Package 'vita'

December 14, 2015

Type Package

Title Variable Importance Testing Approaches

Version 1.0.0

Date 2015-12-12

Author Ender Celik [aut, cre]

Maintainer Ender Celik <celik.p.ender@gmail.com>

Description Implements the novel testing approach by Janitza et al.(2015) <http://nbn-resolving.de/urn/resolver.pl?urn=nbn:de:bvb:19-epub-25587-4> for the permutation variable importance measure in a random forest and the PIMP-algorithm by Altmann et al.(2010) <doi:10.1093/bioinformatics/btq134>. Janitza et al.(2015) <http://nbn-resolving.de/urn/resolver.pl?urn=nbn:de:bvb:19-epub-25587-4> do not use the ``standard" permutation variable importance but the cross-validated permutation variable importance for the novel test approach. The cross-validated permutation variable importance is not based on the out-of-bag observations but uses a similar strategy which is inspired by the cross-validation procedure. The novel test approach can be applied for classification trees as well as for regression trees. However, the use of the novel testing approach has not been tested for regression trees so far, so this routine is meant for the expert user only and its current state is rather experimental.

Depends R (>= 3.1.0)

License GPL (>= 2)

LazyData TRUE

Imports Rcpp (>= 0.11.6),parallel,randomForest,stats

LinkingTo Rcpp

Suggests mnormt

NeedsCompilation yes

Repository CRAN

Date/Publication 2015-12-14 19:05:44

R topics documented:

`vita-package` *Variable importance testing approaches (vita)*

Description

Implements the novel testing approach by Janitza et al.(2015) for the permutation variable importance measure in a random forest and the PIMP-algorithm by Altmann et al.(2010). Janitza et al.(2015) do not use the "standard" permutation variable importance but the cross-validated permutation variable importance for the novel test approach. The cross-validated permutation variable importance is not based on the out-of-bag observations but uses a similar strategy which is inspired by the cross-validation procedure. The novel test approach can be applied for classification trees as well as for regression trees. However, the use of the novel testing approach has not been tested for regression trees so far, so this routine is meant for the expert user only and its current state is rather experimental.

Details

The novel test approach (`NTA`):

The observed non-positive permutation variable importance values are used to approximate the distribution of variable importance for non-relevant variables. The null distribution Fn0 is computed by mirroring the non-positive variable importance values on the y-axis. Given the approximated null importance distribution, the p-value is the probability of observing the `original PerVarImp` or a larger value. This testing approach is suitable for data with large number of variables without any effect.

`PerVarImp` should be computed based on the hold-out permutation variable importance measures. If using standard variable importance measures the results may be biased.

This function has not been tested for regression tasks so far, so this routine is meant for the expert user only and its current state is rather experimental.

Cross-validated permutation variable importance (`CVPVI`):

This method randomly splits the dataset into k sets of equal size. The method constructs k random forests, where the l-th forest is constructed based on observations that are not part of the l-th set. For each forest the fold-specific permutation variable importance measure is computed using all

observations in the l-th data set: For each tree, the prediction error on the l-th data set is recorded. Then the same is done after permuting the values of each predictor variable.

The differences between the two prediction errors are then averaged over all trees. The cross-validated permutation variable importance is the average of all k-fold-specific permutation variable importances. For classification the mean decrease in accuracy over all classes is used and for regression the mean decrease in MSE.

PIMP testing approach (PIMP):

The PIMP-algorithm by Altmann et al.(2010) permutes S times the response variable y. For each permutation of the response vector y^{*s}, a new forest is grown and the permutation variable importance measure ($VarImp^{*s}$) for all predictor variables X is computed. The vector `perVarImp^{s}` for every predictor variables are used to approximate the null importance distributions.

Given the fitted null importance distribution, the p-value is the probability of observing the *original VarImp* or a larger value.

Author(s)

Ender Celik

References

Breiman L. (2001), *Random Forests*, Machine Learning 45(1),5-32, <doi:10.1023/A:1010933404324>

Altmann A.,Tolosi L., Sander O. and Lengauer T. (2010),*Permutation importance: a corrected feature importance measure*, Bioinformatics Volume 26 (10), 1340-1347, <doi:10.1093/bioinformatics/btq134>

Janitza S, Celik E, Boulesteix A-L, (2015), *A computationally fast variable importance test for random forest for high dimensional data*,Technical Report 185, University of Munich <http://nbn-resolving.de/urn/resolver.pl?urn=nbn:de:bvb:19-epub-25587-4>

See Also

PIMP, NTA, CVPVI, importance, randomForest

`compVarImp` *Compute permutation variable importance measure*

Description

Compute permutation variable importance measure from a random forest for classification and regression.

