Package ‘fscaret’

October 18, 2016

Type Package
Title Automated Feature Selection from 'caret'
Version 0.9.4.1
Date 2016-10-05
Depends R (>= 3.1.0), caret, gsubfn, hmeasure, utils, parallel
Suggests ada, arm, Boruta, bst, C50, car, caTools, class, Cubist,
e1071, earth (>= 2.2-3), elasticnet, ellipse, evtree,
extraTrees, fastICA, foba, gam, gbm (>= 2.1), glmnet (>= 1.8),
hda, HDclassif, Hmisc, ipred, kernlab, kknn, klaR, kohonen,
KRLS, lars, leaps, LogicReg, MASS, mboost, mda, mgcv, mlbench,
neuralnet, nnet, nodeHarvest, obliqueRF, pamr, partDSA, party
(>= 0.9-999992), penalized, penalizedLDA, pls, pROC, proxy,
qrnn, quantregForest, randomForest, RANN, relaxo, rFerns, rocc,
rpart, rrcov, RRF, rrlida, RSNNS, RWeka (>= 0.4-1), sda,
sparseLDA (>= 0.1-1), spls, stepPlr, superpc
Maintainer Jakub Szlek <j.szlek@uj.edu.pl>
License GPL-2 | GPL-3
Description Automated feature selection using variety of models
provided by 'caret' package.
This work was funded by Poland-Singapore bilateral cooperation
project no 2/3/POL-SIN/2012.
NeedsCompilation no
Repository CRAN
Date/Publication 2016-10-18 23:03:39
Author Jakub Szlek [aut, cre],
Aleksander Mendyk [ctb]

R topics documented:

fscaret-package ........................................... 2
classVarImp ............................................. 3
dataPreprocess .......................................... 4
This package provide fast and automated feature selection based on caret package modeling methods. The main advantage of this extension is that it requires minimum user involvement. Also the variety of used methods in combination with the scaling according to RMSE or MSE obtained from models profit the user. The idea is based on the assumption that the variety of models will balance the roughness of calculations (default model settings are applied). On Windows OS the time limiting function is off, multicore functionality is enabled via parLapply() function of package 'parallel'. Acknowledgments:
This work was funded by Poland-Singapore bilateral cooperation project no 2/3/POL-SIN/2012

Details

Package: fscaret
Type: Package
Version: 0.9.4.1
Date: 2016-10-05
License: GPL-2 | GPL-3

Author(s)

Jakub Szlek <j.szlek@uj.edu.pl> Contributions from Aleksander Mendyk, also stackoverflow and r-help@r-project.org mailing list community.
Maintainer: Jakub Szlek <j.szlek@uj.edu.pl>.
classVarImp

References


See Also

*train, trainControl, rfeControl* by Max Kuhn <Max.Kuhn@pfizer.com> and *predict* base utilities

Arguments

- **model**: Chosed models as called from function fscaret(), argument Used.funcClassPred.
- **xTrain**: Training data set, data frame of input vector
- **yTrain**: Training data set, vector of observed outputs, must be in binary form 0/1.
- **xTest**: Testing data set, data frame of input vector
- **fitControl**: Fitting controls passed to caret function
- **myTimeLimit**: Time limit in seconds for single model fitting
- **no.cores**: Number of used cores for calculations
- **lk_col**: Number of columns for whole data set (inputs + output)
- **supress.output**: If TRUE output of models are supressed.
- **mySystem**: Called from fscaret() result of function .Platform$OS.type

Author(s)

Jakub Szlek and Aleksander Mendyk
References


dataPreprocess

Description

The functionality is realized in two main steps:

1. Check for near zero variance predictors and flag as near zero if:
   (a) the percentage of unique values is less than 20
   (b) the ratio of the most frequent to the second most frequent value is greater than 20,
2. Check for susceptibility to multicollinearity
   (a) Calculate correlation matrix
   (b) Find variables with correlation 0.9 or more and delete them

Usage

dataPreprocess(trainMatryca_nr, testMatryca_nr, labelsFrame, lk_col, lk_row, with.labels)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>trainMatryca_nr</td>
<td>Input training data matrix</td>
</tr>
<tr>
<td>testMatryca_nr</td>
<td>Input testing data matrix</td>
</tr>
<tr>
<td>labelsFrame</td>
<td>Transposed data frame of column names</td>
</tr>
<tr>
<td>lk_col</td>
<td>Number of columns</td>
</tr>
<tr>
<td>lk_row</td>
<td>Number of rows</td>
</tr>
<tr>
<td>with.labels</td>
<td>If with.labels=TRUE, additional data frame with preprocessed inputs corresponding to original data set column numbers as output is generated</td>
</tr>
</tbody>
</table>

