Package 'mobForest'

July 2, 2014

Type Package

Title Model based Random Forest analysis

Version 1.2

Date 2013-01-28

Author Nikhil Garge, Barry Eggleston and Georgiy Bobashev

Maintainer Nikhil Garge <ngarge@rti.org>

Description This package implements random forest method for model based recursive partitioning. The mob() function, developed by Zeileis et al (2008), within party package, is modified to construct model-based decision trees based on random forests methodology. The main input function mobForestAnalysis() takes all input parameters to construct trees, compute out-of-bag errors, predictions, and overall accuracy of forest. The algorithm performs parallel computation using clusterApply() function within 'parallel' package.

License GPL (>= 2)

LazyLoad yes

Depends parallel, party, lattice

Imports methods, modeltools

Suggests mlbench

Repository CRAN

Date/Publication 2013-01-29 23:13:15

NeedsCompilation no

R topics documented:

mobForest-package . . . 2
computeAcc . . . 3
computeMSE . . . 4
computeR2 . . . 5
getPredictedValues . . . 5
getPredictedValues-methods . . . 6
getVarimp . . . 6
logistic_accuracy . . . 7
mobForestAnalysis . . . 8
mobForestControl-class . . . 10
mobForestOutput-class . . . 11
mobForest_control . . . 12
mobForest_output . . . 13
mob_RF_Tree . . . 14
predictionOutput-class . . . 15
prediction_output . . . 16
PredictiveAccuracy . . . 17
PredictiveAccuracy-methods . . . 18
print.predAccuracyEstimates . . . 18
residualPlot . . . 19
stringFormula . . . 20
treePredictions . . . 20
varimplot . . . 21
varimpOutput-class . . . 22
varimp_output . . . 22

Index **24**

`mobForest-package` *Random Forest methodology for model-based recursive partitioning*

Description

This package implements random forest method for model based recursive partitioning.

Details

Package:	mobForest
Type:	Package
Version:	1.2
Date:	2013-01-28
License:	GPL (>= 2)
LazyLoad:	yes
Depends:	parallel, party

Author(s)

Nikhil Garge, Barry Eggleston, Georgiy Bobashev Maintainer: Nikhil Garge <ngarge@rti.org>

References

Achim Zeileis, Torsten Hothorn, and Kurt Hornik (2008). Model-Based Recursive Partitioning. *Journal of Computational and Graphical Statistics*, 17(2), 492-514.

Hothorn, T., Hornik, K. and Zeileis, A. (2006) Unbiased recursive partitioning: A conditional inference framework, *J Comput Graph Stat*, 15, 651-674.

Strobl, C., Malley, J. and Tutz, G. (2009) An introduction to recursive partitioning: rationale, application, and characteristics of classification and regression trees, bagging, and random forests, *Psychol Methods*, 14, 323-348.

Examples

```
set.seed(290875)
if(require("mlbench")) {
## Random Forest analysis of model based recursive partitioning

## load data
data("BostonHousing", package = "mlbench")

## Recursive partitioning based on linear regression model medv ~ lstat with 3 trees.
## 1 core/processor used. Supply more processors using 'processors' argument
rfout <- mobForestAnalysis(as.formula(medv ~ lstat),
c("rad", "tax", "crim"), mobForest.controls =
mobForest_control(ntree = 3, mtry = 2, replace = TRUE,
alpha = 0.05, bonferroni = TRUE, minsplit = 120),
 data = BostonHousing, processors = 1, model = linearModel)

 rfout ## should produce descriptive output for mobForestOutput object
}
```

`computeAcc` *Predictive accuracy estimates across trees for logistic regression model*

Description

Computes predictive accuracy of response variable with binary outcome. The function takes observed and predicted binary responses as input arguments and computes proportion of observations classified in same group.

Usage

```
computeAcc(response, predictions, prob.cutoff)
```

Arguments

response	a vector of binary outcome
predictions	a matrix of predicted probabilities (logit model) for out-of-bag observations for each tree
prob.cutoff	Predicted probabilities converted into classes (Yes/No, 1/0) based on this probability threshold.

Value

Predictive accuracy estimates ranging between 0 and 1 for each tree. Zero represents worst accuracy and 1 suggests best accuracy.

---

computeMSE *Predictive accuracy estimates (MSE) across trees for linear or poisson regression model.*

Description

Predictive accuracy estimates (MSE) across trees for linear or poisson regression model.