Usage

```
compVarImp(X, y,rForest,nPerm=1)
```

Arguments

X	a data frame or a matrix of predictors.
y	a response vector. If a factor, classification is assumed, otherwise regression is assumed.
rForest	an object of class randomForest, keep.forest,keep.inbag must be set to True.
nPerm	Number of times the OOB data are permuted per tree for assessing variable importance. Number larger than 1 gives slightly more stable estimate, but not very effective. Currently only implemented for regression.

Details

The permutation variable importance measure is computed from permuting OOB data: For each tree, the prediction error on the out-of-bag observations is recorded. Then the same is done after permuting a predictor variable. The differences between the two error rates are then averaged over all trees.

Value

importance	The permutation variable importance measure. A matrix with nclass + 1 (for classification) or one (for regression) columns. For classification, the first nclass columns are the class-specific measures computed as mean decrease in accuracy. The nclass + 1st column is the mean decrease in accuracy over all classes. For regression the mean decrease in MSE is given.
importanceSD	The "standard errors" of the permutation-based importance measure. For classification, a p by nclass + 1 matrix corresponding to the first nclass + 1 columns of the importance matrix. For regression a vector of length p.
type	one of regression, classification

References

Breiman L. (2001), *Random Forests*, Machine Learning 45(1),5-32, <doi:10.1023/A:101093340432>

See Also

importance, randomForest,CVPVI

Examples

```
##############################
#      Classification        #
##############################
## Simulating data
X = replicate(8,rnorm(100))
X= data.frame( X) #"X" can also be a matrix
z  = with(X,5*X1 + 3*X2 + 2*X3 + 1*X4 -
            5*X5 - 9*X6 - 2*X7 + 1*X8 )
pr = 1/(1+exp(-z))         # pass through an inv-logit function
y = as.factor(rbinom(100,1,pr))
```

```
##############################
## Classification with Random Forest:
library("randomForest")
cl.rf= randomForest(X,y,mtry = 3,ntree=100,
                    importance=TRUE,keep.inbag = TRUE)

##############################
## Permutation variable importance measure
vari= compVarImp(X,y,cl.rf)

##############################
#compare them with the original results
cbind(cl.rf$importance[,1:3],vari$importance)
cbind(cl.rf$importance[,3],vari$importance[,3])
cbind(cl.rf$importanceSD,vari$importanceSD)
cbind(cl.rf$importanceSD[,3],vari$importanceSD[,3])
cbind(cl.rf$type,vari$type)


##############################
#      Regression            #
##############################
## Simulating data
X = replicate(8,rnorm(100))
X= data.frame( X) #"X" can also be a matrix
y= with(X,5*X1 + 3*X2 + 2*X3 + 1*X4 -
          5*X5 - 9*X6 - 2*X7 + 1*X8 )
##############################
## Regression with Random Forest:
library("randomForest")
reg.rf= randomForest(X,y,mtry = 3,ntree=100,
                     importance=TRUE,keep.inbag = TRUE)

##############################
## Permutation variable importance measure
vari= compVarImp(X,y,reg.rf)

##############################
#compare them with the original results
cbind(importance(reg.rf, type=1, scale=FALSE),vari$importance)
cbind(reg.rf$importanceSD,vari$importanceSD)
cbind(reg.rf$type,vari$type)
```

CVPVI *Cross-validated permutation variable importance measure*

Description

Compute cross-validated permutation variable importance measure from a random forest for classification and regression.

