Package ‘rfVarImpOOB’

April 5, 2019

Title Unbiased Variable Importance for Random Forests
Version 1.0
Date 2019-03-22
Depends R (>= 3.2.2), stats, randomForest
Imports ggplot2, binaryLogic, dplyr,titanic.prob.ggpubr,magrittr
Suggests knitr,rmarkdown
Author Markus Loecher <Markus.Loecher@gmail.com>
Maintainer Markus Loecher <Markus.Loecher@gmail.com>
License GPL (>= 2)
Repository CRAN
LazyData true
VignetteBuilder knitr
NeedsCompilation no
Date/Publication 2019-04-05 12:50:06 UTC

R topics documented:

Accuracy ................................................................. 2
arabidopsis .......................................................... 3
GiniImportanceForest ........................................... 4
GiniImportanceTree .............................................. 5
gini_index ........................................................... 7
gini_process ......................................................... 9
InOutBags ............................................................. 11
Accuracy computes accuracy of a vector

Description

Accuracy is defined as the proportion of correct labels

Usage

Accuracy(y, yHat, dig = 8)

Arguments

y vector of categorical/nominal values
yHat prediction/estimate
dig number of digits

Value

Accuracy defined as proportion of values equal to majority

Author(s)

Markus Loecher <Markus.Loecher@gmail.com>

Examples

Accuracy(c(rep(0, 9), 1), 1)
RNA editing is the process whereby RNA is modified from the sequence of the corresponding DNA template [1]. For instance, cytidine-to-uridine conversion (abbreviated C-to-U conversion) is common in plant mitochondria. The mechanisms of this conversion remain largely unknown, although the role of neighboring nucleotides is emphasized. Cummings and Myers [1] suggest to use information from sequence regions flanking the sites of interest to predict editing in Arabidopsis thaliana, Brassicanapus and Oryza sativa based on random forests. The Arabidopsis thaliana data of [1] can be loaded from the journal Web site.

For each of the 876 observations, the data set gives the response at the site of interest (binary: edited/not edited) and as potential predictor variables the 40 nucleotides at positions -20 to 20, relative to the edited site (4 categories), cp: the codon position (4 categories), fe: the estimated folding energy (continuous) and dfe: the difference in estimated folding energy between pre- edited and edited sequences (continuous).

Data frame with columns
- edit binary: the response at the site of interest
- X.k nucleotides at positions -k, relative to the edited site (4 categories)
- Xk nucleotides at positions k, relative to the edited site (4 categories)
- cp the codon position (4 categories),
- fe the estimated folding energy (continuous)
- dfe the difference in estimated folding energy between pre- edited and edited sequences (continuous)

Source
GiniImportanceForest  *computes inbag and OOB Gini importance averaged over all trees in a forest*

**Description**

workhorse function of this package

**Usage**

```r
GiniImportanceForest(RF, data, ylabel = "Survived", zeroLeaf = TRUE,

agg = c("mean", "median", "none")[1], score = c("PMDI21",

"MDI", "MDA", "MIA")[1], Predictor = Mode, verbose = 0)
```

**Arguments**

- `RF` object returned by call to `randomForest()`
- `data` data which was used to train the RF. NOTE: assumes setting of inbag=TRUE while training
- `ylabel` name of dependent variable
- `zeroLeaf` if TRUE discard the information gain due to splits resulting in n=1
- `agg` method of aggregating importance scores across trees. If "none" return the raw arrays (for debugging)
- `score` scoring method:MDI=mean decrease impurity (Gini),MDA=mean decrease accuracy (permutation),MIA=mean increase accuracy
- `Predictor` function to estimate node prediction, such as Mode or mean or median. Alternatively, pass an array of numbers as replacement for the yHat column of tree
- `verbose` level of verbosity

**Value**

matrix with variable importance scores and their stdevs

**Author(s)**

Markus Loecher <Markus.Loecher@gmail.com>
GiniImportanceTree

Examples

data("titanic_train", package = "rfVarImpOOB", envir = environment())

set.seed(123)

ranRows = sample(nrow(titanic_train), 300)

data = titanic_train[ranRows,]

RF = randomForest::randomForest(formula = Survived ~ Sex + Pclass + PassengerId,
                                 data = data,
                                 ntree = 5, importance = TRUE,
                                 mtry = 3, keep.inbag = TRUE,
                                 nodesize = 20)

data$Survived = as.numeric(data$Survived) - 1

VI_Titanic = GiniImportanceForest(RF, data, ylab = "Survived")

