Package 'HPdclassifier'

June 9, 2015

Type Package

Title Distributed classifiers for Big Data

Version 1.1.0

Date 2015-04-17

Author HP Vertica Analytics Team

Maintainer HP Vertica Analytics Team <distributedRTeam@external.groups.hp.com>

Depends R (>= 3.0.0), distributedR, randomForest (>= 4.6-10)

Description
Distributed algorithms for learning classifiers. Written using HP Vertica Distributed R package.

License GPL (>= 2) | file LICENSE

R topics documented:

HPdclassifier-package

Distributed algorithms for classifiers

Description

HPdclassifier provides several distributed algorithms for classifiers. It is written based on the infrastructure created in HP-Labs for distributed computing in R.

Details

Package:	HPdclassifier
Type:	Package
Version:	1.0.0
Date:	2015-01-16

Main Functions:

- hpdRF_parallelTree: It is a distributed function for randomForest that utilizes parallelism in creating each tree of the forest
- hpdRF_parallelForest: It is a distributed function for randomForest that utilizes parallelism in creating sub-forests of the forest

Author(s)

HP Vertica Analytics Team <distributedRTeam@external.groups.hp.com>

References

1. Using R for Iterative and Incremental Processing. Shivaram Venkataraman, Indrajit Roy, Alvin AuYoung, Rob Schreiber. HotCloud 2012, Boston, USA.

confusionMatrix *Confusion Matrix*

Description

This function generates confusion matrix for observed and predicted values of a classifier.

Usage

```
confusionMatrix(observed, predicted)
```

Arguments

observed	the response observed in the test data.
predicted	the predicted value for response.

Value

the returned value is the generated confusion matrix.

Note

it is assumed that an appropriate predict function has generated 'provided' input.

Author(s)

HP Vertica Analytics Team

Examples

```
 ## Not run:
library(HPdclassifier)
distributedR_start()

rRF <- randomForest(Species ~ ., data=iris, keep.forest=TRUE,
                    xtest=iris[,-5],ytest=iris[,5])

predicted <- predict(rRF,iris[, -5])
confusionMatrix(iris[,5], predicted)


## End(Not run)
```

---

deploy.hpdRF_parallelTree

Convert hpdRF_parallelTree model to that of randomForest model

---

Description

This function converts the formatting of the trees to match that of randomForest model so that predict.randomForest can be used

Usage

```
deploy.hpdRF_parallelTree <- function(model)
```

Arguments

model	an object of class hpdRF_parallelTree, as that created by the function hpdRF_parallelTree

Details

The randomForest model can only handle categorical variables with less than 32 categories

Value

An object of class randomForest

Author(s)

HP Vertica Analytics Team

`errorRate` *Error Rates*

Description

This function calculates total error rate and error rates of each class for observed and predicted values of a classifier.

Usage

```
errorRate(observed, predicted)
```

Arguments

`observed`	the response observed in the test data.
`predicted`	the predicted value for response.

Value

the returned value is an array. The first element of the array is the error rate, which equals to the total number of correct predictions divided by the total number of predictions. The remaind elements of the array, represent error rates per class. An error rate per class is the error rate for the samples with a particular category in their response.

Note

it is assumed that an appropriate predict function has generated 'provided' input.

Author(s)

HP Vertica Analytics Team

Examples

```
 ## Not run:
library(HPdclassifier)
distributedR_start()

rRF <- randomForest(Species ~ ., data=iris, keep.forest=TRUE,
                              xtest=iris[,-5],ytest=iris[,5])

predicted <- predict(rRF,iris[, -5])
errorRate(iris[,5], predicted)


## End(Not run)
```

hpdRF_parallelForest
Distributed randomForest with parallelism in sub-forest level

Description

hpdRF_parallelForest function runs `randomForest` function of randomForest package in a distributed fashion with parallelism in sub-forest level.

