Package: LOGANTree (via r-universe)

September 2, 2024

Type Package

Title Tree-Based Models for the Analysis of Log Files from Computer-Based Assessments

Version 0.1.1

Description Enables researchers to model log-file data from computer-based assessments using machine-learning techniques. It allows researchers to generate new knowledge by comparing the performance of three tree-based classification models (i.e., decision trees, random forest, and gradient boosting) to predict student's outcome. It also contains a set of handful functions for the analysis of the features' influence on the modeling. Data from the Climate control item from the 2012 Programme for International Student Assessment (PISA, <https://www.oecd.org/pisa/>) is available for an illustration of the package's capability. He, Q., & von Davier, M. (2015) <doi:10.1007/978-3-319-19977-1_13> Boehmke, B., & Greenwell, B. M. (2019) <doi:10.1201/9780367816377> .

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.1.2

Depends R (>= 3.5.0)

Imports ROCR, caret, caretEnsemble, dplyr, ggplot2, rpart.plot, tibble, gbm, stats

NeedsCompilation no

Author Denise Reis Costa [aut, ths], Qi Qin [aut, cre]

Maintainer Qi Qin <logantreeqq@gmail.com>

Repository CRAN

Date/Publication 2022-06-22 22:20:06 UTC

Contents

`ChiSquarePlot` *Plot for Chi-square Statistics*

Description

Plot for Chi-square Statistics

Usage

```
ChiSquarePlot(
  trainingdata = NULL,
  nfeatureNames = NULL,
  outcome = NULL,
  level = NULL,
  ModelObject = NULL
)
```

Arguments

`trainingdata`	A data set used for training
`nfeatureNames`	A vector of feature names that will be used for computing chi-square statistics
`outcome`	A character string with the name of the binary outcome variable.
`level`	A numerical value indicating the number of categories that the outcome contains
`ModelObject`	A model object containing tree-based models

Value

This function returns a barplot of scaled chi-square statistics for the study's features. These measures were computed as described by He & von Davier (2015).

References

He, Q., & von Davier, M. (2015). Identifying feature sequences from process data in problem-solving items with N-grams. In Quantitative Psychology Research: The 79th Annual Meeting of the Psychometric Society (pp. 173–190). Madison, Wisconsin: Springer International Publishing.

Examples

```
colnames(training)[14] <- "perf"
ensemblist <- TreeModels(traindata = training,
methodlist = c("dt", "gbm"),checkprogress = TRUE)

ChiSquarePlot(trainingdata = training,
nfeatureNames = colnames(training[,7:13]),
outcome = "perf", level = 2, ModelObject = ensemblist$ModelObject)
```

ChiSquareTable *Chi-square Statistics Table*

Description

Chi-square Statistics Table

Usage

```
ChiSquareTable(
  trainingdata = NULL,
  nfeatureNames = NULL,
  outcome = NULL,
  level = NULL,
  ModelObject = NULL
)
```

Arguments

`trainingdata`	A data set used for training
`nfeatureNames`	A vector of feature names that will be used for computing chi-square statistics
`outcome`	A character string with the name of the binary outcome variable.
`level`	A numerical value indicating the number of categories that the outcome contains
`ModelObject`	A model object containing tree-based models

Value

This function returns a table with five columns. The chi-square statistics were computed as described by He & von Davier (2015).

Feature: Features names

CvAverageChisq: Average chisquare statistics computed from 10-fold cross validation samples

Rank.CvAverageChisq: Ordem of the feature importance from the CvAverageChisq measures#'

OverallChisq: chisquare scores computed from the whole training sample

Rank.OverallChisq: Ordem of the feature importance from the OverallChisq measures

References

He, Q., & von Davier, M. (2015). Identifying feature sequences from process data in problem-solving items with N-grams. In Quantitative Psychology Research: The 79th Annual Meeting of the Psychometric Society (pp. 173–190). Madison, Wisconsin: Springer International Publishing.