Author(s)

Jakub Szlek and Aleksander Mendyk

References

dataPreprocess

Examples

library(fscaret)

# Create data sets and labels data frame
trainMatrix <- matrix(rnorm(150*120,mean=10, sd=1), 150, 120)

# Adding some near-zero variance attributes
temp1 <- matrix(runif(150,0.0001,0.0005), 150, 12)

# Adding some highly correlated attributes
sampleColIndex <- sample(ncol(trainMatrix), size=10)
temp2 <- matrix(trainMatrix[,sampleColIndex]*2, 150, 10)

# Output variable
output <- matrix(rnorm(150,mean=10, sd=1), 150, 1)
trainMatrix <- cbind(trainMatrix,temp1,temp2, output)
colnames(trainMatrix) <- paste("x",c(1:ncol(trainMatrix)),sep="")

# Subset test data set
testMatrix <- trainMatrix[sample(round(0.1*nrow(trainMatrix)))]
labelsDF <- data.frame("Labels"=paste("X",c(1:(ncol(trainMatrix)-1)),sep=""))

lk_col <- ncol(trainMatrix)
lk_row <- nrow(trainMatrix)

with.labels = TRUE

testRes <- dataPreprocess(trainMatrix, testMatrix, 
labelsDF, lk_col, lk_row, with.labels)

summary(testRes)

# Selected attributes after data set preprocessing
testRes$labelsDF

# Training and testing data sets after preprocessing
testRes$trainMatryca
testRes$testMatryca
Example testing data set

dataset.test

Description

The data set after preprocessing, which resulted in 29 inputs. Original data set was obtained in literature survey with 298 inputs. Input: chemical descriptors and characteristics of 8 PLGA microparticles formulation. Output: mean particle size of PLGA microparticles Number of attributes 29, single output.

Usage

data(dataset.test)

Format

data.frame

Details

Literature survey yielded 68 formulations of PLGA microspheres with protein as active pharmacutica ingredient. In vitro release profiles as well as formulation characteristics and composition were derived from articles. Chemical descriptors were obtained using Marvin ChemAxon software (cxcalc plugin). The final data base consisted of 298 inputs and single output mean particle size.

Source


Example training data set

dataset.train

Description
The data set after preprocessing, which resulted in 29 inputs. Original data set was obtained in literature survey with 298 inputs. Input: chemical descriptors and characteristics of 8 PLGA microparticles formulation. Output: mean particle size of PLGA microparticles Number of attributes 29, single output.

Usage
data(dataset.train)

Format
data.frame

Details
Literature survey yielded 68 formulations of PLGA microspheres with protein as active pharmaceutical ingredient. In vitro release profiles as well as formulation characteristics and composition were derived from articles. Chemical descriptors were obtained using Marvin ChemAxon software (cxcalc plugin). The final data base consisted of 298 inputs and single output mean particle size.

Source
Description

Main function for fast feature selection. It utilizes other functions as regPredImp or impCalc to obtain results in a list of data frames.

Usage

```
fscaret(trainDF, testDF, installReqPckg = FALSE, preprocessData = FALSE, 
with.labels = TRUE, classPred = FALSE, regPred = TRUE, skel_outfile = NULL, 
impCalcMet = "RMSE&MSE", myTimelimit = 24 * 60 * 60, Used.funcRegPred = NULL, 
Used.funcClassPred = NULL, no.cores = NULL, method = "boot", returnResamp = "all", 
missData=NULL, supress.output=FALSE, saveModel=FALSE, lvlScale=FALSE, ...)
```

