Package 'preprocomb'

June 26, 2016

Type Package

Title Tools for Preprocessing Combinations

Version 0.3.0

Date 2016-6-26

Author Markus Vattulainen

Maintainer Markus Vattulainen <markus.vattulainen@gmail.com>

Description Preprocessing is often the most time-consuming phase in data analysis and preprocessing transformations interdependent in unexpected ways. This package helps to make preprocessing faster and more effective. It provides an S4 framework for creating and evaluating preprocessing combinations for classification, clustering and outlier detection. The framework supports adding of user-defined preprocessors and preprocessing phases. Default preprocessors can be used for low variance removal, missing value imputation, scaling, outlier removal, noise smoothing, feature selection and class imbalance correction.

License GPL-2

Depends R (>= 2.10)

Imports DMwR, randomForest, caret, clustertend, stats, e1071, methods, utils, arules, zoo, doParallel, foreach

LazyData TRUE

URL https://github.com/mvattulainen/preprocomb

BugReports https://github.com/mvattulainen/preprocomb/issues

Collate '00Utils.R' '01DataClass.R' '02PhaseClass.R' '03PreprocessorClass.R' '04GridClass.R' '05PredictionControlClass.R' '06PreProCombClass.R' '07AnalysisComponent.R' '08DefaultPreprocessorsAndPhases.R'

Suggests kernlab, rpart, testthat, knitr, rmarkdown, ggplot2, lattice, preproviz

VignetteBuilder knitr

RoxygenNote 5.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2016-06-26 09:05:54

R topics documented:

`exampleresult` *preprocomb example*

Description

Modified Iris-data preprocessed with 540 combinations and evaluated with support vector machine classifier.

Usage

```
exampleresult
```

Format

A PreProCombClass object

Details

```
# testdata
set.seed(1)
testdata <- iris
testdata[sample(1:150,40),3] <- NA # add missing values to the third variable
testdata[,4] <- rnorm(150, testdata[,4], 2) # add noise to the fourth variable
testdata$Irrelevant <- runif(150, 0, 1) # add an irrelevant feature


# grid with five phases totalling 540 combinations
examplegrid <- setgrid(phases=c("imputation", "outliers", "scaling", "smoothing", "selection"), data=testdata)

# evaluation of the grid
exampleresult <- preprocomb(grid=examplegrid, models=c("svmRadial"), nholdout=10, cluster=TRUE,
outlier=TRUE, cores=2)
```

`getpreprocessor` *gets preprocessor definition*

Description

gets preprocessor definition

Usage

```
getpreprocessor(preprocessor = NULL)
```

Arguments

`preprocessor`	(character) name of preprocessor, defaults to NULL for list of all preprocessors

Details

getpreprocessor with the name of the preprocessor shows preprocessing function body defined with setpreprocessor().
getpreprocessor without name shows all preprocessors that can be used by functions prepro() and setphase().

Examples

```
getpreprocessor()
getpreprocessor("basicscale")
```

getprogrammaticprediction *parallel computation of classification accuracy holdout rounds*

Description

This function is used internally and exported for package 'metaheur'.

Usage

```
getprogrammaticprediction(preprocesseddataset, predictors, nholdout)
```

Arguments

preprocesseddataset	(DataClass)
predictors	caret models
nholdout	number of holdout rounds

Details

If model tuning fails, NA is returned as classification accuracy of a combination. If model fitting and prediction for holdout round fails, NA is returned for the holdout round.

GridClass-class *container for preprocessor combinations and preprocessed data sets.*

Description

Preprocessing techniques defined with setpreprocessor() can be combined to a phase. Phases defined with setphase() can be combined to a grid of combinations with setgrid(). The main programmatic use with preprocomb() takes a GridClass object as argument.

Details

GridClass is also an interface for extending the system to package 'metaheur', which takes a GridClass object to find near-optiomal combinations fast.

Slots

grid (data frame) preprocessor combinations

data (list) DataClass objects

validation (data frame) validation results

initializedataclassobject

constructor function for creating a DataClass object

Description

initializedataclassobject() is a constructor function for creating a DataClass object. The main use case is adding of new preprocessing techniques to the framework by the user. Added preprocessing techniques (i.e. functions) take as input and must return a DataClass object. See setpreprocessor().