GiniImportanceTree  computes Gini information gain for one tree from randomForest

Description
computes importance scores for an individual tree.
These can be based on Gini impurity or Accuracy or logloss
GiniImportanceTree

Usage

GiniImportanceTree(bag, RF, k, ylabel = "Survived", returnTree = FALSE,

zeroLeaf = TRUE, score = c("PMDI","MDI", "MDA", "MIA")[1],

Predictor = Mode, verbose = 0)

Arguments

  bag    data to compute the Gini gain for
  RF     object returned by call to randomForest()
  k      which tree
  ylabel name of dependent variable
  returnTree if TRUE returns the tree data frame otherwise the aggregated Gini importance
              grouped by split variables
  zeroLeaf if TRUE discard the information gain due to splits resulting in n=1
  score    scoring method:PMDI=mean decrease penalized Gini impurity (note:the last
digit is the exponent of the penalty!),
          MDI=mean decrease impurity (Gini), MDA=mean decrease accuracy (permutation),
          MIA=mean increase accuracy
  Predictor function to estimate node prediction, such as Mode or mean or median. Alternatively, pass an array of numbers as replacement for the yHat column of tree
  verbose level of verbosity

Value

  if returnTree==TRUE returns the tree data frame otherwise the aggregated Gini importance grouped
  by split variables

Author(s)

  Markus Loecher <Markus.Loecher@gmail.com>

Examples

  rftit = rfTitanic(nRows = 500, nodesize=10)

  rftit$data$Survived = as.numeric(rftit$data$Survived)-1
gini_index

compute Gini impurity for binary values only

Description

simple function to compute simple or penalized Gini impurity
The "penalty" compares the class probabilities \( \hat{p} \) with a reference estimate \( \hat{p} \) which would typically serve as a prediction (e.g. in a tree node).

Usage

gini_index(pHat, pEst = NULL, k = 2, kind = 1, w = 2)

Arguments

\( \hat{p} \)hat \quad \text{probabilities from the current data,}
\( \hat{p} \)est \quad \text{estimated class probabilities (typically from an earlier inbag estimation). Only pass if you intend to compute the "validation-penalized Gini"}
\( k \) \quad \text{exponent of penalty term: } |pHat-pEst|^k
\( \text{kind} \) \quad \text{kind of penalty}
\( w \) \quad \text{weights, default is 2 if you pass just a single probability instead of the vector (p,1-p)}
**Value**

simple or penalized Gini impurity

**Author(s)**

Markus Loecher <Markus.Loecher@gmail.com>

**Examples**

```r
#Test binary case:

gini_index(0.5,0.5,kind=1)
gini_index(0.9,0.1,kind=1)
gini_index(0.1,0.9,kind=1)

gini_index(0.5,0.5,kind=2)
gini_index(0.9,0.1,kind=2)
gini_index(0.1,0.9,kind=2)

gini_index(0.5,0.5,kind=3)
gini_index(0.9,0.1,kind=3)
gini_index(0.1,0.9,kind=3)
```
Description
computes Gini index

Usage
gini_process(classes, splitvar = NULL)

Arguments
  classes vector of factors/categorical vars
  splitvar split variable

Value
Gini index

Author(s)
Markus Loecher <Markus.Loecher@gmail.com>

Examples

  # Test binary case:

  # 50/50 split

  gini_process(c(rep(0,10),rep(1,10))) # 0.5 CORRECT!

  # 10/90 split

  gini_process(c(rep(0,1),rep(1,9))) # 0.18 = CORRECT!
#0/100 split

gini_process(factor(c(rep(0,0),rep(1,10)), levels=c(0,1)))#0

#Test binary case:

#25/25/25/25 split

gini_process(factor(c(rep(0,5),rep(1,5),rep(2,5),
rep(3,5)), levels=c(0:3)))#0.75 = 4*0.25*0.75 CORRECT !

#10/10/10/70 split

gini_process(factor(c(rep(0,1),rep(1,1),rep(2,1),
rep(3,7)), levels=c(0:3)))#0.48 = 3*0.1*0.9+0.7*0.3 CORRECT !