Usage

```
## Default S3 method:
CVPVI(X, y, k = 2, mtry= if (!is.null(y) && !is.factor(y))
                        max(floor(ncol(X)/3), 1) else floor(sqrt(ncol(X))),
    ntree = 500, nPerm = 1, parallel = FALSE, ncores = 0, seed = 123, ...)
## S3 method for class 'CVPVI'
print(x, ...)
```

Arguments

X	a data frame or a matrix of predictors.
y	a response vector.
k	an integer for the number of folds. Default is k = 2
mtry	Number of variables randomly sampled as candidates at each split for the l-th forest. Note that the default values are different for classification (mtry=sqrt(p) where p is number of variables in x) and regression (mtry=p/3).
ntree	Number of trees to grow for the l-th forest. Default is ntree=500.
nPerm	Number of times the l-th data set are permuted per tree for assessing variable fold-specific permutation variable importance. Default is nPerm=1.
parallel	Should the CVPVI implementation run parallel? Default is parallel=FALSE and the number of cores is set to one. The parallelized version of the CVPVI implementation are based on mclapply and so are not available on Windows.
ncores	The number of cores to use, i.e. at most how many child processes will be run simultaneously. Must be at least one, and parallelization requires at least two cores. If ncores=0, then the half of CPU cores on the current host are used.
seed	a single integer value to specify seeds. The "combined multiple-recursive generator" from L'Ecuyer (1999) is set as random number generator for the parallelized version of the CVPVI implementation. Default is seed = 123.
...	optional parameters for randomForest
x	for the print method, an CVPVI object

Details

This method randomly splits the dataset into k sets of equal size. The method constructs k random forests, where the l-th forest is constructed based on observations that are not part of the l-th set. For each forest the fold-specific permutation variable importance measure is computed using all observations in the l-th data set: For each tree, the prediction error on the l-th data set is recorded. Then the same is done after permuting the values of each predictor variable. The differences between the two prediction errors are then averaged over all trees. The cross-validated permutation variable importance is the average of all k-fold-specific permutation variable importances. For classification the mean decrease in accuracy over all classes is used and for regression the mean decrease in MSE.

Value

fold_varim	a p by k matrix of fold-specific permutation variable importances. For classification the mean decrease in accuracy over all classes. For regression mean decrease in MSE.
cv_varim	cross-validated permutation variable importances. For classification the mean decrease in accuracy over all classes. For regression mean decrease in MSE.
type	one of regression, classification

References

Janitza S, Celik E, Boulesteix A-L, (2015), *A computationally fast variable importance test for random forest for high dimensional data*,Technical Report 185, University of Munich, <http://nbn-resolving.de/urn/resolver.pl?urn=nbn:de:bvb:19-epub-25587-4>

See Also

VarImpCVl, importance, randomForest, mclapply

Examples

```
##############################
#      Classification        #
##############################
## Simulating data
X = replicate(10,rnorm(100))
X= data.frame( X) #"X" can also be a matrix
z  = with(X,5*X1 + 3*X2 + 2*X3 + 1*X4 -
            5*X5 - 9*X6 - 2*X7 + 1*X8 )
pr = 1/(1+exp(-z))         # pass through an inv-logit function
y = as.factor(rbinom(100,1,pr))
#####################################################################
# cross-validated permutation variable importance
cv_vi = CVPVI(X,y,k = 2,mtry = 3,ntree = 1000,ncores = 4)
print(cv_vi)

#####################################################################
#compare them with the original permutation variable importance
library("randomForest")
cl.rf = randomForest(X,y,mtry = 3,ntree = 1000, importance = TRUE)

round(cbind(importance(cl.rf, type=1, scale=FALSE),cv_vi$cv_varim),digits=5)


##############################
#      Regression            #
##############################

#####################################################################
## Simulating data:
X = replicate(10,rnorm(100))
X = data.frame( X) #"X" can also be a matrix
```

```
y = with(X,2*X1 + 2*X2 + 2*X3 + 1*X4 - 2*X5 - 2*X6 - 1*X7 + 2*X8 )

#################################################################
# cross-validated permutation variable importance
cv_vi = CVPVI(X,y,k = 3,mtry = 3,ntree = 1000,ncores = 2)
print(cv_vi)
#################################################################
#compare them with the original permutation variable importance
library("randomForest")
reg.rf = randomForest(X,y,mtry = 3,ntree = 1000, importance = TRUE)

round(cbind(importance(reg.rf, type=1, scale=FALSE),cv_vi$cv_varim),digits=5)
```

NTA *Novel testing approach*

Description

Calculates the p-values for each permutation variable importance measure, based on the empirical null distribution from non-positive importance values as described in Janitza et al. (2015).