Examples

```
colnames(training)[14] <- "perf"
ensemblist <- TreeModels(traindata = training,
methodlist = c("dt", "gbm"),checkprogress = TRUE)

ChiSquareTable(trainingdata=training,
nfeatureNames=colnames(training[,7:13]),
outcome = "perf",level = 2, ModelObject = ensemblist$ModelObject)
```

`ComputeChisquared` *Compute the chi-square scores of features*

Description

Compute the chi-square scores of features

Usage

```
ComputeChisquared(data, outcome, level, weight = FALSE, ctable = FALSE)
```

Arguments

`data`	A dataset containing an outcome variable and action features with either raw frequencies or weighted frequencies.
`outcome`	Name of the outcome variable.
`level`	The level of outcome. e.g. correct/incorrect would be of 2 levels; 0/1/2 would be 3 levels

weight	If weight = TRUE, the weighted frequencies will be computed and then be utilized for the chi-square scores ; If weight = F, returning the chisquare scores computed from the raw feature frequencies.
ctable	If ctable = TRUE, returning the contingency tables instead of the chi-square scores.

Value

This function returns a data frame with ranked chi-scores or contingency tables for each feature.

To get the weighted frequencies solely, please run WeightedFeatures() in LOGAN package.

References

He Q., von Davier M. (2015) Identifying Feature Sequences from Process Data in Problem-Solving Items with N-Grams. In: van der Ark L., Bolt D., Wang WC., Douglas J., Chow SM. (eds) Quantitative Psychology Research. Springer Proceedings in Mathematics & Statistics, vol 140. Springer, Cham. https://doi-org.ezproxy.uio.no/10.1007/978-3-319-19977-1_13

Examples

```
ComputeChisquared(data = cp025q01.wgt[,c(7:13,15)],
outcome = "outcome", level = 2, weight = FALSE, ctable = FALSE)

ComputeChisquared(data = training[,7:14],
outcome = "outcome", level = 2, weight = FALSE, ctable = TRUE)
```

`cp025q01.features` *Data for PISA 2012, CP025, Q01 (selected countries)*

Description

A dataset containing the original features generated from 2012 PISA Climate Control CP025Q01 task

Usage

```
cp025q01.features
```

Format

A data frame with 1456 rows and 16 variables.

Source

https://www.frontiersin.org/articles/10.3389/fpsyg.2019.02461/full

cp025q01.wgt — *Treated data for PISA 2012, CP025, Q01 (selected countries)*

Description

A dataset containing the weighted features generated from 2012 PISA Climate Control CP025Q01 task

Usage

```
cp025q01.wgt
```

Format

A data frame with 1456 rows and 15 variables.

Source

https://www.frontiersin.org/articles/10.3389/fpsyg.2019.02461/full

DataPartition — *Data Partition*

Description

Data Partition

Usage

```
DataPartition(data = NULL, outcome = NULL, proportion = 0.7, seed = 2022)
```

Arguments

`data`	A `data.frame` that contains the study's features and the outcome variable.
`outcome`	A character string with the name of the outcome variable from the data.
`proportion`	A numeric value for the proportion of data to be put into model training. Default is set to 0.7.
`seed`	A numeric value for set.seed. It is set to be 2022 by default.

Value

This function returns a list with training and testing data sets using a stratified selection by the outcome variable as performed by the createDataPartition function from the caret package.

Examples

```
dp <- DataPartition(data = cp025q01.wgt, outcome = "outcome")
```

DtResult — *Decision Tree Result in Text View and Plot*

Description

Decision Tree Result in Text View and Plot

Usage

```
DtResult(ModelObject)
```

Arguments

`ModelObject`	A fitted model object from TreeModels() or TreeModelsAllSteps() functions.

Value

This function returns the structure of the decision tree final model as a text view, and a plot of the rpart model object as displayed by the rpart.plot package.

Examples

```
colnames(training)[14] <- "perf"
ensemblist <- TreeModels(traindata = training,
methodlist = "dt",checkprogress = TRUE)

DtResult(ensemblist$ModelObject)
```

LOGANTree — *LOGANTree: Tree-based models for the analysis of log files from computer-based assessments*

Description

This package enables users to model log-file data from computer-based assessments using machine-learning techniques. It allows researchers to generate new knowledge by comparing the performance of three tree-based classification models (i.e., decision trees, random forest, and gradient boosting) to predict student's outcome. It also contains a set of handful functions for the analysis of the features' influence on the modeling. Data from the Climate control item from the 2012 Programme for International Student Assessment (PISA, <https://www.oecd.org/pisa/>) is available for an illustration of the package's capability. An application of the package functions for a math item in PISA 2012 is described in Qin (2022).