#0/0/0/100 split

gini_process(factor(c(rep(0,0),rep(1,0),rep(2,0),
rep(3,20)), levels=c(0:3)))#0. CORRECT !
**Description**

convenience function to mitigate risk of improperly disentangling train/test

NOTE: the original row names (too dangerous for repeated rows) are not kept but instead recorded in a separate column

**Usage**

```r
InOutBags(RF, data, k, inclRowNames = TRUE, NULLRowNames = TRUE,
              verbose = 0)
```

**Arguments**

- `RF`: object returned by call to `randomForest()`
- `data`: data which was used to train the RF. NOTE: assumes setting of inbag=TRUE while training
- `k`: tree number
- `inclRowNames`: create extra column of original row names
- `NULLRowNames`: if TRUE set row names to NULL
- `verbose`: level of verbosity

**Value**

inbag and outbag subsets of the original data

**Author(s)**

Markus Loecher <Markus.Loecher@gmail.com>

**Examples**

```r
tit = rftitanic(nRows = 200, nodesize=10, ntree = 5)
k=1
```
tmp <- InOutBags(rfTit$RF, rfTit$data, k)

### lpnorm

Compute the Lp norm of a vector.

**Description**

Compute the Lp norm of a vector.

**Usage**

`lpnorm(x, p = 2)`

**Arguments**

- `x`: vector to compute the Lp norm of
- `p`: parameter of p norm

**Value**

Lp norm of a vector or NA

**Author(s)**

Markus Loecher <Markus.Loecher@gmail.com>

**Examples**

```r
lpnorm(1:10)

lpnorm(matrix(1:25, 5, 5))

lpnorm(split(1:25, rep(1:5, each = 5)))

lpnorm(1:10, 1)
```
mlogloss

computes log loss for multiclass problem

Description
computes log loss for multiclass problem

Usage
mlogloss(actual, pred_m, eps = 0.001)

Arguments

- **actual**: integer vector with truth labels, values range from 0 to n - 1 classes
- **pred_m**: predicted probs: column 1 => label 0, column 2 => label 1 and so on
- **eps**: numerical cutoff taken very high
Author(s)

Markus Loecher <Markus.Loecher@gmail.com>

Examples

```r
# require(nnet)

# set.seed(1)

# actual = as.integer(iris$Species) - 1

# fit = nnet(Species ~ ., data = iris, size = 2)

# pred = predict(fit, iris)#note this is a 3-column prediction matrix!

#

# mlogloss(actual, pred) # 0.03967

#library(titanic)

#baseline prediction

#data(titanic_train, package="titanic")

yHat = mean(titanic_train$Survived)#0.383838

mlogloss(titanic_train$Survived, yHat)

#try factors
**Mode**

Computes the mode of an array

## Description
returns the mode of a vector

## Usage

Mode(x)

## Arguments

- **x** vector to find mode of

## Author(s)
Markus Loecher <Markus.Loecher@gmail.com>

## Examples

Mode(rep(letters[1:3],1:3))

Mode(c(TRUE,TRUE,FALSE))

Mode(c(TRUE,TRUE,FALSE,FALSE))
plotVI

creates barplots for variable importances

Description
creates barplots for variable importances

Usage
plotVI(VIbench, order_by = "Gini_OOB", decreasing = TRUE)

Arguments
- VIbench: matrix with importance scores as returned by GiniImportanceForest
- order_by: how to order
- decreasing: which direction to sort

Author(s)
Markus Loecher <Markus.Loecher@gmail.com>

Examples

data("titanic_train", package = "rfVarImpOOB", envir = environment())

set.seed(123)

rannRows=sample(nrow(titanic_train), 300)

data=titanic_train[rannRows,]

RF = randomForest::randomForest(formula = Survived ~ Sex + Pclass + PassengerId,
                                 data=data,
                                 ntree=5,importance=TRUE,
```r
mtry=3,keep.inbag=TRUE,
nodesize = 20)

data$Survived = as.numeric(data$Survived)-1

VI_Titanic = GiniImportanceForest(RF, data, ylab="Survived")

plotVI(VI_Titanic, decreasing = TRUE)
```

---

**Description**

creates barplots for variable importances including permutation scores

**Usage**