Description

`hpdRF_parallelForest` calls several instances of `randomForest` distributed across a cluster system in order to create sub-forests concurrently. Therefore, the master distributes the input data among all R-executors of the distributedR environment, and trees on different sub-sections of the forest are created simultaneously. At the end, all these trees are combined to result a single forest.

The interface of `hpdRF_parallelForest` is similar to `randomForest`. Indeed it adds two arguments `nExecutor` and `trace`, and removes several other arguments: `subset`, `do.trace`, `corr.bias`, `keep.inbag`, and `oob.prox`. Nevertheless, it must be noticed that default value of some arguments are changed as well to make the algorithm more scalable for big data problems; e.g, `proximity` is FALSE by default. Its returned result is also completely compatible to the result of `randomForest`.

Usage

```
## S3 method for class 'formula'
hpdRF_parallelForest(formula, data=NULL, ..., ntree=500,
         na.action=na.fail, nExecutor, trace=FALSE,
         completeModel=FALSE)
## Default S3 method:
hpdRF_parallelForest(x, y=NULL,  xtest=NULL, ytest=NULL, ntree=500,
         mtry=if (!is.null(y) && !is.factor(y) && !is.dframe(y))
         max(floor(ncol(x)/3), 1) else floor(sqrt(ncol(x))),
         replace=TRUE, classwt=NULL, cutoff, strata,
         sampsize = if (replace) nrow(x) else ceiling(.632*nrow(x)),
         nodesize = if (!is.null(y) && !is.factor(y) &&
         !is.dframe(y)) 5 else 1,
         maxnodes=NULL, importance=FALSE, localImp=FALSE, nPerm=1,
         proximity=FALSE,norm.votes=TRUE, keep.forest=TRUE,
         nExecutor, trace=FALSE, completeModel=FALSE, ...)
```

Arguments

`data`	a data frame or dframe which contains samples.
`na.action`	A function to specify the action to be taken if NAs are found. (NOTE: If given, this argument must be named.)
`formula`	a formula describing the model to be fitted. It must be a simple formula without any arithmetic operation among columns.

x	when a data frame or a matrix of predictors assigned to x, its size should not be bigger than 2GB. For bigger datasets, darray should be used. darray does not support categorical data. Therefore, dframe and the first interface should be used for classification problems of large datasets with categorical data.
y	a response vector. If a factor, classification is assumed, otherwise regression is assumed. If omitted, hpdRF_parallelForest will run in unsupervised mode. When x is a darray), y should be also a darray with a single column.
xtest	a data frame or matrix (like x) containing predictors for the test set. When x is a darray, it should be of the same type.
ytest	response for the test set. Its type should be consistent with y. Moreover, it should have a single column.
ntree	Number of trees to grow. This should not be set to too small a number, to ensure that every input row gets predicted at least a few times.
mtry	Number of variables randomly sampled as candidates at each split. Note that the default values are different for classification (sqrt(p) where p is number of variables in x) and regression (p/3)
replace	Should sampling of cases be done with or without replacement?
classwt	Priors of the classes. Need not add up to one. Ignored for regression.
cutoff	(Classification only) A vector of length equal to number of classes. The 'winning' class for an observation is the one with the maximum ratio of proportion of votes to cutoff. Default is 1/k where k is the number of classes (i.e., majority vote wins).
strata	A (factor) variable that is used for stratified sampling.
sampsize	Size(s) of sample to draw. For classification, if sampsize is a vector of the length the number of strata, then sampling is stratified by strata, and the elements of sampsize indicate the numbers to be drawn from the strata.
nodesize	Minimum size of terminal nodes. Setting this number larger causes smaller trees to be grown (and thus take less time). Note that the default values are different for classification (1) and regression (5).
maxnodes	Maximum number of terminal nodes trees in the forest can have. If not given, trees are grown to the maximum possible (subject to limits by nodesize). If set larger than maximum possible, a warning is issued.
importance	Should importance of predictors be assessed?
localImp	Should casewise importance measure be computed? (Setting this to TRUE will override importance.)
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.
proximity	a logical value which indicates if the proximity measure among the rows should be calculated. It is FALSE by default because it is very memory inefficient. Moreover, it is calculated only on 'out-of-bag' data
norm.votes	If TRUE (default), the final result of votes are expressed as fractions. If FALSE, raw vote counts are returned (useful for combining results from different runs). Ignored for regression.
keep.forest	If set to FALSE, the forest will not be retained in the output object.

