the saved model

---

externalValidatePlp      *externalValidatePlp - Validate a model on new databases*

---

## 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)
```

## Arguments

plpResult	The object returned by runPlp() containing the trained model
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"> <li>• DEBUG Highest verbosity showing all debug statements</li> <li>• TRACE Showing information about start and end of steps</li> <li>• INFO Show informative information (Default)</li> <li>• WARN Show warning messages</li> <li>• ERROR Show error messages</li> <li>• FATAL Be silent except for fatal errors</li> </ul>
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 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	Calculate the falseNegativeRate
-------------------	---------------------------------

---

Description

Calculate the falseNegativeRate

Usage

falseNegativeRate(TP, TN, FN, FP)

Arguments

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

Details

Calculate the falseNegativeRate

Value

falseNegativeRate value



---

falseOmissionRate	<i>Calculate the falseOmissionRate</i>
-------------------	--

---

**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	<i>Calculate the falsePositiveRate</i>
-------------------	--

---

**Description**

Calculate the falsePositiveRate

**Usage**

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

**Arguments**

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

**Details**

Calculate the falsePositiveRate

**Value**

falsePositiveRate value

---

fitGLMModel	<i>Fit a predictive model</i>
-------------	-------------------------------

---

### Description

Fit a predictive model

### Usage

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

### Arguments

population	A population object generated by <code>createStudyPopulation</code> , potentially filtered by other functions.
plpData	An object of type <code>plpData</code> as generated using <code>getDbPlpData</code> .
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 <a href="#">createPrior</a> for details.
control	The control object used to control the cross-validation used to determine the hyperparameters of the prior (if applicable). See <a href="#">createControl</a> for details.

---

fitPlp	<i>fitPlp</i>
--------	---------------

---

### 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"> <li>• logisticRegressionModel() A lasso logistic regression model</li> <li>• GBMclassifier() A gradient boosting machine</li> <li>• RFclassifier() A random forest model</li> <li>• GLMclassifier () A generalised linear model</li> <li>• KNNclassifier() A KNN model</li> </ul>
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.

---

<code>getCalibration</code>	<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

---

<code>getCovariateData</code>	<i>Get the covariate data for a cohort table</i>
-------------------------------	--

---

**Description**

This function executes some SQL to extract covariate data for a cohort table

**Usage**

