Package 'funModeling'

November 12, 2016

Type Package

Title Learn Data Science Through the ``Data Science Live Book''

Description Around 10% of almost any predictive modeling project is spent in predictive modeling, 'funModeling' and the book <http://livebook.datascienceheroes.com/> are intended to cover remaining 90%: data preparation, profiling, selecting best variables 'dataViz', assessing model performance and other functions.

Version 1.5

Date 2016-11-11

Author Pablo Casas

Maintainer Pablo Casas <pabloc@datascienceheroes.com>

License GPL-2

BugReports https://github.com/pablo14/funModeling/issues

URL livebook.datascienceheroes.com

LazyData true

Imports ROCR, ggplot2, gridExtra, pander, reshape2, scales, dplyr, lazyeval, utils

Depends R (>= 3.2.1), Hmisc (>= 3.17.1)

RoxygenNote 5.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2016-11-12 15:47:09

R topics documented:

correlation_table 2
cross_plot 3
df_status 4
equal_freq 4
filter_vars 5
freq 6

gain_lift 6
get_sample 7
heart_disease 8
model_performance 8
plotar 9
prep_outliers 9
range01 11
v_compare 11

Index **13**

`correlation_table` *Get correlation against target variable*

Description

Obtain correlation table for all variables against target variable. Only numeric variables are analyzed (factor/character are skippted automatically).

Usage

```
correlation_table(data, str_target)
```

Arguments

`data`	data frame
`str_target`	string variable to predict

Value

Correlation index for all data input variable

Examples

```
correlation_table(data=heart_disease, str_target="has_heart_disease")
```

cross_plot *Cross-plotting input variable vs. target variable*

Description

The cross_plot shows how the input variable is correlated with the target variable, getting the likelihood rates for each input's bin/bucket .

Usage

```
cross_plot(data, str_input, str_target, path_out, auto_binning,
  plot_type = "both")
```

Arguments

data	data frame source
str_input	string input variable (if empty, it runs for all numeric variable), it can take a single character value or a character vector.
str_target	string of the variable to predict
path_out	path directory, if it has a value the plot is saved
auto_binning	indicates the automatic binning of str_input variable based on equal frequency (function 'equal_freq'), default value=TRUE
plot_type	indicates if the output is the 'percentual' plot, the 'quantity' or 'both' (default).

Value

cross plot

Examples

```
## Example 1:
cross_plot(data=heart_disease, str_input="chest_pain", str_target="has_heart_disease")

## Example 2: Disabling auto_binning:
cross_plot(data=heart_disease, str_input="oldpeak",
str_target="has_heart_disease", auto_binning=FALSE)

## Example 3: Saving the plot into a folder:
cross_plot(data=heart_disease, str_input="oldpeak",
str_target="has_heart_disease", path_out = "my_folder")

## Example 4: Running with multiple input variables at the same time:
cross_plot(data=heart_disease, str_input=c("age", "oldpeak", "max_heart_rate"),
str_target="has_heart_disease")
```

df_status *Get a summary for the given data frame.*

Description

For each variable it returns: Quantity and percentage of zeros (q_zeros and p_zeros respectevly). Same metrics for NA values (q_NA/p_na), and infinite values (q_inf/p_inf). Last two columns indicates data type and quantity of unique values. This function print and return the results.

Usage

```
df_status(data, print_results)
```

Arguments

data	data frame
print_results	if FALSE then there is not a print in the console, TRUE by default.

Value

Metrics data frame

Examples

```
df_status(heart_disease)
```

equal_freq *Equal frequency binning*

Description

Equal frequency tries to put the same quantity of cases per bin when possible. It's a wrapper of function cut2 from Hmisc package.

Usage

```
equal_freq(var, n_bins)
```

Arguments

var	input variable
n_bins	number of bins to split 'var' by equal frequency, if it not possible to calculate for the desired bins, it returns the closest number

Value

The binned variable.

Examples

```
## Example 1
summary(heart_disease$age)
age_2=equal_freq(var=heart_disease$age, n_bins = 10)
summary(age_2)

## Example 2
age_3=equal_freq(var=heart_disease$age, n_bins = 5)
summary(age_3)
```

---

filter_vars *Filtering variables by string name*

Description

Based on the variables name present in 'str_input', it returns the original data frame (keep=T), or it deletes all except the desired ones.