LOGANTree functions

The LOGANTree functions can be categorized in two types: (a) tree-based modeling and (b) features' analysis. While the first one provides tools for the specification and the evaluation of the three classification models, the second category is devoted to a careful analysis of the data features and their influence on the model's results. We use the caret package to perform most of the analyses and we provide summary reports and data visualization tools to better compare the three classifiers.

What follows is a list of functions organized per category:

Tree-based modeling:

- TreeModels
- DataPartition
- TreeModelsAllSteps
- PerformanceMatrics
- RocPlot

Features' analysis:

- NearZeroVariance
- DtResult(
- VariableImportanceTable
- VariableImportancePlot
- ChisquareTable
- ChisquarePlot
- PartialDependencePlot

Author(s)

- Qi Qin [aut, cre],
- Denise Reis Costa [aut, ths]

References

Qin, Q. (2022). Application of tree-based data mining techniques to examine log file data from a 2012 PISA computer-based Mathematics item. [Unpublished thesis]. University of Oslo.

`NearZeroVariance` *Flag the features that have (near) zero variance*

Description

Flag the features that have (near) zero variance

Usage

```
NearZeroVariance(data)
```

Arguments

`data`	A dataset containing the study's features.

Value

This function returns a dataframe with feature names and their frequency ratio, percentage of the unique value and logic values indicating whether the feature is zero variance or has near zero variance.

feature : name of the features.

flag.zv (Flag Zero Variance) : True/False, flagging zero variance.

fr (Frequency Ratio) : the ratio of the value with the highest frequency over the value with the second highest frequency.

puv (Percentage of Unique Values) : number of the unique values divided by the total number of samples.

flag.nzv (Flag Near Zero Variance) : True/False, flagging near zero variance.

References

Boehmke, B., & Greenwell, B. M. (2019). Hands-on machine learning with R. CRC Press.p.52-55. https://doi-org.ezproxy.uio.no/10.1201/9780367816377

Examples

```
NearZeroVariance(training)
```

PartialDependencePlot *Partial Dependence Plot*

Description

Partial Dependence Plot

Usage

```
PartialDependencePlot(
  data = NULL,
  FeatureNames = NULL,
  FittedModelObject = NULL,
  j = 20
)
```

Arguments

data	A data.frame that contains the study's features and the outcome.
FeatureNames	A vector with the names of features to plot.
FittedModelObject	A fitted model object.
j	A numerical value that indicates the size of the equally spaced values for the feature of interest.

Value

This function returns a plot where X axis presents the values for each feature and Y axis illustrates the predicted proportion of correct answer to the item.

Examples

```
colnames(training)[14] <- "perf"
ensemblist <- TreeModels(traindata = training,
methodlist = c("dt","rf"),checkprogress = TRUE)

PartialDependencePlot(data = training,
FeatureNames = colnames(training[-c(4,14)]),
FittedModelObject = ensemblist$ModelObject$rpart, j = 30)

PartialDependencePlot(data = training,
FeatureNames = colnames(training[-c(4,14)]),
FittedModelObject = ensemblist$ModelObject$ranger, j = 20)
```

PerformanceMetrics *Report table with the performance metrics for tree-based learning methods*

Description

Report table with the performance metrics for tree-based learning methods

Usage

```
PerformanceMetrics(
  testdata,
  DT = NULL,
  RF = NULL,
  GBM = NULL,
  outcome,
  reflevel
)
```

Arguments

testdata	A test dataset that contains the study's features and the outcome variable.
DT	A fitted decision tree model object
RF	A fitted random forest model object
GBM	A fitted gradient boosting model object
outcome	A factor variable with the outcome levels.
reflevel	A character string with the quoted reference level of outcome.

Value

This function returns a `data.frame` with a table that compares five performance metrics from different tree-based machine learning methods. The metrics are: Accuracy, Kappa, Sensitivity, Specificity, and Precision. The results are derived from the confusionMatrix function from the caret package.

Examples

```
colnames(training)[14] <- "perf"
ensemblist <- TreeModels(traindata = training,
methodlist = c("dt", "rf","gbm"),checkprogress = TRUE)

PerformanceMetrics(testdata = testing, RF = ensemblist$ModelObject$ranger,
outcome = "outcome", reflevel = "correct")

PerformanceMetrics(testdata = testing, RF = ensemblist$ModelObject$ranger,
GBM = ensemblist$ModelObject$gbm,
outcome = "outcome", reflevel = "correct")
```

```
PerformanceMetrics(testdata = testing, DT = ensemblist$ModelObject$rpart,
RF = ensemblist$ModelObject$ranger, GBM = ensemblist$ModelObject$gbm,
outcome = "outcome", reflevel = "correct")
```

`RocPlot` *ROC Curves Plot*

Description

ROC Curves Plot

Usage

```
RocPlot(ModelObject, testdata, outcome, reflevel)
```

Arguments

`ModelObject`	An object obtained from TreeModels() or TreeModelsAllSteps() functions.
`testdata`	A testing dataset.
`outcome`	A character string with the name of the binary outcome variable.
`reflevel`	A character string with the quoted reference level of outcome.