nExecutor	a positive integer number indicating the number of tasks for running the function. To have optimal performance, it is recommended to have this number smaller than the number of R-executors in the distributedR environment. It cannot be bigger than ntree.
trace	when this argument is true, intermediate steps of the progress are displayed.
completeModel	when it is FALSE (default), the output values that preserve information per sample are discarded. They are 'oob.times', 'votes', 'predicted', 'confusion', 'err.rate', 'mse', 'rsq', 'proximity', and 'test'. This feature is intended to keep the size of the output model small.
...	optional parameters to be passed to the low level function.

Value

An object of class randomForest. The result is similar to the result of the combine function in randomForest package and will contain the following components.

call	the original call to hpdRF_parallelForest
type	one of regression, classification, or unsupervised.
predicted	(only when completeModel=TRUE) the predicted values of the input data based on out-of-bag samples.
importance	a matrix with nclass + 2 (for classification) or two (for regression) columns. For classification, the first nclass columns are the class-specific measures computed as mean descrease in accuracy. The nclass + 1st column is the mean descrease in accuracy over all classes. The last column is the mean decrease in Gini index. For Regression, the first column is the mean decrease in accuracy and the second the mean decrease in MSE. If importance=FALSE, the last measure is still returned as a vector.
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 length p vector.
localImp	a p by n matrix containing the casewise importance measures, the [i,j] element of which is the importance of i-th variable on the j-th case. NULL if localImp=FALSE.
ntree	number of trees grown.
mtry	number of predictors sampled for spliting at each node.
forest	(a list that contains the entire forest; NULL if hpdRF_parallelForest is run in unsupervised mode or if keep.forest=FALSE.
err.rate	(classification only) vector error rates of the prediction on the input data, the i-th element being the (OOB) error rate for all trees up to the i-th.
confusion	(classification only) the confusion matrix of the prediction (based on OOB data).
votes	(classification only, and only when completeModel=TRUE) a matrix with one row for each input data point and one column for each class, giving the fraction or number of (OOB) 'votes' from the random forest.
oob.times	(only when completeModel=TRUE) number of times cases are 'out-of-bag' (and thus used in computing OOB error estimate)
y	(only when completeModel=TRUE) the response vector if it is made available in the input.

proximity	(only when completeModel=TRUE) a matrix of proximity measures among the input (based on the frequency that pairs of data points are in the same terminal nodes) when proximity=TRUE.
mse	(regression only) vector of mean square errors: sum of squared residuals divided by n.
rsq	(regression only) "pseudo R-squared": 1 - mse / Var(y).
test	(only when completeModel=TRUE) if test set is given (through the xtest or additionally ytest arguments), this component is a list which contains the corresponding predicted, err.rate, confusion, votes (for classification) or predicted, mse and rsq (for regression) for the test set.
terms	it contains a formula identifying response and predictors (for classification and regression types).

Note

When ntree is not big enough in comparison to nExecutor, some of the returned predicted values may become NULL. It is the same for values of 'votes' matrix when they are normalized (norm.votes=TRUE). Returned values for err.rate, votes, and oob.times are valid only for classification type.