```
getCovariateData(connection, cdmDatabaseSchema,
  oracleTempSchema = cdmDatabaseSchema, cohortTable = "#cohort_person",
  cdmVersion = 5, covariateSettings)
```

**Arguments**

connection	Can also use an existing connection rather than the connectionDetails
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.
cohortTable	The temp table containing the cohort of people
cdmVersion	The version of the CDM (default 5)
covariateSettings	An object of type covariateSettings as created using the createCovariateSettings function in the FeatureExtraction package.

**Value**

Returns the covariates for the people in the temp table

---

getModelDetails	<i>Get the predictive model details</i>
-----------------	---

---

**Description**

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

**Usage**

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

**Arguments**

predictiveModel	An object of type predictiveModel as generated using the <a href="#">fitPlp</a> function.
plpData	An object of type plpData as generated using <a href="#">getPlpData</a> .

**Details**

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

---

getPlpData

*Get the patient level prediction data from the server*


---

## Description

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

## Usage

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

## 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. If cohortTable = DRUG\_ERA, cohortId is a CONCEPT\_ID and all descendant concepts within that CONCEPT\_ID will be used to define the cohort. If cohortTable <> DRUG\_ERA, cohortId is used to select the cohort\_concept\_id in the cohort-like table.

outcomeIds

A list of cohort\_definition\_ids used to define outcomes (-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. If cohortTable = DRUG\_ERA, cohortDatabaseSchema is not used by assumed to be cdmSchema. Requires read permissions to this database.

cohortTable	The tablename that contains the at risk cohort. If cohortTable <> DRUG_ERA, then expectation is cohortTable has format of COHORT table: cohort_concept_id, SUBJECT_ID, COHORT_START_DATE, COHORT_END_DATE.
outcomeDatabaseSchema	The name of the database schema that is the location where the data used to define the outcome cohorts is available. If cohortTable = CONDITION_ERA, exposureDatabaseSchema is not used by assumed to be cdmSchema. Requires read permissions to this database.
outcomeTable	The tablename that contains the outcome cohorts. If outcomeTable <> CONDITION_OCCURRENCE, then expectation is outcomeTable has format of COHORT table: COHORT_DEFINITION_ID, SUBJECT_ID, COHORT_START_DATE, COHORT_END_DATE.
cdmVersion	Define the OMOP CDM version used: currently support "4" and "5".
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
baseUrl	If extracting cohorts from atlas enter atlas url to extract cohort creation details

## 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 ffdi 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 ffdi 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.

---

getPlpTable	<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

cdmDatabaseSchema	The schema containing the OMOP CDM data
oracleTempSchema	The oracle schema if needed
covariateSettings	The covariateSettings if different from default
longTermStartDays	How far to look back when looking for the variables in the data
population	The population you want the summary table for
connectionDetails	The connection details used to connect to the CDM database
cohortTable	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

---

grepCovariateNames	<i>Extract covariate names</i>
--------------------	--------------------------------

---

### Description

Extracts covariate names using a regular-expression.

### Usage

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

### Arguments

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

### Details

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

### Value

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

**covariateId** Numerical identifier for use in model fitting using these covariates  
**covariateName** Text identifier  
**analysisId** Analysis identifier  
**conceptId** OMOP common data model concept identifier, or 0

---

insertDbPopulation	<i>Insert a population into a database</i>
--------------------	--

---

### Description

Insert a population into a database

### Usage

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

**Arguments**

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

**Details**

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

---

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

---

`loadEnsemblePlpModel`    *loads the Ensemble plp model and return a model list*

---

**Description**

loads the Ensemble 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`    *loads the Ensemble plp results*

---

**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 prediciton dataframe to csv</i>
----------------	--

---

**Description**

Loads the prediciton dataframe to csv

**Usage**

```
loadPrediction(dirPath)
```

**Arguments**

dirPath	The directory to saved the csv
---------	--------------------------------

**Details**

Loads the prediciton csv file

---

`loadPredictionAnalysisList`*Load the multiple prediction json settings from a file*

---

**Description**

Load the multiple prediction json settings from a file

**Usage**

```
loadPredictionAnalysisList(predictionAnalysisListFile)
```

**Arguments**

`predictionAnalysisListFile`

The prediction 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



---

packageResults	<i>Package the results for sharing with OHDSI researchers</i>
----------------	---

---

**Description**

Package the results for sharing with OHDSI researchers

**Usage**