Arguments

- `trainDF`: Data frame of training data set, MISO (multiple input single output) type
- `testDF`: Data frame of testing data set, MISO (multiple input single output) type
- `installReqPckg`: If TRUE prior to calculations it installs all required packages, please be advised to be logged as root (admin) user
- `preprocessData`: If TRUE data preprocessing is performed prior to modeling
- `with.labels`: If TRUE header of the input files are read
- `classPred`: If TRUE classification models are applied. Please be advised that importance is scaled according to F-measure regardless impCalcMet settings.
- `regPred`: If TRUE regression models are applied
- `skel_outfile`: Skeleton output file, e.g. skel_outfile="_myoutput_")
- `impCalcMet`: Variable importance calculation scaling according to RMSE and MSE, for both please enter impCalcMet="RMSE&MSE"
- `myTimelimit`: Time limit in seconds for single model development
- `Used.funcRegPred`: Vector of regression models to be used, for all available models please enter Used.funcRegPred="all"
- `Used.funcClassPred`: Vector of classification models to be used, for all available models please enter Used.funcClassPred="all"
- `no.cores`: Number of cores to be used for modeling, if NULL all available cores are used, should be numeric type or NULL
- `method`: Method passed to fitControl of caret package
- `returnResamp`: Returned resampling method passed to fitControl of caret package
missData: Handling of missing data values. Possible values: "delRow" - delete observations with missing values, "delCol" - delete attributes with missing values, "meanCol" - replace missing values with column mean.

supress.output: If TRUE output of modeling phase by caret functions are supressed. Only info which model is currently calculated and resulting variable importance.

saveModel: Logical value [TRUE/FALSE] if trained model should be embedded in final model.

lvlScale: Logical value [TRUE/FALSE] if additional scaling should be applied. For more information please refer to impCalc().

Additional arguments, preferably passed to fitControl of caret package

Value

$ModelPred: List of outputs from caret model fitting

$VarImp: Data frames of variable importance and corresponding trained models

$PPLabels: Data frame of resulting preprocessed data set with original input numbers and names

$PPTrainDF: Training data set after preprocessing

$PCTestDF: Testing data set after preprocessing

$VarImp$model: Trained models

Note

Be advised when using fscaret function as it requires hard disk operations for saving fitted models and data frames. Files are written in R temp session folder, for more details see tempdir(), getwd() and setwd()

Author(s)

Jakub Szlek and Aleksander Mendyk

References


Examples

if(!(Sys.info()['sysname']!="SunOS"){
  library(fscaret)

  # Load data sets
  data(dataset.train)
  data(dataset.test)

  requiredPackages <- c("R.utils", "gsubfn", "ipred", "caret", "parallel", "MASS")
mySystem <- .Platform$OS.type

if(mySystem == "windows"){
  myCores <- 2
} else {
  myCores <- 2
}

myFirstRES <- fscaret(dataset.train, dataset.test, installReqPckg=FALSE, preprocessData=FALSE, with.labels=TRUE, classPred=FALSE, regPred=TRUE, skel_outfile=NULL, impCalcMet="RMSE&MSE", myTimeLimit=5, Used.funcRegPred=c("lm","pls","pcr"), Used.funcClassPred=NULL, no.cores=myCores, method="boot", returnResamp="all", supress.output=TRUE,saveModel=FALSE)

# Training data set after preprocessing
myFirstRES$PPTrainDF

# Testing data set after preprocessing
myFirstRES$PPTestDF

# Model predictions
myFirstRES$ModelPred

# Variable importance after scaling according to RMSE and MSE
myFirstRES$VarImp

# Reduced input vector (data set) after preprocessing
myFirstRES$PPlabels

---

funcClass.all  

Classification methods used.

Description

Vector of all classification methods used in solving problems by caret

Usage

data(funcClassPred)
**funcReg.all**

**Format**

vector

---

**funcReg.all** *All regression methods used*

---

**Description**

Vector of all regression methods used in solving problems by caret

**Usage**

```r
data(funcRegPred)
```

**Format**

vector

---

**impCalc** *impCalc*

---

**Description**

impCalc function is designed to scale variable importance according to MSE and RMSE calculations. It also stores the raw MSE, RMSE, F-measure and developed models if saveModel=TRUE. impCalc is low-level function, it shouldn’t be used alone unless user has trained models from caret package stored in RData files.