Usage

```
filter_vars(data, str_input, keep = TRUE)
```

Arguments

data	data frame
str_input	string vector containing variables to delete or to keep
keep	boolean indicating if the variables names in 'str_input' must be kept or

Value

Filtered data frame

Examples

```
# Selecting variables
my_data_1=filter_vars(mtcars, str_input=c('mpg', 'cyl'))
colnames(my_data_1)

# Deleting all except desired variables
my_data_2=filter_vars(mtcars, str_input=c('mpg', 'cyl', 'qsec', 'vs'), keep=FALSE)
colnames(my_data_2)
```

freq *Frequency table for categorical variables*

Description

Retrieves the frequency and percentage for str_input

Usage

```
freq(data, str_input, plot = T, path_out)
```

Arguments

data	input data containing the variable to describe
str_input	string input variable (if empty, it runs for all numeric variable), it can take a single character value or a character vector.
plot	flag indicating if the plot is desired, TRUE by default
path_out	path directory, if it has a value the plot is saved

Value

vector with the values scaled into the 0 to 1 range

Examples

```
freq(data=heart_disease, str_input = c('thal','chest_pain'))
```

gain_lift *Generates lift and cumulative gain performance table and plot*

Description

It retrieves the cumulative positive rate -gain curve- and the lift chart & plot when score is divided in 5, 10 or 20 segments. Both metrics give a quality measure about how well the model predicts. Higher values at the beginning of the population implies a better model.

Usage

```
gain_lift(data, str_score, str_target, q_segments)
```

Arguments

data	input data source
str_score	the variable which contains the score number, or likelihood of being positive class
str_target	target binary variable indicating class label
q_segments	quantity of segments to split str_score, valid values: 5, 10 or 20

Value

lift/gain table, column: gain implies how much positive cases are catched if the cut point to define the positive class is set to the column "Score Point"

Examples

```
fit_glm=glm(has_heart_disease ~ age + oldpeak, data=heart_disease, family = binomial)
heart_disease$score=predict(fit_glm, newdata=heart_disease, type='response')
gain_lift(data=heart_disease,str_score='score',str_target='has_heart_disease')
```

---

get_sample *Sampling training and test data*

Description

Split input data into training and test set, retrieving always same sample by setting the seed.

Usage

```
get_sample(data, percentage_tr_rows = 0.8, seed = 987)
```

Arguments

data	input data source
percentage_tr_rows	percentage of training rows, range value from 0.1 to 0.99, default value=0.8 (80 percent of training data)
seed	to generate the sample randomly, default value=987

Value

TRUE/FALSE vector same length as 'data' param. TRUE represents that row position is for training data

Examples

```
## Training and test data. Percentage of training cases default value=80%.
index_sample=get_sample(data=heart_disease, percentage_tr_rows=0.8)
## Generating the samples
data_tr=heart_disease[index_sample,]
data_ts=heart_disease[-index_sample,]
```

heart_disease *Heart Disease Data*

Description

There are variables related to patient clinic trial. The variable to predict is 'has_heart_disease'.

Usage

```
heart_disease
```

Format

A data frame with 303 rows and 16 variables:
https://archive.ics.uci.edu/ml/datasets/Heart+Disease

model_performance *Get model perfomance metrics (KS, AUC and ROC)*

Description

Get model performance for tree models (rpart library), or glm. It returns quality metrics: AUC (Area Under ROC Curve) and KS (Kolmogorov-Smirnov), and plots the ROC curve.

Usage

```
model_performance(fit, data, target_var)
```

Arguments

fit	model, it could be any of the following: decision tree from rpart package, glm model or randomForest.
data	data frame used to build the model. Also it supports data for testing, (it has to contain same columns as training data.
target_var	It's the name of the column used as target/outcome. It's an string value, write it between apostrohpe.

Value

None.

Examples

```
fit_glm=glm(has_heart_disease ~ age + oldpeak, data=heart_disease, family = binomial)
model_performance(fit=fit_glm, data = heart_disease, target_var = "has_heart_disease")
```

plotar *Correlation plots*

Description

Visual correlation analysis. Plot different graphs in order to expose the inner information of any numeric variable against the target variable

Usage

```
plotar(data, str_input, str_target, plot_type, path_out)
```

Arguments

data	data frame source
str_input	string input variable (if empty, it runs for all numeric variable), it can take a single character value or a character vector.
str_target	string of the variable to predict
plot_type	Indicates the type of plot to retrieve, available values: "boxplot" or "histdens".
path_out	path directory, if it has a value the plot is saved

Value

Single or multiple plots specified by 'plot_type' parameter

Examples

```
## Run for all numeric variables
plotar(data=heart_disease, str_target="has_heart_disease",
plot_type="histdens")
```

prep_outliers *Outliers Data Preparation*

Description

Deal with outliers by setting an 'NA value' or by 'stopping' them at a certain. The parameters: 'top_percent'/'bottom_percent' are used to consider a value as outlier.