```
packageResults(mainFolder, includeROCplot = T, includeCalibrationPlot = T,
  includePRPlot = T, includeTable1 = T, includeThresholdSummary = T,
  includeDemographicSummary = T, includeCalibrationSummary = T,
  includePredictionDistribution = T, includeCovariateSummary = T,
  removeLessThanN = T, N = 5)
```

**Arguments**

mainFolder	The location of the folder with the standard output
includeROCplot	Whether to include ROC plot
includeCalibrationPlot	Whether to include calibration plot
includePRPlot	Whether to include precision recall plot
includeTable1	Whether to include table1
includeThresholdSummary	Whether to include thresholdSummary evaluation
includeDemographicSummary	Whether to include demographicSummary evaluation
includeCalibrationSummary	Whether to include calibrationSummary evaluation
includePredictionDistribution	Whether to include predictionDistribution evaluation
includeCovariateSummary	Whether to include covariateSummary evaluation
removeLessThanN	Whether to remove any entry with less than N people
N	If removeLessThanN is TRUE the value for N

**Details**

This function packages the results.

---

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

---

**Description**

PatientLevelPrediction

---

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 $\geq 1$ 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	<i>Plot the Observed vs. expected incidence, by age and gender</i>
------------------------	--

---

**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 <a href="#">runPlp</a> 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	<i>Plot the F1 measure efficiency frontier using the sparse thresholdSummary data frame</i>
---------------	---

---

**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 <a href="#">runPlp</a> function.
type	options: 'train' or test'
fileName	Name of the file where the plot should be saved, for example 'plot.png'. See the function <a href="#">ggsave</a> in the <a href="#">ggplot2</a> 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	<i>Plot the train/test generalizability diagnostic</i>
----------------------	--

---

**Description**

Plot the train/test generalizability diagnostic

**Usage**

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

**Arguments**

covariateSummary	A prediction object as generated using the <a href="#">runPlp</a> 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**

learningCurve	An object returned by <a href="#">createLearningCurve</a> function.
metric	Specifies the metric to be plotted: <ul style="list-style-type: none"> <li>'AUROC' - use the area under the Receiver Operating Characteristic curve</li> <li>'AUPRC' - use the area under the Precision-Recall curve</li> <li>'sBrier' - use the scaled Brier score</li> </ul>
abscissa	Specify the abscissa metric to be plotted: <ul style="list-style-type: none"> <li>'observations' - use number of observations</li> <li>'outcomes' - use number of positive outcomes</li> </ul>
plotTitle	Title of the learning curve plot.
plotSubtitle	Subtitle of the learning curve plot.
fileName	Filename of plot to 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.

**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.

**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 <a href="#">runPlp</a> function.
type	options: 'train' or test'
fileName	Name of the file where the plot should be saved, for example 'plot.png'. See the function <a href="#">ggsave</a> in the <a href="#">ggplot2</a> 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 <a href="#">runPlp</a> function.
type	options: 'train' or test'
fileName	Name of the file where the plot should be saved, for example 'plot.png'. See the function <a href="#">ggsave</a> in the <a href="#">ggplot2</a> package for supported file formats.

**Details**

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

**Value**

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

---

plotPredictionDistribution

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


---

### Description

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

### Usage

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

### Arguments

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

### Details

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

### Value

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

---

plotPreferencePDF

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


---

### Description

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

### Usage

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

### Arguments

evaluation	A prediction object as generated using the <a href="#">runPlp</a> function.
type	options: 'train' or test'
fileName	Name of the file where the plot should be saved, for example 'plot.png'. See the function <a href="#">ggsave</a> in the <a href="#">ggplot2</a> 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 <a href="#">predictProbabilities</a> function.
fileName	Name of the file where the plot should be saved, for example 'plot.png'. See the function <a href="#">ggsave</a> in the <a href="#">ggplot2</a> 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	<i>Plot the smooth calibration as detailed in Calster et al. "A calibration heirarchy for risk models was defined: from utopia to empirical data" (2016)</i>
-----------------------	--

---

**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 <a href="#">runPlp</a> 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 [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 <a href="#">runPlp</a> 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 <a href="#">runPlp</a> function.
type	options: 'train' or test'
fileName	Name of the file where the plot should be saved, for example 'plot.png'. See the function <a href="#">ggsave</a> in the <a href="#">ggplot2</a> 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 <a href="#">runPlp</a> function.
type	options: 'train' or test'
fileName	Name of the file where the plot should be saved, for example 'plot.png'. See the function <a href="#">ggsave</a> in the <a href="#">ggplot2</a> 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

*Calculate the positiveLikelihoodRatio*

---

**Description**

Calculate the positiveLikelihoodRatio

**Usage**

positiveLikelihoodRatio(TP, TN, FN, FP)

**Arguments**

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

**Details**

Calculate the positiveLikelihoodRatio

**Value**

positiveLikelihoodRatio value

---

positivePredictiveValue

*Calculate the positivePredictiveValue*

---

**Description**

Calculate the positivePredictiveValue

**Usage**

positivePredictiveValue(TP, TN, FN, FP)

**Arguments**

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

**Details**

Calculate the positivePredictiveValue

**Value**

positivePredictiveValue value

---

predictFfdf	<i>Generated predictions from a regression model</i>
-------------	--

---

**Description**

Generated predictions from a regression model

**Usage**

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

**Arguments**

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

**Details**

These columns are expected in the outcome object:

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

These columns are expected in the covariates object:

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

---

predictPlp	<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, covariates)
```

**Arguments**

predictiveModel	An object of type predictiveModel as generated using <a href="#">fitPlp</a> .
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.