Usage

```
computeMSE(response, predictions)
```

Arguments

response	a vector of actual response of outcome variable.
predictions	a vector of predicted response for the same outcome variable.

Value

MSE estimates

computeR2 *Predictive accuracy estimates across trees for linear or poisson regression model*

Description

pseudo R-square (R2) computation - proportion of total variance in response variable explained by the tree model. The function takes observed and predicted responses as input arguments and computes pseudo-R2 to determine how well the tree model fits the given data.

Usage

```
computeR2(response, predictions)
```

Arguments

response	a vector of actual response of outcome variable.
predictions	a vector of predicted response for the same outcome variable.

Value

Predictive accuracy estimates ranging between 0 and 1. Zero represents worst accuracy and 1 suggests best accuracy.

getPredictedValues *Predictions*

Description

Get predictions summarized across trees for out-of-bag cases or all cases or cases from new test data.

Usage

```
getPredictedValues(object, OOB = TRUE, newdata = FALSE)
```

Arguments

object	An object of class mobForestOutput
OOB	a logical determining whether to return predictions from the out-of-bag sample or the learning sample (not suggested).
newdata	a logical determining whether to return predictions from test data. If newdata = TRUE, then OOB argument is ignored.

Value

matrix with three columns: 1) Mean Predictions across trees, 2) Standard deviation of predcitions across trees, and 3) Residual (mean predicted - observed). The third column is applicable only when linear regression is considered as the node model.

Examples

```
## Random Forest analysis of model based recursive partitioning

## load data
data("BostonHousing", package = "mlbench")
BostonHousing <- BostonHousing[1:90,c("rad", "tax", "crim", "medv", "lstat")]

## Recursive partitioning based on linear regression model medv ~ lstat with 3 trees.
## 1 core/processor used. Supply more processors using 'processors' argument
rfout <- mobForestAnalysis(as.formula(medv ~ lstat),
c("rad", "tax", "crim"), mobForest.controls =
mobForest_control(ntree = 3, mtry = 2, replace = TRUE,
alpha = 0.05, bonferroni = TRUE, minsplit = 25),
data = BostonHousing, processors = 1, model = linearModel)

## Obtain out-of-bag predicted values
OOB.predMat <- getPredictedValues(rfout, OOB = TRUE)
OOB.pred = OOB.predMat[,1]
```

---

getPredictedValues-methods

Get fitted values from mobForestOutput object

Description

~~ Methods for function `getPredictedValues` ~~

Methods

```
signature(object = "mobForestOutput")
```

---

getVarimp

Variable Importance

Description

Variable importance scores computed through random forest analysis

Usage

```
getVarimp(object)
```

Arguments

object	An object of class mobForestOutput returned by mobForestAnalysis()

References

Leo Breiman (2001). Random Forests. *Machine Learning*, 45(1), 5-32.

Examples

```
if(require("mlbench")) {

## recursive partitioning of a linear regression model

## load data
data("BostonHousing", package = "mlbench")
BostonHousing <- BostonHousing[1:90,c("rad", "tax", "crim", "medv", "lstat")]

## Recursive partitioning based on linear regression model
## medv ~ lstat with 3 trees.
## 1 core/processor used. Supply more processors using
## 'processors' argument
rfout <- mobForestAnalysis(as.formula(medv ~ lstat),
c("rad", "tax", "crim"), mobForest.controls =
mobForest_control(ntree = 3, mtry = 2, replace = TRUE,
alpha = 0.05, bonferroni = TRUE, minsplit = 25),
data = BostonHousing, processors = 1, model = linearModel)

## Returns a vector of variable importance scores
getVarimp(rfout)

}
```

---

logistic_accuracy *Contingency table: Predictive Vs Observed outcome*

Description

This function takes predicted probabilities (for out of bag cases) obtained through logistic regression-based tree models and converts them into binary classes (based on specified probability threshold). The predicted classifications are then compared to actual binary response.

Usage

```
logistic_accuracy(response, predicted, prob.thresh)
```

Arguments

response	A vector of binary classes of out-of-cases for a given tree
predicted	A vector of predicted probabilities of out-of-cases using same tree
prob.thresh	Probibility threshold for classification. It is 0.5 by default.

---

mobForestAnalysis *Model-based Random Forest Analysis*

Description

Main function that takes all the neccessary arguments to start model-based random forest analysis.