Usage

```
## Default S3 method:
NTA(PerVarImp)
## S3 method for class 'NTA'
print(x, ...)
```

Arguments

`PerVarImp`	permutation variable importance measures in a vector.
`x`	for the print method, an `NTA` object
`...`	optional parameters for `print`

Details

The observed non-positive permutation variable importance values are used to approximate the distribution of variable importance for non-relevant variables. The null distribution Fn0 is computed by mirroring the non-positive variable importance values on the y-axis. Given the approximated null importance distribution, the p-value is the probability of observing the `original PerVarImp` or a larger value. This testing approach is suitable for data with large number of variables without any effect.

`PerVarImp` should be computed based on the hold-out permutation variable importance measures. If using standard variable importance measures the results may be biased.

This function has not been tested for regression tasks so far, so this routine is meant for the expert user only and its current state is rather experimental.

Value

PerVarImp	the orginal permutation variable importance measures.
M	The non-positive variable importance values with the mirrored values on the y-axis.
pvalue	the p-value is the probability of observing the orginal PerVarImp or a larger value, given the approximated null importance distribution.

References

Janitza S, Celik E, Boulesteix A-L, (2015), *A computationally fast variable importance test for random forest for high dimensional data*,Technical Report 185, University of Munich, <http://nbn-resolving.de/urn/resolver.pl?urn=nbn:de:bvb:19-epub-25587-4>

See Also

CVPVI,importance, randomForest

Examples

```
##############################
#      Classification        #
##############################
## Simulating data
X = replicate(100,rnorm(200))
X= data.frame( X) #"X" can also be a matrix
z  = with(X,2*X1 + 3*X2 + 2*X3 + 1*X4 -
            2*X5 - 2*X6 - 2*X7 + 1*X8 )
pr = 1/(1+exp(-z))         # pass through an inv-logit function
y = as.factor(rbinom(200,1,pr))
#####################################################################
# cross-validated permutation variable importance

cv_vi = CVPVI(X,y,k = 2,mtry = 3,ntree = 500,ncores = 2)
#####################################################################
#compare them with the original permutation variable importance
library("randomForest")
cl.rf = randomForest(X,y,mtry = 3,ntree = 500, importance = TRUE)
#####################################################################
# Novel Test approach
cv_p = NTA(cv_vi$cv_varim)
summary(cv_p,pless = 0.1)
pvi_p = NTA(importance(cl.rf, type=1, scale=FALSE))
summary(pvi_p)


##############################
#      Regression            #
##############################
#####################################################################
## Simulating data:
X = replicate(100,rnorm(200))
```

```
X = data.frame( X) #"X" can also be a matrix
y = with(X,2*X1 + 2*X2 + 2*X3 + 1*X4 - 2*X5 - 2*X6 - 1*X7 + 2*X8 )

#################################################################
# cross-validated permutation variable importance
cv_vi = CVPVI(X,y,k = 2,mtry = 3,ntree = 500,ncores = 2)
#################################################################
#compare them with the original permutation variable importance
reg.rf = randomForest(X,y,mtry = 3,ntree = 500, importance = TRUE)
#################################################################
# Novel Test approach (not tested for regression so far!)
cv_p = NTA(cv_vi$cv_varim)
summary(cv_p,pless = 0.1)
pvi_p = NTA(importance(reg.rf, type=1, scale=FALSE))
summary(pvi_p)
```

PIMP *PIMP-algorithm for the permutation variable importance measure*

Description

PIMP implements the test approach of Altmann et al. (2010) for the permutation variable importance measure VarImp in a random forest for classification and regression.

Usage

```
## Default S3 method:
PIMP(X, y, rForest, S = 100, parallel = FALSE, ncores=0, seed = 123, ...)
## S3 method for class 'PIMP'
print(x, ...)
```

Arguments

X	a data frame or a matrix of predictors
y	a response vector. If a factor, classification is assumed, otherwise regression is assumed.
rForest	an object of class randomForest, importance must be set to True.
S	The number of permutations for the response vector y. Default is S=100.
parallel	Should the PIMP-algorithm run parallel? Default is parallel=FALSE and the number of cores is set to one. The parallelized version of the PIMP-algorithm are based on mclapply and so is not available on Windows.
ncores	The number of cores to use, i.e. at most how many child processes will be run simultaneously. Must be at least one, and parallelization requires at least two cores. If ncores=0, then the half of CPU cores on the current host are used.
seed	a single integer value to specify seeds. The "combined multiple-recursive generator" from L'Ecuyer (1999) is set as random number generator for the parallelized version of the PIMP-algorithm. Default is seed = 123.