---

```
registerParallelBackend
      registerParallelBackend
```

---

**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
      registerSequentialBackend
```

---

**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 ensembling 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"> <li>• DEBUGHighest verbosity showing all debug statements</li> <li>• TRACEShowing information about start and end of steps</li> <li>• INFOShow informative information (Default)</li> <li>• WARNShow warning messages</li> </ul>



- **ERROR**Show error messages
- **FATAL**Be 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.

---

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 = "time", 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"> <li>• logisticRegressionModel() A lasso logistic regression model</li> <li>• GBMclassifier() A gradient boosting machine</li> <li>• RFclassifier() A random forest model</li> <li>• GLMclassifier () A generalised linear model</li> <li>• KNNclassifier() A KNN model</li> </ul> |

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.
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"> <li>• DEBUGHighest verbosity showing all debug statements</li> <li>• TRACEShowing information about start and end of steps</li> <li>• INFOShow informative information (Default)</li> <li>• WARNShow warning messages</li> <li>• ERRORShow error messages</li> <li>• FATALBe silent except for fatal errors</li> </ul>
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 <a href="#">createPlpModelSettings</a> 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 'length(cohortIds) \* length(outcomeIds) \* length(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 csv</i>
----------------	--

---

**Description**

Saves the prediction dataframe to csv

**Usage**

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

**Arguments**

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

**Details**

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



---

savePredictionAnalysisList

*Saves a json prediction settings given R settings*


---

## 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	Fractiuon 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

*Create setting for CIReNN model***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(1), batchSize = c(100),
  epochs = c(100), earlyStoppingMinDelta = c(1e-04),
  earlyStoppingPatience = c(10), useVae = T,
  vaeDataSamplingProportion = 0.1, vaeValidationSplit = 0.2,
  vaeBatchSize = 100L, vaeLatentDim = 10L, vaeIntermediateDim = 256L,
  vaeEpoch = 100L, vaeEpsilonStd = 1, seed = NULL)
```

**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 recurrent 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
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.
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
seed	Random seed used by deep learning model

### Examples

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

## End(Not run)
```

---

setCNNTorch	<i>Create setting for CNN model with python</i>
-------------	---

---

### Description

Create setting for CNN model with python

### Usage

```
setCNNTorch(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 <- setCNNTorch()

## End(Not run)
```

---

setCovNN	Create setting for multi-resolution CovNN model (stucture based on <a href="https://arxiv.org/pdf/1608.00647.pdf">https://arxiv.org/pdf/1608.00647.pdf</a> 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	Create setting for CovNN2 model - convolution across input and time - <a href="https://arxiv.org/pdf/1608.00647.pdf">https://arxiv.org/pdf/1608.00647.pdf</a>
-----------	---

---

### 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)
```

---

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

---

*Create setting for DeepNN model*


---

### 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)
```

---

```
setGradientBoostingMachine
```

*Create setting for gradient boosting machine model using gbm\_xgboost implementation*

---

### Description

Create setting for gradient boosting machine model using gbm\_xgboost implementation

### Usage

```
setGradientBoostingMachine(ntrees = c(10, 100), nthread = 20,
  maxDepth = c(4, 6, 17), minRows = 20, learnRate = c(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?)
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"))
```

### Arguments

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



**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)
```

---

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)
```

---

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)
```

---

similarPlpData	<i>Extract new plpData using plpModel settings use metadata in plp-Model to extract similar data and population for new databases:</i>
----------------	--