Usage

```
mobForestAnalysis(formula, partitionVariables, data, mobForest.controls = mobForest_control(), newTes
```

Arguments

formula	An object of class formula specifying the model. This should be of type y ~ x1 + ... + xk, where the variables x1, x2, ..., xk are predictor variables and y represents an outcome variable. This model is referred to as the node model
partitionVariables	A character vector specifying the partition variables
data	An input dataset that is used for constructing trees in random forest.
mobForest.controls	An object of class "mobForestControl" returned by mobForest_control(), that contains parameters controlling the construction of random forest.
newTestData	A data frame representing test data for validating random forest model. This data is not used in in tree building process.
processors	A number of processors/cores on your computer that should be used for parallel computation.
model	A model of class "StatModel" used for fitting observations in current node. This parameter allows fitting a linear model or generalized linear model with formula y ~ x1 + ... + xk. The Parameter "linearModel" fits linear model. The parameter "glinearModel" fits Poisson or logistic regression model depending upon the specification of parameter "family" (explained next). If "family" is specified as binomial() then logistic regression is performed. If the "family" is specified as poisson() then Poisson regression is performed.
family	A description of error distribution and link function to be used in the model. This parameter needs to be specified if generalized linear model is considered. The parameter "binomial()" is to be specified when logistic regression is considered and "poisson()" when Poisson regression is considered as the node model. The values allowed for this parameter are binomial() and poisson().

`prob.cutoff` In case of logistic regression as a node model, the predicted probabilities for OOB cases are converted into classes (yes/no, high/low, etc as specified) based on this probability cutoff. If logistic regression is not cosndiered as nodel model, the prob.cutoff = NULL. By default it is 0.5 when parameter not specieid (and logistic regression cponsidered).

Details

`mobForestAnalysis` is the main function that takes all the input parameters - model, partition variables, and forest control parameters - and starts the model-based random forest analysis. `mobForestAnalysis` calls `bootstrap` function which constructs decesion trees, computes out-of-bag (OOB) predictions, OOB predictive accuracy and perturbation in OOB predictive accuracy through permutation. `bootstrap` constructs trees on multiple cores/processors simultaneously through parallel computation. Later, the `getmobForestObject` function wraps the analysis output into `mobForestOutput` object.

Predictive accuracy estimates are computed using pseudo-R2 metric, defined as the proportion of total variation in outcome variable explained by a tree model on out-of-bag cases. R2 ranges from 0 to 1. R2 of zero suggests worst tree model (in terms of predicting outcome) and R2 of 1 suggests perfect tree model.

Value

An object of class `mobForestOutput`.

References

Achim Zeileis, Torsten Hothorn, and Kurt Hornik (2008). Model-Based Recursive Partitioning. *Journal of Computational and Graphical Statistics*, 17(2), 492-514.

Hothorn, T., Hornik, K. and Zeileis, A. (2006) Unbiased recursive partitioning: A conditional inference framework, *J Comput Graph Stat*, 15, 651-674.

Strobl, C., Malley, J. and Tutz, G. (2009) An introduction to recursive partitioning: rationale, application, and characteristics of classification and regression trees, bagging, and random forests, *Psychol Methods*, 14, 323-348.

See Also

mobForest_control(), `mobForestOutput-class`

Examples

```
set.seed(290875)

if(require("mlbench")) {
```

```
## Random Forest analysis of model based recursive partitioning
## load data
data("BostonHousing", package = "mlbench")
BostonHousing <- BostonHousing[1:90,c("rad", "tax", "crim", "medv", "lstat")]

## Recursive partitioning based on linear regression model medv ~ lstat with 3 trees.
## 1 core/processor used. Supply more processors using 'processors' argument
rfout <- mobForestAnalysis(as.formula(medv ~ lstat),
c("rad", "tax", "crim"), mobForest.controls =
mobForest_control(ntree = 3, mtry = 2, replace = TRUE,
alpha = 0.05, bonferroni = TRUE, minsplit = 25),
data = BostonHousing, processors = 1, model = linearModel)


rfout ## should produce descriptive output for mobForestOutput object


}
```

`mobForestControl-class`

Class `"mobForestControl"`

Description

Control parameters for random forest

Objects from the Class

Objects can be created by `mobForest_control`.