Three scenarios can be imagined for the type of input data. When ordinary R types are used (matrix or data.frame) the behavior is similar to the randomForest function; however, the total size of the input data cannot be bigger than 2GB. In fact, for bigger data size distributed data types; i.e., darray or dframe, should be used. When x is of type darray, in the case of existence y must be of type darray as well. Regarding the fact that darray does not support categorical data, this data type cannot be used for classification mode. When x is of type dframe, no value can be assigned to y; indeed for this data type, the formula interface should be used for classification and the default interface for unsupervised mode.

Author(s)

HP Vertica Analytics Team

References

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

Breiman, L (2002), "Manual On Setting Up, Using, And Understanding Random Forests V3.1", http://oz.berkeley.edu/users/breiman/Using_random_forests_V3.1.pdf.

Random Forests V4.6-10, http://cran.r-project.org/web/packages/randomForest/randomForest.pdf.

Examples

```
 ## Not run:

library(HPdclassifier)
distributedR_start()
drs <- distributedR_status()
nparts <- sum(drs$Ins)

## Classification:
##data(iris)
iris.rf <- hpdRF_parallelForest(Species ~ ., data=iris, importance=TRUE,
```

```
                          nExecutor=nparts)
print(iris.rf)

## The 'unsupervised' case:
iris.urf <- hpdRF_parallelForest(iris[, -5], nExecutor=nparts,
                            proximity=TRUE, completeModel=TRUE)
MDSplot(iris.urf, iris$Species)

## stratified sampling: draw 20, 30, and 20 of the species to grow each tree.
(iris.rf2 <- hpdRF_parallelForest(iris[1:4], iris$Species,
                           sampsize=c(20, 30, 20), nExecutor=nparts))

## Regression:
## data(airquality)
ozone.rf <- hpdRF_parallelForest(Ozone ~ ., data=airquality, mtry=3,
                    importance=TRUE, na.action=na.omit,
                    nExecutor=nparts, completeModel=TRUE)
print(ozone.rf)
## Show "importance" of variables: higher value mean more important:
round(importance(ozone.rf), 2)

## "x" can be a matrix instead of a data frame:
x <- matrix(runif(5e2), 100)
y <- gl(2, 50)
(myrf <- hpdRF_parallelForest(x, y, nExecutor=nparts))
(predict(myrf, x))

## "complicated" formula:
(swiss.rf <- hpdRF_parallelForest(sqrt(Fertility)~. - Catholic + I(Catholic<50),
                           data=swiss, nExecutor=nparts))
(predict(swiss.rf, swiss))
## Test use of 32-level factor as a predictor:
x <- data.frame(x1=gl(32, 5), x2=runif(160), y=rnorm(160))
(rf1 <- hpdRF_parallelForest(x[-3], x[[3]], ntree=10, nExecutor=nparts))

## Grow no more than 4 nodes per tree:
(treesize(hpdRF_parallelForest(Species ~ ., data=iris, maxnodes=4, ntree=30,
                                nExecutor=nparts)))

distributedR_shutdown()

## End(Not run)
```

`hpdRF_parallelTree` *Random Forest Models over Distributed Data*

Description

Distributed alternative for randomForest package that uses distributedR framework

Usage

```
hpdRF_parallelTree <- function(formula, data, ntree = 50, xtest, ytest, mtry, re
```

