Type Package
Title Tree-Based Models for the Analysis of Log Files from Computer-Based Assessments
Version 0.1.0
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 <qiq@student.uv.uio.no>
Repository CRAN
Date/Publication 2022-03-29 19:20:05 UTC

R topics documented:

ChiSquarePlot .................................................. 2
ChiSquareTable .................................................. 3
ChiSquarePlot

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).
ChiSquareTable

References


Examples

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

ChiSquarePlot(trainingdata = training,
featureNames = 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,
 featureNames = NULL,
 outcome = NULL,
 level = NULL,
 ModelObject = NULL
)

Arguments

trainingdata A data set used for training
featureNames 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
ComputeChisquared

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


Examples

```r
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

<table>
<thead>
<tr>
<th>data</th>
<th>A dataset containing an outcome variable and action features with either raw frequencies or weighted frequencies.</th>
</tr>
</thead>
<tbody>
<tr>
<td>outcome</td>
<td>Name of the outcome variable.</td>
</tr>
<tr>
<td>level</td>
<td>The level of outcome. e.g. correct/incorrect would be of 2 levels; 0/1/2 would be 3 levels</td>
</tr>
</tbody>
</table>
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


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)

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

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


DataPartition

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**  
```r
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**

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

---

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

**Description**  
Flag the features that have (near) zero variance

**Usage**  
```r
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


Examples

NearZeroVariance(training)

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.
PerformanceMetrics

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

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

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

PartialDependencePlot(data = training,
FeatureNames = colnames(training[-c(4,14)]),
FittedModelObject = ensembi$ModelObject$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**

```r
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"
ensamblist <- 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, reflevel)

Arguments

ModelObject An object obtained from TreeModels() or TreeModelsAllSteps() functions.
testdata A testing dataset.
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

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

RocPlot(ModelObject = ensemblist$ModelObject, testdata = testing, 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**


---

**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**

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

```r
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))
```

Description

Data Partition and Tree-based Model Training

Usage

```r
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**

```r
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**

```r
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

```r
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

* datasets
  cp025q01.features, 5
  cp025q01.wgt, 6
  testing, 11
  training, 11

ChiSquarePlot, 2
ChiSquareTable, 3
ComputeChisquared, 4
cp025q01.features, 5
cp025q01.wgt, 6

DataPartition, 6
DtResult, 7

NearZeroVariance, 7

PartialDependencePlot, 8
PerformanceMetrics, 9

RocPlot, 10

testing, 11
training, 11
TreeModels, 12
TreeModelsAllSteps, 13

VariableImportancePlot, 15
VariableImportanceTable, 16