```r
plotVI2(VI_bench, decreasing = TRUE, with_MDA = TRUE, ordered_by = "inbag",
        score = "Gini Importance", horizontal = TRUE, fill = "order",
        labelSize = 10, nrow = 3)
```

**Arguments**

- **VI_bench** matrix with importance scores as returned by GiniImportanceForest
- **decreasing** which direction to sort
- **with_MDA** also visualize mean decrease in accuracy (permutation importance)
- **ordered_by** how to order
- **score** type of importance score: Gini, MIA...
- **horizontal** horizontal barplot instead of vertical?
- **fill** fill style for barplots; use e.g. `shQuote("blue")` to pass color strings
- **labelSize** size of axis labels
- **nrow** number of rows of ploztz arrangement
Author(s)

Markus Loecher <Markus.Loecher@gmail.com>

Examples

data("titanic_train", package = "rfVarImpOOB", envir = environment())

set.seed(123)

ranRows = sample(nrow(titanic_train), 300)

data = titanic_train[ranRows,]

RF = randomForest::randomForest(formula = Survived ~ Sex + Pclass + PassengerId,
                                  data = data,
                                  ntree = 5, importance = TRUE,
                                  mtry = 3, keep.inbag = TRUE,
                                  nodesize = 20)

data$Survived = as.numeric(data$Survived) - 1

VI_Titanic = GiniImportanceForest(RF, data, ylab = "Survived")

plotVI2(VI_Titanic, decreasing = TRUE)
preorder2

**Recursive traversal of tree assigning row numbers of data for each node and leaf**

### Description
Recursive calling stops at leaf after which the function propagates back up the tree

### Usage

```r
preorder2(treeRow, bag, tree, verbose = 0)
```

### Arguments
- `treeRow`: current row of tree dataframe to be
- `bag`: The data for the current row
- `tree`: tree (from `randomForest::getTree` to be traversed
- `verbose`: level of verbosity

### Value
- tree with rownames in column node

### Author(s)
Markus Loecher &lt;Markus.Loecher@gmail.com&gt;

### Examples

```r
data("titanic_train", package = "rfVarImpOOB", envir = environment())

set.seed(123)

ranRows=sample(nrow(titanic_train), 300)

RF = randomForest::randomForest(formula = Survived ~ Sex + Pclass + PassengerId,
data=titanic_train[ranRows,],

ntree=5,importance=TRUE,

mtry=3,keep.inbag=TRUE,

nodesize = 1)

k=2

tree = randomForest::getTree(RF, k, labelVar = TRUE)

tree$node=NA

inbag = rep(rownames(RF$inbag),time=RF$inbag[,k])

#trainBag=titanic_train[inbag,]

trainBag=titanic_train[ranRows][inbag,]

tree=preorder2(1,trainBag,tree)

---

rfTitanic  

**fit a random forest model on the titanic data**

---

**Description**

convenience function to reduce overhead of repeatedly fitting RF to titanic data

**Usage**

rfTitanic(formel = Survived ~ Sex + Pclass + PassengerId, nRows = 500,

          ntree = 10, mtry = 3, nodesize = 1)
splitBag

Arguments

- `formel`: formula
- `nRows`: subsample size
- `ntree`: number of trees
- `mtry`: mtry
- `nodesize`: nodesize

Author(s)

Markus Loecher <Markus.Loecher@gmail.com>

Examples

```r
rftit = rfTitanic(nRows = 500, nodesize = 10)
```

Description

The function properly splits on factor levels

Usage

```r
splitBag(treeRow, bag, tree)
```

Arguments

- `treeRow`: current row of tree dataframe to be
- `bag`: The data for the current row
- `tree`: tree (from randomForest::getTree)

Value

- list with elements left_daughter, right_daughter

Author(s)

Markus Loecher <Markus.Loecher@gmail.com>
Description

Titanic train data.

Usage

titanic_train

Format

Data frame with columns

- **PassengerId**: Passenger ID
- **Survived**: Passenger Survival Indicator
- **Pclass**: Passenger Class
- **Name**: Name
- **Sex**: Sex
- **Age**: Age
- **SibSp**: Number of Siblings/Spouses Aboard
- **Parch**: Number of Parents/Children Aboard
- **Ticket**: Ticket Number
- **Fare**: Passenger Fare
- **Cabin**: Cabin
- **Embarked**: Port of Embarkation

Source

https://www.kaggle.com/c/titanic/data

Examples

titanic_train
Index

*Topic datasets
  arabidopsis, 3
  titanic_train, 22

Accuracy, 2
arabidopsis, 3

gini_index, 7
gini_process, 9
GiniImportanceForest, 4
GiniImportanceTree, 5

InOutBags, 11

lpnorm, 12

mlogloss, 13
Mode, 15

plotVI, 16
plotVI2, 17
preorder2, 19

rfTitanic, 20

splitBag, 21

titanic_train, 22