Arguments

formula	a formula describing the model to be fitted
data	a data.frame or dframe containing the variables in the model
xtest	a dframe or data.frame containing predictors for the test set
ytest	response for the test set
ntree	Number of trees to grow. This should not be set to too small a number, to ensure that every input row gets predicted at least a few times. However it should also not be too large a number as it will take a long time and a lot of memory to train
mtry	Number of variables randomly sampled as candidates at each split. Note that the default values are different for classification (sqrt(p) where p is number of variables in x) and regression (p/3)
replace	Should sampling of cases be done with or without replacement?
cutoff	(Classification only) A vector of length equal to number of classes. The <e2><80><98>winning<e2>< class for an observation is the one with the maximum ratio of proportion of votes to cutoff. Default is 1/k where k is the number of classes (i.e., majority vote wins).
nodesize	Minimum size of terminal nodes. Setting this number larger causes smaller trees to be grown (and thus take less time). Note that the default values are different for classification (1) and regression (5).
maxnodes	Maximum number of terminal nodes trees in the forest can have. If not given, trees are grown to the maximum possible (subject to limits by nodesize).
do.trace	If set to TRUE, give a more verbose output
keep.forest	If set to FALSE, the forest will not be retained in the output object
na.action	A function to specify the action to be taken if NAs are found. This does not apply to xtest,ytest. All NA observations for xtest will be predicted (with perhaps lower accuracy). Any NA observations in ytest are ignored (corresponding observations of xtest are ignored as well)
nBins	Number of bins to use for numerical variables. Number of bins for categorical variables is set to the number of categories for the variable
completeModel	If set to FALSE, xtest,ytest will be ignored and out of bag samples will not be predicted

Details

predictors and responses must align with each other for variables xtest and ytest (have the same number of rows and the same number of blocks in the case of a dframe) because row names are not used for alignment. The alias name for this function is `hpdrandomForest`.

Value

An object of class hpdRF_parallelTree, which is a list with the following components:

call	the original call to hpdRF_parallelTree
type	either regression or classification
predicted	the predicted values of the input data based on out-of-bag samples.
ntree	number of trees grown.

mtry	number of predictors sampled for spliting at each node.
forest	A list that contains entire forest. This is not available if keep.forest is set to FALSE
err.rate	(classification only, and only when completeModel=TRUE) vector error rates of the prediction on the input data, the i-th element being the (OOB) error rate for all trees up to the i-th
confusion	(classification only, and only when completeModel=TRUE) the confusion matrix of the prediction (based on OOB data).
mse	(regression only, and only when completeModel=TRUE) vector of mean square errors: sum of squared residuals divided by n.
rsq	(regression only, and only when completeModel=TRUE) <e2><80><9c>pseudo R-squared<e2><80><9d>: 1 - mse / Var(y)
test	if test set is given (through the xtest or additionally ytest arguments), this component is a list which contains the corresponding predicted, err.rate, confusion, votes (for classification) or predicted, mse and rsq (for regression) for the test set.

Author(s)

HP Vertica Analytics Team

`meanSquared` *Mean Squared Residuals*

Description

This function calculates mean squared residuals for observed and predicted values.

Usage

```
meanSquared(observed, predicted, na.rm=FALSE)
```

Arguments

observed	the response observed in the test data.
predicted	the predicted value for response.
na.rm	logical. Should missing values (including <e2><80><98>NaN<e2><80><99>) be removed?

Value

the mean squared of residuals is returned.

Note

it is assumed that an appropriate predict function has generated 'provided' input.

Author(s)

HP Vertica Analytics Team

Examples

```
 ## Not run:
library(HPdclassifier)
distributedR_start()

testData <- na.omit(airquality)
rRF.ozone <- randomForest(Ozone ~ ., data=airquality,
            mtry=3, na.action=na.omit,
            xtest=testData[,-1],ytest=testData[,1],keep.forest=TRUE)

predicted <- predict(rRF.ozone,testData[,-1])
meanSquared(testData[,1], predicted)


## End(Not run)
```

`predict.hpdRF_parallelForest` *distributed predict method for applying a random forest objects on a darray or a dframe*

Description

This function can be used to apply a model of type `hpdRF_parallelForest` or `randomForest` to a new data for prediction.

Usage

```
predict.hpdRF_parallelForest(object, newdata, trace=FALSE)
```

Arguments

`object`	an object of class `randomForest`, as that created by the function `randomForest` or `hpdRF_parallelForest`.
`newdata`	a darray, a dframe, a data.frame, or a matrix that contains new data. darray is highly recommended to dframe when there is no categorial data
`trace`	when this argument is true, intermediate steps of the progress are displayed.