**Usage**

```r
impCalc(skel_outfile, xTest, yTest, lk_col, 
labelsFrame,with.labels,regPred,classPred,saveModel,lv1Scale)
```

**Arguments**

- `skel_outfile`: Skeleton name of output file
- `xTest`: Input vector of testing data set
- `yTest`: Output vector of testing data set
- `lk_col`: Number of columns of whole data set
- `labelsFrame`: Labels to sort variable importance
- `with.labels`: Pass with.labels argument. It is advised to ALWAYS use labels as in some cases VarImp returns importance in descending values. If you insist turning with.labels FALSE, then make sure data base contains pure data and you read it (read.csv) to data.frame with option header=FALSE.
regPred  Indicating if regression predictions are computed. Logical value [TRUE/FALSE].
If regPred is set TRUE, then classPred should be set FALSE.

classPred  Indicating if classification predictions are computed. Possible values TRUE/FALSE.
If classPred is set TRUE, then regPred should be set FALSE. Please be advised that importance is scaled according to F-measure.

saveModel  Logical value [TRUE/FALSE] if trained model should be embedded in final model.

t1vlScale  Indicating if use additional scaling. The option is especially useful when large number of features are getting NA's or are not included in feature ranking. It levels the scores of the features taking the overall number of features. Default value is FALSE. Logical value [TRUE/FALSE].

Details

impCalc function lists RData files in working directory assuming there are only models derived by caret. In a loop function loads models and tries to get the variable importance.

Author(s)

Jakub Szlek and Aleksander Mendyk

Examples

```R
# Hashed to comply with new CRAN check
# library(fscaret)

# Read working directory
myWD <- getwd()

# Set working directory to tmp
setwd(tempdir())

# Load dataset
data(dataset.train)
data(dataset.test)

# Make objects
trainDF <- dataset.train
testDF <- dataset.test
model <- c(“lm”, “pls”, “pcr”)
fitControl <- trainControl(method = “boot”, returnResamp = “all”)
myTimeLimit <- 5
no.cores <- 2
supress.output <- TRUE
skel_outfile <- paste(“_default_”, sep=“”)
mySystem <- .Platform$OS.type
with.labels <- TRUE
redPred <- TRUE
classPred <- FALSE
```
impCalc

```r
saveModel <- FALSE
lvlScale <- FALSE

if(mySystem=="windows"){
  no.cores <- 1
}

# Scan dimensions of trainDF [lk_row x lk_col]
lk_col = ncol(trainDF)
lk_row = nrow(trainDF)

# Read labels of trainDF
labelsFrame <- as.data.frame(colnames(trainDF))
labelsFrame <- cbind(c(1:nrow(trainDF)),labelsFrame)

# Create a train data set matrix
trainMatryca_nr <- matrix(data=NA,nrow=lk_row,ncol=lk_col)

row=0
col=0

for(col in 1:(lk_col)) {
  for(row in 1:(lk_row)) {
    trainMatryca_nr[row,col] <- as.numeric(trainDF[row,col])
  }
}

# Pointing standard data set train
xTrain <- data.frame(trainMatryca_nr[-1,lk_col])
yTrain <- as.vector(trainMatryca_nr[,lk_col])

#------Scan dimensions of testDataFrame1 [lk_row x lk_col]
lk_col_test = ncol(testDF)
lk_row_test = nrow(testDF)

testMatryca_nr <- matrix(data=NA,nrow=lk_row_test,ncol=lk_col_test)

row=0
col=0

for(col in 1:(lk_col_test)) {
  for(row in 1:(lk_row_test)) {
    testMatryca_nr[row,col] <- as.numeric(testDF[row,col])
  }
}

# Pointing standard data set test
xTest <- data.frame(testMatryca_nr[-1,lk_col])
yTest <- as.vector(testMatryca_nr[,lk_col])

# Calling low-level function to create models to calculate on
myVarImp <- regVarImp(model, xTrain, yTrain, xTest,
```

imputeMean

Description
Secondary function imputes the mean to columns with NA data.

Usage
impute.mean(x)

Arguments
x
a vector to calculate mean

Author(s)
Jakub Szlek and Aleksander Mendyk

Examples

library(fscaret)

# Make sample matrix
testData <- matrix(data=rep(1:5), ncol=10, nrow=15)

# Replace random values with NA's
n <- 15
replace <- TRUE
set.seed(1)

rand.sample <- sample(length(testData), n, replace=replace)
testData[rand.sample] <- NA

# Print out input matrix
testData

# Record cols with missing values
missing.colsTestMatrix <- which(colSums(is.na(testData))>0)

for(i in 1:length(missing.colsTestMatrix)){
    testData[,i] <- mean(testData[,i, drop=FALSE])
}
rowToReplace <- missing.colsTestMatrix[i]
testData[,rowToReplace] <- impute.mean(testData[,rowToReplace])

}  

# Print out matrix with replaced NA's by column mean  
testData

installPckg  

Description

Function installs the packages that are listed in data(requiredPackages). The function is called within fscaret function. If argument "installReqPckg = TRUE" the function installs required packages.

