Package 'logiBin'

May 8, 2026

Type Package

Title Binning Variables to Use in Logistic Regression

Version 0.3

Author Sneha Tody

Maintainer Sneha Tody <sn.tody1@gmail.com>

Description Fast binning of multiple variables using parallel processing. A summary of all the variables binned is generated which provides the information value, entropy, an indicator of whether the variable follows a monotonic trend or not, etc. It supports rebinning of variables to force a monotonic trend as well as manual binning based on pre specified cuts. The cut points of the bins are based on conditional inference trees as implemented in the partykit package. The conditional inference framework is described by Hothorn T, Hornik K, Zeileis A (2006) <doi:10.1198/106186006X133933>.

License GPL-2

Encoding UTF-8

LazyData true

Imports partykit, doParallel, data.table, foreach, iterators, parallel, stats

Suggests knitr, rmarkdown

RoxygenNote 6.0.1

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2018-05-21 15:07:46 UTC

Contents

`binTest` *Checking the performance of the bins created on test data*

Description

This function uses parallel processing to replicate the bins on test data. This can be used to check the stability of the variable.

Usage

```
binTest(binObj, testDf, y, xVars, nCores = 1)
```

Arguments

`binObj`	- An object returned by getBins or any other function (except createBins) in this package
`testDf`	- A data frame containing the test data
`y`	- The name of the dependent variable
`xVars`	- A vector names of variables which are to be tested
`nCores`	- The number of cores used for parallel processing. The default value is 1

Value

Returns a list containing 2 elements. The first is a data frame called varSummary which contains a summary of the performance of the variables on the test data including their IV value, entropy, flag which indicates if bad rate increases/decreases with variable value, flag to indicate if a monotonic trend is present, number of bins which flip (i.e. do not follow a monotonic trend), number of bins of the variable and a flag to indicate whether it includes pure nodes (node which do not have any defaults). The second element is a data frame called bin which contains details of all the bins of the variables.

Examples

```
b1 <- getBins(loanData, "bad_flag", c('LTV', 'balance'))
b2 <- binTest(b1, loanData[1:50,], "bad_flag", c('LTV', 'balance'))
```

createBins *Add binned variables to data*

Description

This function creates a data frame with binned variables

Usage

```
createBins(binObj, df, xVars, prefix = "b_")
```

Arguments

binObj	- An object returned by getBins or any other function in this package
df	- A data frame
xVars	- A vector of names of variables for which bins have to be created
prefix	- The prefix to be added to the variable name to create the new variable. Default value is b_

Value

Returns a dataframe which adds the binned variables to the original data frame

Examples

```
b1 <- getBins(loanData, "bad_flag", c('age', 'score', 'balance'), minCr=0.8)
loanData <- createBins(b1, loanData, c('age', 'balance'))
```

forceDecrTrend *Force a numerical variable to follow a monotonically decreasing trend*

Description

This function forces a variable to follow a monotonically decreasing trend by grouping bins. In case such a trend can not be forced a message is printed to the console

Usage

```
forceDecrTrend(binObj, xVars)
```

Arguments

binObj	- An object returned by getBins or any other function (except createBins) in this package
xVars	- A vector of the name of variables

Value

Returns a list containing 3 objects. Similar to the getBins function

Examples

```
b1 <- getBins(loanData, "bad_flag", c('age', 'score'), minCr=0.6, minProp = 0.01)
b1 <- forceDecrTrend(b1, c('score','age'))
```

`forceIncrTrend` *Force a numerical variable to follow a monotonically increasing trend*

Description

This function forces a variable to follow a monotonically increasing trend by grouping bins. In case such a trend can not be forced a message is printed to the console

Usage

```
forceIncrTrend(binObj, xVars)
```

Arguments

`binObj`	- An object returned by getBins or any other function (except createBins) in this package
`xVars`	- A vector of the name of variables

Value

Returns a list containing 3 objects. Similar to the getBins function

Examples

```
b1 <- getBins(loanData, "bad_flag", c('age', 'score'), minCr=0.6, minProp = 0.01)
b1 <- forceIncrTrend(b1, c('score','age'))
```

getBins *Bins variables to be used in logistic regression*

Description

This function uses parallel processing to compute bins for continuous and categorical variables. The splits are computed using the partykit package which uses conditional inferencing trees. Refer to the package documentation for more details. A separate bin is created for NA values. This can be combined using naCombine function. Categorical variables with a maximum of 10 distinct values are supported.

Usage

```
getBins(df, y, xVars, minProp = 0.03, minCr = 0.9, nCores = 1)
```

Arguments

df	- A data frame
y	- The name of the dependent variable
xVars	- A vector names of variables
minProp	- The minimum proportion of observations that must be exceeded in order to implement a split. Default value is 0.03
minCr	- The value of test statistic that must be exceeded in order to implement a split. Increasing this value will decrease the number of splits. Refer to the partykit package documentation for more details. Default value is 0.9
nCores	- The number of cores used for parallel processing. The default value is 1

Value

Returns a list containing 3 elements. The first is a data frame called varSummary which contains a summary of all the variables along with their IV value, entropy, p value from ctree function in partykit package, flag which indicates if bad rate increases/decreases with variable value, flag to indicate if a monotonic trend is present, number of bins which flip (i.e. do not follow a monotonic trend), number of bins of the variable and a flag to indicate whether it includes pure nodes (node which do not have any defaults). The second element is a data frame called bin which contains details of all the bins of the variables. The third element is a dataframe called err which contains details of all the variables that could not be split and the reason for the same.

Examples

```
b1 <- getBins(loanData, "bad_flag", c('age', 'score', 'balance'))
```

loanData *Simulated default data of 100 customers*

Description

A dataset containing simulated data about the characteristic of a customer applying for a loan. The dependent variable is "bad_flag" which indicates whether the customer defaults or not.

Usage

```
loanData
```

Format

A data frame with 100 rows and 6 variables:

bad_flag Indicates whether a customer has defaulted or not

age Age of the customer

LTV Ratio of amount of loan to amount of collateral

location Indicates the location of the customer

balance Account balance

score credit score of the customer

manualSplit *Split a variable based on specified cuts*

Description

This function splits variables based on cuts that have been input manually

Usage

```
manualSplit(binObj, splitVar, y, splits, df)
```

Arguments

binObj	- An object returned by getBins or any other function (except createBins) in this package
splitVar	- The name of the variable that has to be split
y	- The dependent variable
splits	- The splits for the variable
df	- A data frame

Value

Returns a list containing 3 objects. Similar to the getBins function

Examples

```
b1 <- getBins(loanData, "bad_flag", c('age', 'score', 'balance'), minCr=0.8)
b1 <- manualSplit(b1, 'age', 'bad_flag', c(25,40,55), loanData)
```

---

naCombine *Combine NA bins*

Description

This function combines the NA bin with either the bin having the closest bad rate or the average bad rate if the count of observations in NA bin is low

Usage

```
naCombine(binObj, xVars, cutoffPropn = 0.01)
```

Arguments

binObj	- An object returned by getBins or other functions (except createBins) in this package
xVars	- A vector of names of variables for which NA bins have to be combined
cutoffPropn	- The minimum proportion of observations that must be present in the NA bin for it to be combined with the bin with closest bad rate. If the proportion s below this, the NA bin will be combined with bin having average bad rate

Value

Returns a list containing 3 objects. Similar to the getBins function

Examples

```
b1 <- getBins(loanData, "bad_flag", c('age', 'score', 'LTV'))
b1 <- naCombine(b1, c('LTV'))
```

Index