---

**Description**

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

**Usage**

```
similarPlpData(plpModel = NULL, createCohorts = T, newConnectionDetails,
  newCdmDatabaseSchema = NULL, newCohortDatabaseSchema = NULL,
  newCohortTable = NULL, newCohortId = NULL,
  newOutcomeDatabaseSchema = NULL, newOutcomeTable = NULL,
  newOutcomeId = NULL, newOracleTempSchema = newCdmDatabaseSchema,
  sample = NULL, createPopulation = T)
```

**Arguments**

plpModel	The trained PatientLevelPrediction model or object returned by runPlp()
createCohorts	Create the tables for the target and outcome - requires sql in the plpModel object
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

**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

---

standardOutput	<i>standardOutput - takes the output of runPlp or evaluatePlp and converts it into the standardised output for a network study - three directories (plots, results, summary)</i>
----------------	--

---

**Description**

This function saves the plp study results into standardised output

**Usage**

```
standardOutput(result, table1, outputLocation, studyName, databaseName,  
cohortName, outcomeName)
```

**Arguments**

result	The result of the network study
table1	the table1 result
outputLocation	The location where the results will be saved - need to have write access
studyName	The name of the network study
databaseName	The name of the cdm database
cohortName	The name of the target population cohort
outcomeName	The name of the outcome cohort

**Details**

This is used to ensure each study collects results consistently

**Value**

The location of the saved results

---

submitResults	<i>submitResults - sends a zipped folder to the OHDSI network study repository</i>
---------------	--

---

**Description**

This function takes as input a zipped folder location and sends it to the OHDSI amazon repository

**Usage**

```
submitResults(exportFolder, key, secret)
```

**Arguments**

exportFolder	The path to the folder containing the study results compressed file.
key	The key string as provided by the study coordinator
secret	The secret string as provided by the study coordinator

**Details**

This is used at the end of a network study to submit the results once a user has checked the folder

**Value**

TRUE if the upload was successful.

---

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 $\geq 1$ 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

---

toPlpData	<i>Convert matrix into plpData</i>
-----------	------------------------------------

---

**Description**

Converts a matrix (rows = people, columns=variables) into the standard plpData

**Usage**

```
toPlpData(data, columnInfo, outcomeId, outcomeThreshold = 0.5,
  indexTime = 0, includeIndexDay = T)
```

**Arguments**

data	An data.frame or matrix.
columnInfo	A dataframe with three columns, column 1 contains columnId, column 2 contains columnName for each column id and column 3 contains the columnTime - the time prior to index the variable was recorded
outcomeId	The column id containing the outcome
outcomeThreshold	The outcome value must be higher or equal to this for the person to have the outcome
indexTime	The time defining the index date
includeIndexDay	Boolean - whether to include variables recorded on index date

**Details**

This function converts matrix into plpData

**Value**

Returns an object of class plpData

**Examples**

```
#TODO
```

---

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 ffd in COO format into a sparse matrix from the package Matrix

**Value**

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

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

**covariateRef** The plpData covariateRef.

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

**Examples**

```
#TODO
```

---

toSparsePython	<i>Convert the plpData in COO format into a sparse python matrix</i>
----------------	--

---

## Description

Converts the standard plpData to a sparse matrix directly into python

## Usage

```
toSparsePython(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 ffdF in COO format into a sparse matrix from the package Matrix

## Value

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

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

**covariateRef** The plpData covariateRef.

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

## Examples

```
#TODO
```

---

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 firectly 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 ffdF in COO format into a sparse matrix from the package Matrix

## Value

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

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

**covariateRef** The plpData covariateRef.

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

## Examples