Value

This function returns a plot with ROC curves for the selected tree-based models (i.e., decision tree, random forest, or gradient boosting).

Examples

```
colnames(training)[14] <- "perf"
colnames(testing)[14] <- "perf"
ensemblist <- TreeModels(traindata = training,
methodlist = c("dt", "gbm","rf"),checkprogress = TRUE)

RocPlot(ModelObject = ensemblist$ModelObject, testdata = testing,
outcome = "perf", reflevel = "incorrect")
```

`testing` *PISA 2012, CP025, Q01 (selected countries) Testing Data Set*

Description

A testing set partitioned from the cp025q01.wgt dataset with 30

Usage

```
testing
```

Format

A data frame with 436 rows and 14 variables.

Source

https://www.frontiersin.org/articles/10.3389/fpsyg.2019.02461/full

`training` *PISA 2012, CP025, Q01 (selected countries) Training Data Set*

Description

A training set partitioned from the cp025q01.wgt dataset with 70

Usage

```
training
```

Format

A data frame with 1020 rows and 14 variables.

Source

https://www.frontiersin.org/articles/10.3389/fpsyg.2019.02461/full

---

`TreeModels` *Tree-based Model Training*

Description

Tree-based Model Training

Usage

```
TreeModels(
  traindata = NULL,
  seed = 2022,
  methodlist = c("dt", "rf", "gbm"),
  iternumber = 10,
  dt.gridsearch = NULL,
  rf.gridsearch = NULL,
  gbm.gridsearch = NULL,
  checkprogress = FALSE
)
```

Arguments

`traindata`	A `data.frame` with the training data set. Please name the outcome variable as "perf".
`seed`	A numeric value for set.seed. It is set to be 2022 by default.
`methodlist`	A list of the tree-based methods to model. The default is methodlist = c("dt", "rf", "gbm").
`iternumber`	Number of resampling iterations/Number of folds for the cross-validation scheme.
`dt.gridsearch`	A `data.frame` of the tuning grid, which allows for specifying parameters for decision tree model.
`rf.gridsearch`	A `data.frame` of the tuning grid, which allows for specifying parameters for random forest model.
`gbm.gridsearch`	A `data.frame` of the tuning grid, which allows for specifying parameters for gradient boosting model.
`checkprogress`	Logical. Print the modeling progress if it is TRUE. The default is FALSE.

Details

This function performs the modeling step of a predictive analysis. The selected classifiers are used for modeling the provided training dataset under a cross-validation scheme. Users have the possibility to choose which model they want to compare by specifying it on the `methodlist` argument. The caretEnsemble package is used in the modeling process to ensure that all models follow the same resampling procedures. ROC is used to select the optimal model for each tree-based method using the largest value. Finally, a summary report is displayed.

Value

This function returns two lists:

ModelObject An object with results from selected models

SummaryReport A `data.frame` with the summary of model parameters. The summary report is shown automatically in the output.

Examples

```
colnames(training)[14] <- "perf"
ensemblist <- TreeModels(traindata = training,
methodlist = c("rf","gbm","dt"),checkprogress = TRUE)

ensemblist <- TreeModels(traindata = training,
methodlist = c("rf"),
rf.gridsearch = data.frame(mtry = 2, splitrule = "gini", min.node.size = 1))
```

TreeModelsAllSteps *Data Partition and Tree-based Model Training*

Description

Data Partition and Tree-based Model Training

Usage

```
TreeModelsAllSteps(
  data = NULL,
  proportion = 0.7,
  seed = 2022,
  methodlist = c("dt", "rf", "gbm"),
  iternumber = 10,
  dt.gridsearch = NULL,
  rf.gridsearch = NULL,
  gbm.gridsearch = NULL,
  checkprogress = FALSE
)
```