Usage

```
initializedataclassobject(data)
```

Arguments

data	(data.frame)

Details

Argument 'data' must have only numeric columns and one factor column.

Examples

```
## dataobject <- initializedataclassobject(iris)
```

prepro

the MAIN function for interactive use.

Description

prepro() takes data, transforms it according to the given preprocessor and computes statistics of the transformed data. The main use case is the chaining of the preprocessors as show in the examples below.

Usage

```
prepro(dataobject, classname, model = "rpart", nholdout = 2, cores = 1)
```

Arguments

dataobject	(sub class/ data frame/ DataClass) object
classname	(character) name of preprocessor (i.e. PreprocessorClass sub class as defined by setpreprocessor())
model	(character) caret model name, note: the required model library must be attached, defaults to "rpart"
nholdout	(integer) number of holdout rounds used in computation of classification accuracy, must be two or more, defaults to 2
cores	(integer) number of cores used in parallel processing of holdout rounds, defaults to 1

Details

If a data object has missing values, one of the imputation preprocessors must be applied first.

Value

object of PreprocessorClass sub class

Examples

```
## a <- prepro(iris, "basicscale")
## b <- prepro(a, "rfselect75")
## d <- prepro(iris, "basicscale", "rf", nholdout=20, cores=2)
```

PreprocessorClass-class

an abstract class from which concrete preprocessor (sub) classes are inhereted.

Description

Inheritance is controlled by setpreprocessor() function.

Slots

objectname (character) object name

objectoperation (character) operation (expression as character string)

data (DataClass) object

classificationaccuracy (numeric) classification accuracy

hopkinsstatistic (numeric) clustering tendency

ORHskewness (numeric) skewness value of ORH scores

callhistory (character) vector of current and previous calls

preprocessordefinitionstorage *environment for storing preprocessor definitions*

Description

an environment to save and get the preprocessing technique function bodies. Note, this environment is only created for function getpreprocessor().

Usage

```
preprocessordefinitionstorage
```

Format

An object of class `environment` of length 19.

preprocomb *the MAIN function of programmatic use.*

Description

preprocomb executes the computation of classification accuracy, hopkins statistic and ORH outlier score. An alternative to preprocomb is to use package 'metaheur' for faster finding of near-optimal combinations.

Usage

```
preprocomb(models = "rpart", gridclassobject, nholdout = 2,
  searchmethod = "exhaustive", predict = TRUE, cluster = FALSE,
  outlier = FALSE, cores = 1)
```

Arguments

`models`	(character) vector of models (names of models as defined in package caret), defaults to "rpart"
`gridclassobject`	(GridClass) object representing the grid of combinations
`nholdout`	(integer) number of holdout rounds for predictive classification, must be two or more, defaults to two
`searchmethod`	(character) defaults to "exhaustive" full blind search, "random" search 20 percent of grid, "grid" grid search 10 percent
`predict`	(boolean) compute predictions, defaults to TRUE

cluster	(boolean) compute clustering tendency, defaults to FALSE
outlier	(boolean) compute outlier tendency, defaults to FALSE
cores	(integer) number of cores used in parallel processing of holdout rounds, defaults to 1

Details

caret messages will be displayed during processing

Value

a PreProCombClass object

Examples

```
## modifiediris <- droplevels(iris[-c(1:60),])
## grid <- setgrid(phases=c("outliers", "scaling"), data=modifiediris)
## library(kernlab)
## result <- preprocomb(models=c("svmRadial"), grid=grid, nholdout=1, search="grid")
## result@allclassification
## result@allclustering
## result@alloutliers
## result@rawall
## result@catclassification
##
## newphases <- c("outliers", "smoothing", "scaling", "selection", "sampling")
## newmodels <- c("knn", "rf", "svmRadial")
## grid1 <- setgrid(phases=newphases, data=modifiediris)
## result1 <- preprocomb(models=newmodels, grid=grid1, nholdout=1, search="grid")
```

PreProCombClass-class *container for combination evaluation*

Description

This class implements the separation of data used for analysis and analysis of the data. The latter can include computation of association rules as in showrules().