Usage

installPckg(requiredPackages)

Arguments

requiredPackages

Vector of packages to be installed

Details

Be advised setting "installReqPckg = TRUE" installs packages in your home directory (.R). To install packages for all users please login as root (admin).

Author(s)

Jakub Szlek and Aleksander Mendyk
**Description**

Function calculates mean squared error as predicted vs. observed

**Usage**

\[
\text{MSE}(\text{vect1}, \text{vect2}, \text{rows\_no})
\]

**Arguments**

- **vect1**: Numeric vector of predicted values
- **vect2**: Numeric vector of observed values
- **rows\_no**: Number of observations

**Author(s)**

Jakub Szlek and Aleksander Mendyk

---

**Description**

The function uses the caret package advantage to perform fitting of numerous regression models.

**Usage**

\[
\text{regVarImp}(\text{model}, \text{xTrain}, \text{yTrain}, \text{xTest}, \\
\text{fitControl}, \text{myTimeLimit}, \text{no\_cores}, \\
\text{lk\_col}, \text{supress\_output}, \text{mySystem})
\]

**Arguments**

- **model**: Chosed models as called from function fscaret(), argument Used.funcRegPred.
- **xTrain**: Training data set, data frame of input vector
- **yTrain**: Training data set, vector of observed outputs
- **xTest**: Testing data set, data frame of input vector
- **fitControl**: Fitting controls passed to caret function
- **myTimeLimit**: Time limit in seconds for single model fitting
- **no\_cores**: Number of used cores for calculations
- **lk\_col**: Number of columns for whole data set (inputs + output)
- **supress\_output**: If TRUE output of models are supressed.
- **mySystem**: Called from fscaret() result of function .Platform$OS.type
requiredPackages

Author(s)

Jakub Szlek and Aleksander Mendyk

References


---

requiredPackages

Description

Character vector of names of required packages to fully take advantage of fscaret

Usage

data(requiredPackages)

Format

vector

Examples

data(requiredPackages)

---

RMSE

Description

Function calculates root mean squared error.

Usage

RMSE(vect1, vect2, rows_no)

Arguments

| vect1  | Numeric vector of predicted values |
| vect2  | Numeric vector of observed values  |
| rows_no| Number of observations             |

Author(s)

Aleksander Mendyk
timeout

Description
This function limits elapsed time spent on single model development. It uses low-level functions of parallel package and sets the fork process with time limit. If the result is not returned within set time, it kills fork. Function shouldn’t be called from R console. The function is not used under Windows OS. Only Unix-like systems have fork functionality.

Usage
timeout(..., seconds)

Arguments

Expression to be time limited

seconds Number of seconds

Author(s)
Original code by Jeroen Ooms <jeroen.ooms at stat.ucla.edu> of OpenCPU package. Modifications by Jakub Szlek and Aleksander Mendyk.
Index

*Topic arith
  MSE, 18
  RMSE, 19
*Topic array
  fscaret, 10
*Topic datasets
  dataset.test, 6
  dataset.train, 8
  funcClass.all, 12
  funcReg.all, 13
  requiredPackages, 19
*Topic data
  installPckg, 17
*Topic design
  impCalc, 13
*Topic error
  timeout, 20
*Topic iteration
  fscaret, 10
*Topic logic
  imputeMean, 16
*Topic math
  imputeMean, 16
  MSE, 18
  RMSE, 19
*Topic methods
  fscaret, 10
  timeout, 20
*Topic models
  classVarImp, 3
  impCalc, 13
  regVarImp, 18
*Topic optimize
  fscaret, 10
*Topic package
  fscaret-package, 2
  installPckg, 17
*Topic robust
  classVarImp, 3
  dataPreprocess, 4
  regVarImp, 18
*Topic univar
  dataPreprocess, 4
  classVarImp, 3
  dataPreprocess, 4
  dataset.test, 6
  dataset.train, 8
  fscaret, 10
  fscaret-package, 2
  funcClass.all, 12
  funcClassPred(funcClass.all), 12
  funcReg.all, 13
  funcRegPred(funcReg.all), 13
  impCalc, 13
  impute.mean(imputeMean), 16
  imputeMean, 16
  installPckg, 17
  MSE, 18
  predict, 3
  regVarImp, 18
  requiredPackages, 19
  rfeControl, 3
  RMSE, 19
  timeout, 20
  train, 3
  trainControl, 3