Arguments

`data`	A `data.frame` that contains the study's features and the outcome variable. Please name the outcome variable as "perf".
`proportion`	A numeric value for the proportion of data to be put into model training. Default is set to 0.7.
`seed`	A numeric value for set.seed. It is set to be 2022 by default.

methodlist	A list of the tree-based methods to model. The default is methodlist = c("dt", "rf", "gbm").
iternumber	A numeric value for the number of resampling iterations/number of folds for the cross-validation scheme.
dt.gridsearch	A data.frame of the tuning grid, which allows for specifying parameters for decision tree model.
rf.gridsearch	A data.frame of the tuning grid, which allows for specifying parameters for random forest model.
gbm.gridsearch	A data.frame of the tuning grid, which allows for specifying parameters for gradient boosting model.
checkprogress	Logical. Print the modeling progress if it is TRUE. The default is FALSE.

Details

This function performs all the steps of a predictive analysis. First, the data is partitioned in the training and testing datasets using a stratified selection by the outcome variable as performed by the createDataPartition function from the caret package. Then, the selected classifiers are used for modeling the training dataset under a cross-validation scheme. Users have the possibility to choose which model they want to compare by specifying it on the methodlist argument. The caretEnsemble package is used in the modeling process to ensure that all models follow the same resampling procedures. ROC is used to select the optimal model for each tree-based method using the largest value. Finally, a summary report is displayed.

Value

This function returns three lists:

DataPartition The partitioned datasets: training (cv_train) and testing (cv_test).

ModelObject An object with results from selected models

SummaryReport A data.frame with the summary of model parameters. The summary report is shown automatically in the output.

Examples

```
cp025q01.wgt <- cp025q01.wgt[,-14]
colnames(cp025q01.wgt)[14] <- "perf"

ensemblist <- TreeModelsAllSteps(data = cp025q01.wgt,
checkprogress = TRUE)

ensemblist <- TreeModelsAllSteps(data = cp025q01.wgt,
methodlist = c("dt", "gbm"), checkprogress = TRUE)

ensemblist <- TreeModelsAllSteps(data = cp025q01.wgt,
methodlist = c("rf"),
rf.gridsearch = data.frame(mtry = 2, splitrule = "gini", min.node.size = 1),
checkprogress = TRUE)
```

`VariableImportancePlot` *Barplot comparing the feature importance across different learning methods.*

Description

Barplot comparing the feature importance across different learning methods.

Usage

```
VariableImportancePlot(DT = NULL, RF = NULL, GBM = NULL)
```

Arguments

DT	A fitted decision tree model object
RF	A fitted random forest model object
GBM	A fitted gradient boosting model object

Value

This function returns a barplot that compares the standardized feature importance across different tree-based machine learning methods. These measures are computed via the caret package.

Examples

```
library(gbm)
colnames(training)[14] <- "perf"
ensemblist <- TreeModels(traindata = training,
methodlist = c("dt", "rf","gbm"),checkprogress = TRUE)

VariableImportancePlot(DT = ensemblist$ModelObject$rpart,
RF = ensemblist$ModelObject$ranger,GBM = ensemblist$ModelObject$gbm)

VariableImportancePlot(RF = ensemblist$ModelObject$ranger,
GBM = ensemblist$ModelObject$gbm)

VariableImportancePlot(DT = ensemblist$ModelObject$rpart)
```

VariableImportanceTable *Table comparing the feature importance for tree-based learning methods.*

Description

Table comparing the feature importance for tree-based learning methods.

Usage

```
VariableImportanceTable(DT = NULL, RF = NULL, GBM = NULL)
```

Arguments

DT	A fitted decision tree model object
RF	A fitted random forest model object
GBM	A fitted gradient boosting model object

Value

This function returns a data frame that compares the feature importance from different tree-based machine learning methods. These measures are computed via the caret package.

Examples

```
library(gbm)
colnames(training)[14] <- "perf"
ensemblist <- TreeModels(traindata = training,
methodlist = c("dt", "rf","gbm"),checkprogress = TRUE)

VariableImportanceTable(DT = ensemblist$ModelObject$rpart,
RF = ensemblist$ModelObject$ranger,GBM = ensemblist$ModelObject$gbm)

VariableImportanceTable(DT = ensemblist$ModelObject$rpart,
RF = ensemblist$ModelObject$ranger)

VariableImportanceTable(DT = ensemblist$ModelObject$rpart)
```

Index