```
#TODO
```

---

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)
```

---

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, 59
- accuracy, 5
- applyEnsembleModel, 5
- applyModel, 6
- averagePrecision, 7
- brierScore, 8
- bySumFf, 8
- calibrationLine, 9
- checkfffFolder, 9
- checkPlpInstallation, 10
- clearfffTempDir, 10
- combinePlpModelSettings, 11
- computeAuc, 11
- computeAucFromDataFrames, 12
- createCohort, 12
- createControl, 34
- createExistingModelSql, 13
- createLearningCurve, 14, 52
- createLearningCurvePar, 16
- createLrSql, 17
- createPlpJournalDocument, 18
- createPlpModelSettings, 19, 69
- createPlpReport, 19
- createPrior, 34
- createStudyPopulation, 20
- createStudyPopulationSettings, 22
- diagnosticOddsRatio, 23
- drawAttritionDiagramPlp, 24
- evaluateExistingModel, 24
- evaluateMultiplePlp, 26
- evaluatePlp, 27
- exportPlpDataToCsv, 28
- exportPlpResult, 29
- externalValidatePlp, 30
- f1Score, 31
- falseDiscoveryRate, 31
- falseNegativeRate, 32
- falseOmissionRate, 33
- falsePositiveRate, 33
- fitGLMModel, 34
- fitPlp, 34, 37, 62
- getAttritionTable, 35
- getCalibration, 36
- getCovariateData, 36
- getModelDetails, 37
- getPlpData, 37, 38
- getPlpTable, 40
- getPredictionDistribution, 41
- getThresholdSummary, 41
- ggsave, 24, 51, 52, 54–59
- grepCovariateNames, 42
- insertDbPopulation, 42
- interpretInstallCode, 43
- loadEnsemblePlpModel, 44
- loadEnsemblePlpResult, 44
- loadPlpData, 45
- loadPlpModel, 45
- loadPlpResult, 46
- loadPrediction, 46
- loadPredictionAnalysisList, 47
- negativeLikelihoodRatio, 47
- negativePredictiveValue, 48
- packageResults, 49
- PatientLevelPrediction, 49
- PatientLevelPrediction-package  
(PatientLevelPrediction), 49
- personSplitter, 50
- plotDemographicSummary, 50
- plotF1Measure, 51
- plotGeneralizability, 51
- plotLearningCurve, 52
- plotPlp, 53
- plotPrecisionRecall, 53
- plotPredictedPDF, 54
- plotPredictionDistribution, 55
- plotPreferencePDF, 55
- plotRoc, 56
- plotSmoothCalibration, 56



plotSparseCalibration, 57  
plotSparseCalibration2, 58  
plotSparseRoc, 58  
plotVariableScatterplot, 59  
plpDataSimulationProfile, 59  
positiveLikelihoodRatio, 60  
positivePredictiveValue, 60  
predict, 11, 36  
predictFfdF, 61  
predictPlp, 61  
predictProbabilities, 7–9, 41, 56, 62  
  
registerParallelBackend, 63  
registerSequentialBackend, 63  
runEnsembleModel, 64  
runPlp, 50–52, 54, 55, 57–59, 65  
runPlpAnalyses, 68  
  
save\_plot(), 57  
saveEnsemblePlpModel, 69  
saveEnsemblePlpResult, 70  
savePlpData, 70  
savePlpModel, 71  
savePlpResult, 72  
savePrediction, 72  
savePredictionAnalysisList, 73  
sensitivity, 74  
setAdaBoost, 74  
setCIReNN, 75  
setCNNTorch, 76  
setCovNN, 77  
setCovNN2, 77  
setDecisionTree, 78  
setDeepNN, 79  
setGradientBoostingMachine, 80  
setKNN, 80  
setLassoLogisticRegression, 81  
setLRTorch, 81  
setMLP, 82  
setMLPTorch, 82  
setNaiveBayes, 83  
setRandomForest, 84  
setRNNTorch, 84  
similarPlpData, 85  
simulatePlpData, 86  
specificity, 87  
standardOutput, 87  
submitResults, 88  
  
timeSplitter, 88  
toPlpData, 89  
toSparseM, 90  
toSparsePython, 91  
toSparseTorchPython, 92  
transportModel, 93  
transportPlp, 93  
  
viewPlp, 94