Slots

`ntree`: number fo trees in forest

`mtry`: random subset of partition variables to be considered at each node of decesion tree

`replace`: sampling with replacement or without replacement

`fraction`: proportion of observations to be sampled without replacement

`mob.control`: An object of class `mob_control`, which can be obtained using `mobForest_control`

Methods

No methods defined with class "mobForestControl" in the signature.

Examples

```
#showClass("mobForestControl")
## The following code creates following forest controls: 400 trees to be
## constructed, sampling with replacement, a node contains at least 200 observations
mobForest.controls = mobForest_control(ntree = 400, mtry = 4, replace = TRUE,
minsplit = 200)
```

mobForestOutput-class *Class* "mobForestOutput"

Description

Random Forest output for model based recursive partitioning

Objects from the Class

Objects can be created by mobForest_output.

Slots

oobPredictions: Object of class "predictionOutput" for out-of-bag cases.

GeneralPredictions: Object of class "predictionOutput" for all cases.

NewDataPredictions: Object of class "predictionOutput" for new Test dataset cases.

VarimpObject: Object of class "varimpOutput" ~~

modelUsed: the model considered during random forest analysis.

fam: error distribution assumption

train.response: data.frame of predictions for training set

new.response: data.frame of predictions for new test dataset

Methods

getPredictedValues signature(object = "mobForestOutput", OOB = "logical", newdata = "logical"): get predictions summarized across trees for OOB cases or learning data or new test data.

getVarimp signature(object = "mobForestOutput"): get variable importance scores computed through random forest analysis

PredictiveAccuracy signature(object = "mobForestOutput", newdata = "logical", plot = "logical"): predictive performance across all trees

residualPlot signature(object = "mobForestOutput"): produces two plots on same panel: a) histogram of residuals, b) predicted Vs residuals

show signature(object = "mobForestOutput"): print mobForestOutput object

varimplot signature(object = "mobForestOutput"): produces a plot with variable importance scores on X-axis and variable names on Y-axis.

See Also

predictionOutput, varimpOutput

Examples

```
if(require("mlbench")) {

## Random Forest analysis of model based recursive partitioning
## load data
data("BostonHousing", package = "mlbench")

## Recursive partitioning based on linear regression model medv ~ lstat with 3 trees.
## 1 core/processor used. Supply more processors using 'processors' argument
rfout <- mobForestAnalysis(as.formula(medv ~ lstat),
c("rad", "tax", "crim"), mobForest.controls =
mobForest_control(ntree = 3, mtry = 2, replace = TRUE,
alpha = 0.05, bonferroni = TRUE, minsplit = 120),
data = BostonHousing, processors = 1, model = linearModel)

## print method for mobForestOutput object
show(rfout)
}
```

`mobForest_control` *Control parameters for random forest*

Description

Various parameters that control the forest growing.

Usage

```
mobForest_control(ntree = 300, mtry = 0, replace = FALSE, fraction = 0.632,
alpha = 1, bonferroni = FALSE, minsplit = 20, trim = 0.1, objfun = deviance,
breakties = FALSE, parm = NULL, verbose = FALSE)
```

Arguments

`ntree`	Number of trees to be constructed in forest (default = 300)
`mtry`	number of input variables randomly sampled as candidates at each node
`replace`	TRUE or FALSE. replace = TRUE (default) performs bootstrapping. replace = FALSE performs sampling without replacement.
`fraction`	number of observations to draw without replacement (only relevant if replace = FALSE)
`alpha`	A node is considered for splitting if the p value for any partitioning variable in that node falls below alpha (default 0.05) as mentioned in mob_control()
`bonferroni`	logical, Should p values be Bonferroni corrected? (default TRUE) as mentioned in mob_control()

minsplit	integer, The minimum number of observations in a node (default 20) as mentioned in mob_control()
trim	numeric, as defined in mob_control()
objfun	function, as defined in mob_control()
breakties	logical, as defined in mob_control()
parm	numeric or vector, as defined in mob_control()
verbose	logical, as defined in mob_control()

Details

This function is used to set up forest controls. The mob_control (from party 'package') object is used to set up control parameters for single tree model.

Value

An object of class mobForestControl.

References

Achim Zeileis, Torsten Hothorn, and Kurt Hornik (2008). Model-Based Recursive Partitioning. *Journal of Computational and Graphical Statistics*, 17(2), 492-514.