Value

It returns predicted classes in a distributed or non-distributed objects depending on the type of the input. When the newdata is of type darray, the type of returned value will be also darray unless the output is categorical data. When the output is a dframe when the newdata is of type dframe.

Author(s)

HP Vertica Analytics Team

References

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

See Also

hpdRF_parallelForest

Examples

```
 ## Not run:
# example for darray
library(HPdclassifier)
distributedR_start()
drs <- distributedR_status()
nparts <- sum(drs$Ins)

nSamples <- 100
nAttributes <- 5
nPartitions <- 2

dax <- darray(c(nSamples,nAttributes),
              c(ceiling(nSamples/nPartitions),nAttributes))
day <- darray(c(nSamples,1), c(ceiling(nSamples/nPartitions),1))

foreach(i,1:npartitions(dax),function(x=splits(dax,i),
                                      y=splits(day,i),id=i){
    x <- matrix(runif(nrow(x)*ncol(x)), nrow(x),ncol(x))
    y <- matrix(runif(nrow(y)), nrow(y), 1)
    update(x)
    update(y)
})

(myrf1 <- hpdRF_parallelForest(dax, day, nExecutor=nparts))
dp <- predict.hpdRF_parallelForest(myrf1, dax)

# example for dframe
nSamples <- nrow(iris)
nAttributes <- ncol(iris)
nPartitions <- 4

df <- dframe(c(nSamples,nAttributes),
             c(ceiling(nSamples/nPartitions),nAttributes))
end = cumsum(partitionsize(df)[,1])
start = c(0,end[-length(end)])

foreach(i, 1:npartitions(df), function(xi=splits(df,i),
  start = start[i]+1, end = end[i], iris=iris){
     xi <- iris[start:end,]
     update(xi)
})
# the following line will be redundant in the next release
colnames(df) <- colnames(iris)

(myrf2 <- hpdRF_parallelForest(Species ~ ., df, nExecutor=nPartitions))
fp2 <- predict.hpdRF_parallelForest(myrf2, df)


## End(Not run)
```

predict.hpdRF_parallelTree

Predict function for distributed random forest model

Description

given a distributed random forest model and new observations of the feature variables, predict the responses of the new observations

Usage

```
predict.hpdRF(object, newdata, cutoff, do.trace, na.action = na.fail)
```

Arguments

object	an object of class hpdRF_parallelTree, as that created by the function hpdRF_parallelTree
newdata	a dframe containing new data
cutoff	(Classification only) A vector of length equal to number of classes. The <e2><80><98>winning<e2>< class for an observation is the one with the maximum ratio of proportion of votes to cutoff. Default is taken from the forest$cutoff component of object
do.trace	If set to TRUE, give a more verbose output as randomForest is run.
na.action	A function to specify the action to be taken if NAs are found

Value

A list that has the following components:

response	predictions of the newdata

Author(s)

HP Vertica Analytics Team

predictHPdRF

distributed predict method for applying a random forest objects on a darray or a dframe

Description

DEPRECATED - This function is deprecated, and we suggest using the predict.hpdRF_parallelForest or predict.hpdRF_parallelTree functions depending on the type of the model. The new functions are aligned with generic predict function format.

Usage

```
predictHPdRF(object, newdata, trace=FALSE)
```

Arguments

object	an object of class randomForest, as that created by the function randomForest or hpdRF_parallelForest.
newdata	a darray or a dframe containing new data. darray is highly recommended when there is no categorial data
trace	when this argument is true, intermediate steps of the progress are displayed.

Value

It returns a darray or a dframe of predicted classes. When the newdata is of type darray, the type of returned value will be also darray unless the output is categorical data. When the output is a dframe when the newdata is of type dframe.

