

Package ‘EnsemblePatientLevelPrediction’

January 27, 2022

Type Package

Title Expanding PatientLevelPrediction with ensemble capabilities

Version 0.0.1

Author Jenna Reys [aut, cre]

Maintainer Jenna M Reys <reys@ohdsi.org>

Description

This package uses functions in PatientLevelPrediction to develop and evaluate ensemble models.

Depends R (>= 3.5.0)

Imports dplyr,

PatientLevelPrediction (>= 5.0.0),

ParallelLogger,

rlang,

tidyr

Suggests DT,

Eunomia,

knitr,

rmarkdown,

shiny,

testthat

Remotes ohdsi/PatientLevelPrediction,

ohdsi/Eunomia

License Apache License 2.0

VignetteBuilder knitr

LazyData TRUE

RoxygenNote 7.1.1

R topics documented:

applyEnsemble	2
applyEnsembleToPredictions	2
createFusionCombiner	3
EnsemblePatientLevelPrediction	3
loadEnsemble	4
loadEnsembleModel	4
runEnsemble	5

saveEnsemble	5
saveEnsembleModel	6
setEnsembleFromDesign	6
setEnsembleFromFiles	7
setEnsembleFromResults	8

Index	10
--------------	-----------

applyEnsemble	<i>Apply an ensembleModel to new data</i>
---------------	---

Description

Apply an ensembleModel to new data

Usage

```
applyEnsemble(  
  ensembleModel,  
  newDatabaseDetails,  
  logSettings = PatientLevelPrediction::createLogSettings(),  
  outputFolder  
)
```

Arguments

- ensembleModel An ensembleModel
- newDatabaseDetails A databaseDetails object for the new database created using PatientLevelPrediction::createDatabaseDetails
- logSettings The log settings
- outputFolder The location to save the base model results

applyEnsembleToPredictions	<i>Apply an ensembleModel to list of base model prediction objects</i>
----------------------------	--

Description

Apply an ensembleModel to list of base model prediction objects

Usage

```
applyEnsembleToPredictions(predictionList, ensemble)
```

Arguments

- predictionList A list of base model prediction objects
- ensemble An ensembleModel

createFusionCombiner *Create the settings for a fusion ensemble*

Description

Create the settings for a fusion ensemble

Usage

```
createFusionCombiner(
  type = c("uniform", "AUROC", "AUPRC")[1],
  evaluation = "CV",
  scaleFunction = "normalize"
)
```

Arguments

type	The type of fusion ensemble pick from: 'uniform', 'AUROC', 'AUPRC' or any other metric from evaluationSummary
evaluation	The evaluation type used to learn the weights (if evaluation is CV and type is 'AUROC' then the cross validation AUROC is used to determine the weight given to of the base models)
scaleFunction	How to scale the weights (normalize means weights add up to 1)

Examples

```
## Not run:

comSettingNormAUC <- createFusionCombiner(type = "AUROC",
                                           evaluation = "CV",
                                           scaleFunction = "normalize")

comSettingUniform <- createFusionCombiner(type = "uniform",
                                           evaluation = "CV",
                                           scaleFunction = "normalize")

## End(Not run)
```

EnsemblePatientLevelPrediction
EnsemblePatientLevelPrediction

Description

A package for developing ensembles using the PatientLevelPrediction framework

loadEnsemble	<i>Load a previously saved ensemble result</i>
--------------	--

Description

Load a previously saved ensemble result

Usage

```
loadEnsemble(dirPath)
```

Arguments

dirPath	The directory to the saved ensemble result
---------	--

Examples

```
## Not run:  
  
ensemble <- loadEnsemble("C:/bestEnsemble")  
  
## End(Not run)
```

loadEnsembleModel	<i>Load a previously saved ensemble model (object of class ensemble-Model)</i>
-------------------	--

Description

Load a previously saved ensemble model (object of class ensembleModel)

Usage

```
loadEnsembleModel(dirPath)
```

Arguments

dirPath	The directory containing the saved ensemble model
---------	---

Examples

```
## Not run:  
  
ensembleModel <- loadEnsembleModel("C:/bestEnsembleModel")  
  
## End(Not run)
```

`runEnsemble`*Code to run the ensemble model development*

Description

Code to run the ensemble model development

Usage

```
runEnsemble(  
  ensembleSettings,  
  logSettings = PatientLevelPrediction::createLogSettings(logName = "ensemble"),  
  saveDirectory  
)
```

Arguments

`ensembleSettings`

The ensemble specifications created using `setEnsemble()`

`logSettings`

The settings used to specify the logging, created using `PatientLevelPrediction::createLogSettings()`

`saveDirectory`

The location to save the ensemble

`saveEnsemble`*Save an ensemble result*

Description

Save an ensemble result

Usage

```
saveEnsemble(ensemble, dirPath)
```

Arguments

`ensemble`

An `ensemblePlp` created by `EnsemblePatientLevelPrediction::runEnsemble()`

`dirPath`

The directory to save the ensemble result

Examples

```
## Not run:
```

```
saveEnsemble(ensemble, "C:/bestEnsemble")
```

```
## End(Not run)
```

saveEnsembleModel	<i>Save an ensemble model (object of class ensembleModel)</i>
-------------------	---

Description

Save an ensemble model (object of class ensembleModel)