Examples

```
## create forest controls before starting random forest analysis
mobForest.controls = mobForest_control(ntree = 400, mtry = 4,
replace = TRUE, minsplit = 200)
```

---

mobForest_output *Model-based Random forest Object*

Description

Random Forest Output object that stores all the results including predictions, variable importance matrix, model, family of error distributions, and observed responses.

Usage

```
mobForest_output(oobPredictions, GeneralPredictions, NewDataPredictions,
VarimpObject, modelUsed, fam, train.response,
new.response = data.frame(matrix(0, 0, 0)))
```

Arguments

oobPredictions	Predictions on Out-of-bag data
GeneralPredictions	Predictions on learning data
NewDataPredictions	Predictions on new test data
VarimpObject	variable importance object
modelUsed	Model used
fam	a description of the error distribution and link function to be used in the model.
train.response	Response outcome of training data
new.response	Response outcome of test data

See Also

predictionOutput, varimpOutput

---

mob_RF_Tree *model based recursive partitioning - randomized subset of partition variables considered during each split.*

Description

model based recursive partitioning - randomized subset of partition variables considered during each split.

Usage

```
mob_RF_Tree(mainModel, partitionVars, mtry, weights, data = list(),
na.action = na.omit, model = glinearModel, control = mob_control(), ...)
```

Arguments

mainModel	A model in character format
partitionVars	A vector of partition variables
mtry	A Random subset of partition variables to be considered at each node of decesion tree
weights	An optional vector of weights, as described in mob
data	A data frame containing the variables in the model.
na.action	A function which indicates what should happen when the data contain NAs, as described in mob
model	A model of class StatModel
control	A list with control parameters as returned by mob_control
...	Additional arguments passed to the fit call for the model.

Details

The mob function in party package is modified so that a random subset of predictor variables are considered during each split. mtry represents the number of predictor variables to be considered during each split.

Value

An object of class mob inheriting from BinaryTree. Every node of the tree is additionally associated with a fitted model.

References

Achim Zeileis, Torsten Hothorn, and Kurt Hornik (2008). Model-Based Recursive Partitioning. *Journal of Computational and Graphical Statistics*, 17(2), 492-514.

---

predictionOutput-class

Class "predictionOutput"

Description

The object of this class stores predictions and predictive accuracy estimates.

Objects from the Class

Objects can be created by calls of the form prediction_output.

Slots

predMat: a matrix containing three columns: mean and standard deviation of predictions across trees and residuals (observed - predicted)

R2: a vector of predictive accuracy estimates (ranging between 0 and 1) for each tree

mse: a vector of mean-square-estimates (MSE) for each tree. Valid only if the outcome is continuous.

overallR2: Overall predictive accuracy estimate obtained by combining predictions across trees.

predType: character specifying the type of prediction output: out-of-bag cases, learning set, or new Test dataset

Methods

No methods defined with class "predictionOutput" in the signature.

See Also

prediction_output, PredictiveAccuracy

`prediction_output` *Predictions and predictive accuracy*

Description

Predictions and predictive accuracy estimates

Usage

```
prediction_output(predMean = numeric(), predSd = numeric(),
residual = numeric(), R2 = numeric(), mse = numeric(),
overallR2 = numeric(), predType = character())
```

Arguments

`predMean`	Mean predictions across trees.
`predSd`	Standard deviation predictions across trees.
`residual`	Residuals (predicted outcome - observed outcome).
`R2`	Predictive accuracy across trees
`mse`	MSE across trees
`overallR2`	Overall R2
`predType`	Out-of-bag data or test data or learning data.

Details

This function takes predictions and predictive accuracy estimates as input arguments and creates objects of class `predictionOutput`.

Value

An object of class `predictionOutput`.

See Also

`predictionOutput`, `mobForestAnalysis`

PredictiveAccuracy *Predictive Performance*

Description

Predictive performance across all trees

Usage

```
PredictiveAccuracy(object, newdata = FALSE, prob.cutoff = NULL, plot = TRUE)
```

Arguments

object	An object of class mobForestOutput
newdata	A logical value specifying if the performance needs to be summarized on test data supplied as newTestdata argument to mobForestAnalysis function.
prob.cutoff	Predicted probabilities converted into classes (Yes/No, 1/0) based on this probability threshold. Only used for producing predicted Vs actual classes table.
plot	A logical value specifying if the use wishes to veiw performance plots

Value

A list with performance parameters with class "predAccuracyEstimates"