Slots

`rawall` (data frame) all results

`catclassification` (data frame) classification accuracy categorized, "high" is more than 80 percent quantile value

`allclassification` (data frame) classification accuracy, means and standard deviations

`bestclassification` (data frame) best classification accuracy combinations

`allclustering` (data frame) hopkins statistics values

`bestclustering` (data frame) best hopkins statistics combinations

`alloutliers` (data frame) ORH outlier score for 95 percent quantile value

`walltime` (integer) execution time in minutes by wall time (not computation time)

`preprodefault` *seven default phases with preprocessing techniques*

Description

Totals 3200 combinations. preprodefault object can be used as default phases for setgrid().

Usage

```
preprodefault
```

Format

An object of class `character` of length 7.

Examples

```
## grid <- setgrid(preprodefault, iris)
```

`setgrid` *constructor function for creating the combinations*

Description

setgrid takes the preprocessing phases, which contain preprocessors and creates the combinations of them as a grid. It then computes and stores the transformed data sets for each combination. setgrid initializes a GridClass object.

Usage

```
setgrid(phases, data, diagnostics = TRUE)
```

Arguments

`phases`	(character) vector of phases
`data`	(data frame)
`diagnostics`	(logical) run testpreprocessor(), defaults to TRUE

Details

If there are missing values, imputation phase must be set as first phase. Default phase "sampling" can only be used with data, which has binary class labels.

Value

a GridClass object

Examples

```
grid <- setgrid(phases=c("outliers", "selection"), data=iris)
```

`setphase` *constructor function for defining a preprocessing phase.*

Description

Preprocessing phases consist of preprocessing techniques defined with setpreprocessor(). Phases can be defined with setphase() and combined to a grid of combinations with setgrid().

Usage

```
setphase(phasename, preprocessor, preimpute)
```

Arguments

`phasename`	(character) name of the phase
`preprocessor`	(character) vector of preprocessors (see ?setpreprocessor) belonging to the phase
`preimpute`	(logical) whether phase is missing value imputation

Details

All elements of argument 'preprocessor' must point to PreprocessorClass objects constructed with function 'setpreprocessor()'.
If dataset contains missing values, missing value imputation must be the first phase.

Value

a PhaseClass object

Examples

```
## imputation <- setphase("imputation", c("naomit", "meanimpute"), TRUE)
```

setpreprocessor — *constructor function for adding a new preprocessing technique to the system*

Description

The main argument is the operation that is executed to transform the data such as "na.omit(basedata)" for removing rows that have missing values. An operation can process either only the numeric columns or also the class label column.

Usage

```
setpreprocessor(classname, operation)
```

Arguments

classname	(character)
operation	(expression as character string)

Details

Preprocessing techniques defined with setpreprocessor() can be combined to a phase. Phases defined with setphase() can be combined to a grid of combinations with setgrid().

The user-defined S4 class definitions are stored in global environment and thus the function can not be used from an other package.

scaleexample <- function(dataobject) dataobject <- initializedataclassobject(data.frame(x=scale(dataobject@x), dataobject@y)) setpreprocessor("scaleexample", "scaleexample(dataobject)")

Value

NULL, side-effect: definition of S4 class derived from PreprocessorClass and corresponding transformdata-method

showrules — *shows association rules for classification accuracy.*

Description

Classification accuracy label 'high' corresponds to best twenty percent and 'low' for the rest.

Usage

```
showrules(PreProCombClassobject, support = 0.05, confidence = 0.5)
```

Arguments

PreProCombClassobject	(PreProCombClass)
support	(numeric) support for association rules, default to 0.05
confidence	(numeric) confidence for association rules, defaults to 0.5

---

testpreprocessors *test preprocessing techniques against data*

Description

Intended to be used when adding new preprocessing techniques with setpreprocessor().

Usage

```
testpreprocessors(preprocessors = NULL, data = NULL)
```

Arguments

preprocessors	(character) vector of preprocessors, by default gets all preprocessors with getpreprocessors()
data	(data frame) to be tested against, defaults to random data frame without missing values

Examples

```
testpreprocessors()
```

---

transformdata *transformdata*

Description

transformdata is a generic preprocessing function. Its methods are defined by setpreprocessor(). The function is intented for package internal use, but exported so that classes can be inhereted from it.

Usage

```
transformdata(object, dataobject)
```

Arguments

object	(PreprocessorClass) object
dataobject	(DataClass/data frame) object

Index