Author(s)

HP Vertica Analytics Team

References

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

See Also

hpdRF_parallelForest

Examples

```
 ## Not run:
# example for darray
library(HPdclassifier)
distributedR_start()
drs <- distributedR_status()
nparts <- sum(drs$Ins)

nSamples <- 100
nAttributes <- 5
nPartitions <- 2

dax <- darray(c(nSamples,nAttributes),
              c(ceiling(nSamples/nPartitions),nAttributes))
day <- darray(c(nSamples,1), c(ceiling(nSamples/nPartitions),1))

foreach(i,1:npartitions(dax),function(x=splits(dax,i),
                                      y=splits(day,i),id=i){
    x <- matrix(runif(nrow(x)*ncol(x)), nrow(x),ncol(x))
    y <- matrix(runif(nrow(y)), nrow(y), 1)
    update(x)
    update(y)
})

(myrf1 <- hpdRF_parallelForest(dax, day, nExecutor=nparts))
dp <- predictHPdRF(myrf1, dax)

# example for dframe
nSamples <- nrow(iris)
```

```
nAttributes <- ncol(iris)
nPartitions <- 4

df <- dframe(c(nSamples,nAttributes),
             c(ceiling(nSamples/nPartitions),nAttributes))
end = cumsum(partitionsize(df)[,1])
start = c(0,end[-length(end)])

foreach(i, 1:npartitions(df), function(xi=splits(df,i),
  start = start[i]+1, end = end[i], iris=iris){
     xi <- iris[start:end,]
     update(xi)
})
# the following line will be redundant in the next release
colnames(df) <- colnames(iris)

(myrf2 <- hpdRF_parallelForest(Species ~ ., df, nExecutor=nPartitions))
fp2 <- predictHPdRF(myrf2, df)


## End(Not run)
```

`print.hpdRFtree` *Print Trees returned by hpdRF_parallelTree*

Description

A function that can print the trees returned by hpdRF_parallelTree

Usage

```
print.hpdRFtree <- function(tree, max_depth = 2,classes)
```

Arguments

`model`	an object of class hpdRFtree, as created by the function hpdRF_parallelTree
`max_depth`	The maximum depth the trees will be printed (trees can be very deep)
`classes`	(Classification only) The list of classes. Default value is to check if there are any classes associated with the tree

Details

Classes should be be passed in otherwise numerical values will be displayed. The class of subtrees is not set to hpdRFtree so to print them, explicity use print.hpdRFtree(subtree) instead of just print(subtree)

Value

Tree is printed in an XML format

Author(s)

HP Vertica Analytics Team

print.hpdRF_parallelTree *Print hpdRF_parallelTree models*

Description

A function that can print summary information for models of class hpdRF_parallelTree

Usage

```
print.hpdRF_parallelTree <- function(model, max_depth = 2)
```

Arguments

model	an object of class hpdRF_parallelTree, as that created by the function hpdRF_parallelTree
max_depth	The maximum depth the trees will be printed (trees can be very deep)

Details

Does not display the trees in the model.

Value

A summary of the model is printed

Author(s)

HP Vertica Analytics Team

rSquared *R-squared*

Description

This function calculates R-squared (1 - mse / Var(y)) for observed and predicted values.

Usage

```
rSquared(observed, predicted, na.rm=FALSE)
```

Arguments

observed	the response observed in the test data.
predicted	the predicted value for response.
na.rm	logical. Should missing values (including <e2><80><98>NaN<e2><80><99>) be removed?

Value

the value of R-squared is returned.

Note

it is assumed that an appropriate predict function has generated 'provided' input.

Author(s)

HP Vertica Analytics Team

Examples

```
 ## Not run:
library(HPdclassifier)
distributedR_start()

testData <- na.omit(airquality)
rRF.ozone <- randomForest(Ozone ~ ., data=airquality,
            mtry=3, na.action=na.omit,
            xtest=testData[,-1],ytest=testData[,1],keep.forest=TRUE)

predicted <- predict(rRF.ozone,testData[,-1])
rSquared(testData[,1], predicted)

## End(Not run)
```

Index