Examples

```
## Random Forest analysis of model based recursive partitioning

## load data
data("BostonHousing", package = "mlbench")
BostonHousing <- BostonHousing[1:90,c("rad", "tax", "crim", "medv", "lstat")]

## Recursive partitioning based on linear regression model medv ~ lstat with 3 trees.
## 1 core/processor used. Supply more processors using 'processors' argument
rfout <- mobForestAnalysis(as.formula(medv ~ lstat),
c("rad", "tax", "crim"), mobForest.controls =
mobForest_control(ntree = 3, mtry = 2, replace = TRUE,
alpha = 0.05, bonferroni = TRUE, minsplit = 25),
data = BostonHousing, processors = 1, model = linearModel)

## get predictive performance estimates and produce a performance plot
pacc <- PredictiveAccuracy(rfout)
```

PredictiveAccuracy-methods

Predictive Performance

Description

~~ Methods for function PredictiveAccuracy ~~

Methods

```
signature(object = "mobForestOutput")
```

print.predAccuracyEstimates

Predictive Accuracy Report

Description

Summarizes predictive Accuracy

Usage

```
## S3 method for class 'predAccuracyEstimates'
print(x, ...)
```

Arguments

x	An object of class 'predAccuracyEstimates' return by "PredictiveAccuracy()" function
...	Additional arguments to print method

Examples

```
data("BostonHousing", package = "mlbench")

rfout <- mobForestAnalysis(as.formula(medv ~ lstat),
c("rad", "tax", "crim"), mobForest.controls =
mobForest_control(ntree = 3, mtry = 2, replace = TRUE,
alpha = 0.05, bonferroni = T, minsplit = 120),
data = BostonHousing, processors = 1, model = linearModel)

## prints predictive accuracy output
pacc <- PredictiveAccuracy(rfout)
pacc
```

`residualPlot` *Residual Diagnostics*

Description

Produces two plots: a) histogram of residuals, b) predicted Vs residuals. This feature is applicable only when linear regression is considered as the node model.

Usage

```
residualPlot(object)
```

Arguments

`object`	An object of class 'mobForestOutput'

Details

Residuals are computed as difference between the predicted values of outcome (summarized across all trees) and observed values of outcome. The residual plots are typical when the fitted values are obtained through linear regression but not when logistic or Poisson regression is considered as a node model. Therefore, the residual plots are produced only when linear regression is considered. For logistic or Poisson models, a message is printed saying "Residual Plot not produced when logistic of Poisson regression is considered as the node model".

Examples

```
## Random Forest analysis of model based recursive partitioning

## load data
data("BostonHousing", package = "mlbench")

## Recursive partitioning based on linear regression model medv ~ lstat with 3 trees.
## 1 core/processor used. Supply more processors using 'processors' argument
rfout <- mobForestAnalysis(as.formula(medv ~ lstat),
c("rad", "tax", "crim"), mobForest.controls =
mobForest_control(ntree = 3, mtry = 2, replace = TRUE,
alpha = 0.05, bonferroni = TRUE, minsplit = 120),
 data = BostonHousing, processors = 1, model = linearModel)

## Produces residual plot
residualPlot(rfout)
```

stringFormula *formula converted to character format*

Description

Model in the formula object converted to a character character

Usage

```
stringFormula(formula)
```

Arguments

formula	formula object

Value

character. model

treePredictions *Predictions from tree model*

Description

Predictions from tree model

Usage

```
treePredictions(j, data, tree)
```

Arguments

j	jth observation
data	A data frame containing the variables in the model.
tree	An object of class mob inheriting from BinaryTree

Details

This method computes predicted outcome for each observation in the data frame using the tree model supplied as an input argument.

Value

A vector of predcited outcome

varimplot *Variable importance plot*

Description

A plot with variable importance score on X-axis and variable name on Y-axis.

Usage

```
varimplot(object)
```

Arguments

object	An object of class mobForestOutput returned by mobForestAnalysis()

References

Leo Breiman (2001). Random Forests. *Machine Learning*, 45(1), 5-32.