...	optional parameters for randomForest
x	for the print method, an PIMP object

Details

The PIMP-algorithm by Altmann et al. (2010) permutes S times the response variable y. For each permutation of the response vector y^{*s}, a new forest is grown and the permutation variable importance measure ($VarImp^{*s}$) for all predictor variables X is computed. The vector perVarImp of S $VarImp$ measures for every predictor variables are used to approximate the null importance distributions (PimpTest).

Value

VarImp	the *original permutation variable importance* measures of the random forest.
PerVarImp	a matrix, where each row is a vector containing the S permuted VarImp measures for each predictor variables.
type	one of regression, classification

References

Breiman L. (2001), *Random Forests*, Machine Learning 45(1),5-32, <doi:10.1023/A:1010933404324>

Altmann A.,Tolosi L., Sander O. and Lengauer T. (2010),*Permutation importance: a corrected feature importance measure*, Bioinformatics Volume 26 (10), 1340-1347, <doi:10.1093/bioinformatics/btq134>

See Also

PimpTest, importance, randomForest, mclapply

Examples

```
##############################
#      Regression            #
##############################
##############################
## Simulating data
X = replicate(12,rnorm(100))
X = data.frame(X) #"X" can also be a matrix
y = with(X,2*X1 + 1*X2 + 2*X3 + 1*X4 - 2*X5 - 1*X6 - 1*X7 + 2*X8 )

##############################
## Regression with Random Forest:
library("randomForest")
reg.rf = randomForest(X,y,mtry = 3,ntree=500,importance=TRUE)
##############################
## PIMP-Permutation variable importance measure
# the parallelized version of the PIMP-algorithm
system.time(pimp.varImp.reg<-PIMP(X,y,reg.rf,S=10, parallel=TRUE, ncores=2))
# the non parallelized version of the PIMP-algorithm
system.time(pimp.varImp.reg<-PIMP(X,y,reg.rf,S=10, parallel=FALSE))
```

```
##############################
#      Classification        #
##############################
## Simulating data
X = replicate(12,rnorm(100))
X= data.frame( X) #"X" can also be a matrix
z  = with(X,2*X1 + 3*X2 + 2*X3 + 1*X4 -
           2*X5 - 2*X6 - 2*X7 + 1*X8 )
pr = 1/(1+exp(-z))         # pass through an inv-logit function
y = as.factor(rbinom(100,1,pr))

##############################
## Classification with Random Forest:
cl.rf = randomForest(X,y,mtry = 3,ntree = 500, importance = TRUE)
##############################
## PIMP-Permutation variable importance measure
# the parallelized version of the PIMP-algorithm
system.time(pimp.varImp.cl<-PIMP(X,y,cl.rf,S=10, parallel=TRUE, ncores=2))
# the non parallelized version of the PIMP-algorithm
system.time(pimp.varImp.cl<-PIMP(X,y,cl.rf,S=10, parallel=FALSE))
```

`PimpTest` *PIMP testing approach*

Description

Uses permutations to approximate the null importance distributions for all variables and computes the p-values based on the null importance distribution according to the approach of Altmann et al. (2010).

Usage

```
## Default S3 method:
PimpTest(Pimp, para = FALSE, ...)
## S3 method for class 'PimpTest'
print(x, ...)
```

Arguments

`Pimp`	an object of class `PIMP`
`para`	If para is TRUE the null importance distributions are approximated with Gaussian distributions else with empirical cumulative distributions. Default is `para = FALSE`
`...`	optional parameters, not used
`x`	for the print method, an `PimpTest` object

Details

The vector perVarImp of S variable importance measures for every predictor variables from code PIMP are used to approximate the null importance distributions. If para is TRUE this implementation of the PIMP algorithm fits for each variable a *Gaussian distribution* to the S null importances. If para is FALSE the PIMP algorithm uses the empirical distribution of the S null importances. Given the fitted null importance distribution, the p-value is the probability of observing the *original VarImp* or a larger value.