Usage

```
saveEnsembleModel(ensembleModel, dirPath)
```

Arguments

ensembleModel	An ensembleModel
dirPath	The directory to save the ensemble model

Examples

```
## Not run:

saveEnsembleModel(ensembleModel, "C:/bestEnsembleModel")

## End(Not run)
```

setEnsembleFromDesign	<i>Create setting for creating ensemble from model settings</i>
-----------------------	---

Description

Create setting for creating ensemble from model settings

Usage

```
setEnsembleFromDesign(
  modelDesignList,
  databaseDetails,
  splitSettings = PatientLevelPrediction::createDefaultSplitSetting(),
  filterSettings,
  combinerSettings
)
```

Arguments

modelDesignList	A list of model designs to develop and then combine. Each model design is created by PatientLevelPrediction::createModelDesign()
databaseDetails	The OMOP CDM database details and connection for extracting the data

`splitSettings` The test/train and cross validation settings created using `PatientLevelPrediction::createDefaultSplitSettings()` or via a custom function

`filterSettings` Setting specifying rules to use to filter (remove) any model specified in the list of model designs that performs insufficiently (these models get ignored from the ensemble)

`combinerSettings` Settings specifying how to combine the remaining models into an ensemble

Examples

```
## Not run:
modelDesign1 <- PatientLevelPrediction::createModelDesign(targetId = 4,
  outcomeId = 3,
  restrictPlpDataSettings = restrictPlpDataSettings,
  covariateSettings = covSet,
  runCovariateSummary = F,
  modelSettings = PatientLevelPrediction::setLassoLogisticRegression(),
  populationSettings = populationSet,
  preprocessSettings = PatientLevelPrediction::createPreprocessSettings(),
modelDesign2 <- PatientLevelPrediction::createModelDesign(targetId = 4,
  outcomeId = 3,
  restrictPlpDataSettings = restrictPlpDataSettings,
  covariateSettings = covSet,
  runCovariateSummary = F,
  modelSettings = PatientLevelPrediction::setGradientBoostingMachine(),
  populationSettings = populationSet,
  preprocessSettings = PatientLevelPrediction::createPreprocessSettings(),

ensembleSettings <- setEnsembleFromDesign(modelDesignList = list(modelDesign1, modelDesign2),
  databaseDetails = PatientLevelPrediction::createDatabaseDetails(),
  splitSettings = PatientLevelPrediction::createDefaultSplitSetting(),
  filterSettings = list(minValue = 0.5, maxValue = 1),
  combinerSettings = createFusionCombiner(type = "uniform",
    evaluation = "CV",
    scaleFunction = "normalize"))

## End(Not run)
```

`setEnsembleFromFiles` *Create setting for creating ensemble from model settings*

Description

Create setting for creating ensemble from model settings

Usage

```
setEnsembleFromFiles(fileVector, filterSettings, combinerSettings)
```

Arguments

fileVector	A vector of files containing the location of the PatientLevelPrediction::runPlp() results
filterSettings	Setting specifying rules to use to filter (remove) any model specified in the list of model designs that performs insufficiently (these models get ignored from the ensemble)
combinerSettings	Settings specifying how to combine the remaining models into an ensemble

Examples

```
## Not run:

ensembleSettings <- setEnsembleFromFiles(fileVector = c("./result1", "./result2", "./result3"),
                                         filterSettings = list(minValue = 0.5, maxValue = 1),
                                         combinerSettings = createFusionCombiner(type = "uniform",
                                                                                   evaluation = "CV",
                                                                                   scaleFunction = "normalize"))

## End(Not run)
```

```
setEnsembleFromResults
```

Create setting for creating ensemble from model settings

Description

Create setting for creating ensemble from model settings

Usage

```
setEnsembleFromResults(resultList, filterSettings, combinerSettings)
```

Arguments

resultList	A list of runPlp results to combine.
filterSettings	Setting specifying rules to use to filter (remove) any model specified in the list of model designs that performs insufficiently (these models get ignored from the ensemble)
combinerSettings	Settings specifying how to combine the remaining models into an ensemble

Examples

```
## Not run:

plpResult1 <- PatientLevelPrediction::loadPlpResult("./result1")
plpResult2 <- PatientLevelPrediction::loadPlpResult("./result2")
plpResult3 <- PatientLevelPrediction::loadPlpResult("./result3")

ensembleSettings <- setEnsembleFromResults(resultList = list(plpResult1, plpResult2, plpResult3),
```



```
        filterSettings = list(minValue = 0.5, maxValue = 1),
combinerSettings = createFusionCombiner(type = "uniform",
                                         evaluation = "CV",
                                         scaleFunction = "normalize"))

## End(Not run)
```

Index

`applyEnsemble`, [2](#)
`applyEnsembleToPredictions`, [2](#)

`createFusionCombiner`, [3](#)

`EnsemblePatientLevelPrediction`, [3](#)

`loadEnsemble`, [4](#)
`loadEnsembleModel`, [4](#)

`runEnsemble`, [5](#)

`saveEnsemble`, [5](#)
`saveEnsembleModel`, [6](#)
`setEnsembleFromDesign`, [6](#)
`setEnsembleFromFiles`, [7](#)
`setEnsembleFromResults`, [8](#)