See Also

getVarimp

Examples

```
if(require("mlbench")) {

## recursive partitioning of a linear regression model
## load data
data("BostonHousing", package = "mlbench")

## Recursive partitioning based on linear regression model medv ~ lstat with 3 trees.
## 1 core/processor used. Supply more processors using 'processors' argument
rfout <- mobForestAnalysis(as.formula(medv ~ lstat),
c("rad", "tax", "crim"), mobForest.controls =
mobForest_control(ntree = 3, mtry = 2, replace = TRUE,
alpha = 0.05, bonferroni = TRUE, minsplit = 120),
 data = BostonHousing, processors = 1, model = linearModel)

## Produce variable importance plot
varimplot(rfout)
}
```

varimpOutput-class *Class* "varimpOutput"

Description

Variable importance

Objects from the Class

Objects can be created by calls of the form varimp_output.

Slots

varimpMatrix: a matrix containing raw importance scores for all the variables per tree

Methods

No methods defined with class "varimpOutput" in the signature.

References

Strobl, C., Malley, J. and Tutz, G. (2009) An introduction to recursive partitioning: rationale, application, and characteristics of classification and regression trees, bagging, and random forests, *Psychol Methods*, 14, 323-348.

See Also

varimpOutput

varimp_output *Variable importance*

Description

Variable importance matrix containing the decrease in predictive accuracy after permuting the variables across all trees

Usage

```
varimp_output(varimpMatrix)
```

Arguments

varimpMatrix	a matrix containing decrease in predictive accuracy for all variables for each tree

Details

Values of variable 'm' in the oob cases are randomly permuted and R2 obtained through variable-m-permuted oob data is subtracted from R2 obtained on untouched oob data. The average of this number over all the trees in the forest is the raw importance score for variable m.

Value

An object of class `varimpOutput`.

References

Strobl, C., Malley, J. and Tutz, G. (2009) An introduction to recursive partitioning: rationale, application, and characteristics of classification and regression trees, bagging, and random forests, *Psychol Methods*, 14, 323-348.

Index

∗Topic **\textasciitilde\textasciitilde other possible keyword(s) \textasciitilde\textasciitilde**
  getPredictedValues-methods, 6
  PredictiveAccuracy-methods, 18
∗Topic **classes**
  mobForestControl-class, 10
  mobForestOutput-class, 11
  predictionOutput-class, 15
∗Topic **methods**
  getPredictedValues-methods, 6
  PredictiveAccuracy-methods, 18
∗Topic **package**
  mobForest-package, 2

BinaryTree, *15*, *20*
bootstrap *(*mobForestAnalysis*)*, 8

computeAcc, 3
computeMSE, 4
computeR2, 5

getmobForestObject.GLM *(*mobForestAnalysis*)*, 8
getmobForestObject.LM *(*mobForestAnalysis*)*, 8
getPredictedValues, 5
getPredictedValues,mobForestOutput,logical,logical-method *(*mobForestOutput-class*)*, 11
getPredictedValues,mobForestOutput-method *(*getPredictedValues-methods*)*, 6
getPredictedValues-methods, 6
getVarimp, 6, *21*
getVarimp,mobForestOutput-method *(*mobForestOutput-class*)*, 11

logistic_accuracy, 7

mob, *14*, *15*
mob_control, *10*, *14*
mob_control(), *12*, *13*
mob_RF_Tree, 14
mobForest *(*mobForest-package*)*, 2
mobForest-package, 2
mobForest_control, *10*, 12
mobForest_control(), *8*, *9*
mobForest_output, *11*, 13
mobForestAnalysis, 8, *16*
mobForestAnalysis(), *7*, *21*
mobForestControl, *8*, *13*
mobForestControl-class, 10
mobForestOutput, *7*, *9*, *21*
mobForestOutput-class, 11

prediction_output, *15*, 16
predictionOutput, *11*, *14*, *16*
predictionOutput-class, 15
PredictiveAccuracy, *15*, 17
PredictiveAccuracy(), *18*
PredictiveAccuracy,mobForestOutput,logical,logical-method *(*mobForestOutput-class*)*, 11
PredictiveAccuracy,mobForestOutput-method *(*PredictiveAccuracy-methods*)*, 18
PredictiveAccuracy-methods, 18
print.predAccuracyEstimates, 18

residualPlot, 19
residualPlot,mobForestOutput-method *(*mobForestOutput-class*)*, 11

show,mobForestOutput-method *(*mobForestOutput-class*)*, 11
StatModel, *8*, *14*
stringFormula, 20

treePredictions, 20

varimp_output, *22*, 22
varimplot, 21
varimplot,mobForestOutput-method *(*mobForestOutput-class*)*, 11

varimpOutput, *11*, *14*, *22*, *23*
varimpOutput-class, 22