Value

VarImp	the *original permutation variable importance* measures of the random forest.
PerVarImp	a matrix, where the l-th row contains the S permuted VarImp measures for the l-th predictor variable.
para	Was the null distribution approximated by a Gaussian distribution or by the empirical distribution?
meanPerVarImp	mean for each row of PerVarImp. NULL if para = FALSE
sdPerVarImp	standard deviation for each row of PerVarImp.NULL if para = FALSE
p.ks.test	the p-values of the Kolmogorov-Smirnov Tests for each row PerVarImp. Is the null importance distribution significantly different from a normal distribution with the mean(PerVarImp) and sd(PerVarImp)? NULL if para = FALSE
pvalue	the p-value is the probability of observing the original VarImp or a larger value, given the fitted null importance distribution.

References

Breiman L. (2001), *Random Forests*, Machine Learning 45(1),5-32, <doi:10.1023/A:1010933404324>

Altmann A.,Tolosi L., Sander O. and Lengauer T. (2010),*Permutation importance: a corrected feature importance measure*, Bioinformatics Volume 26 (10), 1340-1347, <doi:10.1093/bioinformatics/btq134>

See Also

PIMP, summary.PimpTest

Examples

```
##############################
#      Regression            #
##############################

## Simulating data
X = replicate(15,rnorm(100))
X = data.frame(X) #"X" can also be a matrix
y = with(X,2*X1 + 1*X2 + 2*X3 + 1*X4 - 2*X5 - 1*X6 - 1*X7 + 2*X8 )

##############################
## Regression with Random Forest:
library("randomForest")
```

```
reg.rf = randomForest(X,y,mtry = 3,ntree=500,importance=TRUE)
##############################
## PIMP-Permutation variable importance measure

system.time(pimp.varImp.reg<-PIMP(X,y,reg.rf,S=100, parallel=TRUE, ncores=2))
pimp.t.reg = PimpTest(pimp.varImp.reg)
summary(pimp.t.reg,pless = 0.1)

##############################
#      Classification        #
##############################

## Simulating data
X = replicate(10,rnorm(200))
X= data.frame( X) #"X" can also be a matrix
z  = with(X,2*X1 + 3*X2 + 2*X3 + 1*X4 -
            2*X5 - 2*X6 - 2*X7 + 1*X8 )
pr = 1/(1+exp(-z))         # pass through an inv-logit function
y = as.factor(rbinom(200,1,pr))

##############################
## Classification with Random Forest:
cl.rf = randomForest(X,y,mtry = 3,ntree = 500, importance = TRUE)
##############################
## PIMP-Permutation variable importance measure
system.time(pimp.varImp.cl<-PIMP(X,y,cl.rf,S=100, parallel=TRUE, ncores=2))
pimp.t.cl = PimpTest(pimp.varImp.cl,para = TRUE)
summary(pimp.t.cl,pless = 0.1)
```

`summary.NTA` *Summarizing the results of novel testing approach*

Description

`summary` method for class "`NTA`".

Usage

```
## S3 method for class 'NTA'
summary(object, pless=0.05,...)
## S3 method for class 'summary.NTA'
print(x, ...)
```

Arguments

`object`	an object of class `NTA`, a result of a call to `NTA`.
`pless`	print only p-values less than pless. Default is `pless=0.05`.
`x`	an object of class `summary.NTA`, a result of a call to `summary.NTA`.
`...`	further arguments passed to or from other methods.

Details

print.summary.NTA tries to be smart about formatting the permutation variable importance values, pvalue and gives "significance stars".

Value

cmat	a p x 2 matrix with columns for the *original permutation variable importance* values and corresponding p-values.
pless	p-values less than pless
call	the matched call to NTA.

See Also

NTA

---

summary.PimpTest *Summarizing PIMP-algorithm outcomes*

Description

summary method for class "PimpTest".

Usage

```
## S3 method for class 'PimpTest'
summary(object, pless=0.05,...)
## S3 method for class 'summary.PimpTest'
print(x, ...)
```

Arguments

object	an object of class PimpTest, a result of a call to PimpTest.
pless	print only p-values less than pless. Default is pless=0.05.
x	an object of class summary.PimpTest, a result of a call to summary.PimpTest.
...	further arguments passed to or from other methods.