Setting NA is recommended when doing statistical analysis, parameter: type='set_na'. Stopping is recommended when creating a predictive model without biasing the result due to outliers, parameter: type='stop'.

Automatization: 'prep_outliers' skip all factor/char columns, so it can receive a whole data frame, removing outliers by finally, returning a the cleaned data.

Usage

```
prep_outliers(data, str_input, type = c("stop", "set_na"), top_percent,
  bottom_percent)
```

Arguments

data	data frame
str_input	string input variable (if empty, it runs for all numeric variable).
type	can be 'stop' or 'set_na', in the first case the original variable is stopped at the desiered percentile, 'set_na' sets NA to the same values.
top_percent	value from 0 to 1, represents the highest X percentage of values to treat
bottom_percent	value from 0 to 1, represents the lowest X percentage of values to treat

Value

A data frame with the desired outlier transformation

Examples

```
# Creating data frame with outliers
set.seed(10)
df=data.frame(var1=rchisq(1000,df = 1), var2=rnorm(1000))
df=rbind(df, 1135, 2432) # forcing outliers
df$id=as.character(seq(1:1002))

# for var1: mean is ~ 4.56, and max 2432
summary(df)

#########################################################
### PREPARING OUTLIERS FOR DESCRIPTIVE STATISTICS
#########################################################

#### EXAMPLE 1: Removing top 1% for a single variable
# checking the value for the top 1% of highest values (percentile 0.99), which is ~ 7.05
quantile(df$var1, 0.99)

# Setting type='set_na' sets NA to the highest value)
var1_treated=prep_outliers(data = df,  str_input = 'var1', type='set_na', top_percent  = 0.01)

# now the mean (~ 0.94) is more accurate, and note that: 1st, median and 3rd
#  quartiles remaining very similar to the original variable.
summary(var1_treated)

#### EXAMPLE 2: if 'str_input' is missing, then it runs for all numeric variables
#  (which have 3 or more distinct values).
df_treated2=prep_outliers(data = df, type='set_na', top_percent  = 0.01)
summary(df_treated2)

#### EXAMPLE 3: Removing top 1% (and bottom 1%) for 'N' specific variables.
vars_to_process=c('var1', 'var2')
```

```
df_treated3=prep_outliers(data = df, str_input = vars_to_process, type='set_na',
 bottom_percent = 0.01, top_percent  = 0.01)
summary(df_treated3)

########################################################
### PREPARING OUTLIERS FOR PREDICTIVE MODELING
########################################################

#### EXAMPLE 4: Stopping outliers at the top 1% value for all variables. For example
#   if the top 1% has a value of 7, then all values above will be set to 7. Useful
#   when modeling because outlier cases can be used.
df_treated4=prep_outliers(data = df, top_percent  = 0.01, type='stop')
```

`range01` *Transform a variable into the [0-1] range*

Description

Range a variable into [0-1], assigning 0 to the min and 1 to the max of the input variable. All NA values will be removed.

Usage

```
range01(var)
```

Arguments

`var`	numeric input vector

Value

vector with the values scaled into the 0 to 1 range

Examples

```
range01(mtcars$cyl)
```

`v_compare` *Compare two vectors*

Description

Obtaing coincident and not coincident elements between two vectors.

Usage

```
v_compare(vector_x, vector_y)
```

Arguments

vector_x	1st vector to compare
vector_y	2nd vector to compare

Value

Correlation index for all data input variable

Examples

```
v1=c("height","weight","age")
v2=c("height","weight","location","q_visits")
res=v_compare(vector_x=v1, vector_y=v2)
# Print the keys that didn't match
res
# Accessing the keys not present in
```

Index

∗Topic **datasets**
heart_disease, 8

correlation_table, 2
cross_plot, 3

df_status, 4

equal_freq, 4

filter_vars, 5
freq, 6

gain_lift, 6
get_sample, 7

heart_disease, 8

model_performance, 8

plotar, 9
prep_outliers, 9

range01, 11

v_compare, 11