Details

print.summary.PimpTest tries to be smart about formatting the VarImp, pvalue etc. and gives "significance stars".

Value

cmat	a p x 3 matrix with columns for the mean(PerVarImp),sd(PerVarImp) and the the p-values of the Kolmogorov-Smirnov Tests.
cmat2	a p x 2 matrix with columns for the *original permutation variable importance* values and corresponding p-value.
para	Shall the null distribution be modelled by a Gaussian distribution?
pless	p-values less than pless
call	the matched call to PimpTest.
call.PIMP	the matched call to PIMP.
type	one of regression, classification

See Also

PimpTest, PIMP

VarImpCVl *Fold-specific permutation variable importance measure*

Description

Compute fold-specific permutation variable importance measure from a random forest for classification and regression.

Usage

```
VarImpCVl(X_l, y_l, rForest, nPerm = 1)
```

Arguments

X_l	a data frame or a matrix of predictors from the l-th data set
y_l	a response vector from the l-th data set. If a factor, classification is assumed, otherwise regression is assumed.
rForest	an object of class randomForest, keep.forest must be set to True. The l-th Forest based on observations that are not part of the l-th data set.
nPerm	Number of permutations performed per tree for computing fold-specific permutation variable importance. Currently only implemented for regression.

Details

The fold-specific permutation variable importance measure is computed from permuting predictor values for the l-th data set: For each tree, the prediction error on the l-th data set is recorded. Then the same is done after permuting each predictor variable from the l-th data set. The difference between the two prediction errors are then averaged over all trees.

Value

`fold_importance`
: Fold-specific permutation variable importance measure. For classification the mean decrease in accuracy over all classes is used, for regression the mean decrease in MSE.

`type`
: one of regression, classification

References

Janitza S, Celik E, Boulesteix A-L, (2015), *A computationally fast variable importance test for random forest for high dimensional data*,Technical Report 185, University of Munich, <http://nbn-resolving.de/urn/resolver.pl?urn=nbn:de:bvb:19-epub-25587-4>

See Also

importance, randomForest

Examples

```
##############################
#      Classification        #
##############################
## Simulating data
X = replicate(8,rnorm(100))
X= data.frame( X) #"X" can also be a matrix
z  = with(X,5*X1 + 3*X2 + 2*X3 + 1*X4 -
          5*X5 - 9*X6 - 2*X7 + 1*X8 )
pr = 1/(1+exp(-z))         # pass through an inv-logit function
y = as.factor(rbinom(100,1,pr))

######################################################################
## Split indexes 2- folds
k = 2
cuts = round(length(y)/k)
from = (0:(k-1)*cuts)+1
to = (1:k*cuts)
rs = sample(1:length(y))
l = 1
######################################################################
## Compute fold-specific permutation variable importance
library("randomForest")

lth = rs[from[l]:to[l]]
# without the l-th data set
Xl = X[-lth,]
yl = y[-lth]
cl.rf_l = randomForest(Xl,yl,keep.forest = TRUE)
# the l-th data set
X_l = X[lth,]
y_l = y[lth]
# Compute l-th fold-specific variable importance
```

```
cvl_varim=VarImpCVl(X_l,y_l,cl.rf_l)

##############################
#      Regression            #
##############################
#####################################################################
## Simulating data:
X = replicate(15,rnorm(120))
X = data.frame( X) #"X" can also be a matrix
y = with(X,2*X1 + 2*X2 + 2*X3 + 1*X4 - 2*X5 - 2*X6 - 1*X7 + 2*X8 )
#####################################################################
## Split indexes 2- folds
k = 2
cuts = round(length(y)/k)
from = (0:(k-1)*cuts)+1
to = (1:k*cuts)
rs = sample(1:length(y))
l = 1

#####################################################################
## Compute fold-specific permutation variable importance
library("randomForest")

lth = rs[from[l]:to[l]]
# without the l-th data set
Xl = X[-lth,]
yl = y[-lth]
reg.rf_l = randomForest(Xl,yl,keep.forest = TRUE)
# the l-th data set
X_l = X[lth,]
y_l = y[lth]
# Compute l-th fold-specific variable importance
CVVI_l = VarImpCVl(X_l,y_l,reg.rf_l)
```

Index