Package 'scorecard'

June 12, 2018

Type Package

Title Credit Risk Scorecard

Description Makes the development of credit risk scorecard easily and efficiently by providing functions such as information value, variable filter, optimal woe binning, scorecard scaling and performance evaluation etc. The references including:
1. Refaat, M. (2011, ISBN: 9781447511199). Credit Risk Scorecard: Development and Implementation Using SAS.
2. Siddiqi, N. (2006, ISBN: 9780471754510). Credit risk scorecards. Developing and Implementing Intelligent Credit Scoring.

Version 0.1.8

Author Shichen Xie

Maintainer Shichen Xie <xie@shichen.name>

Depends R (>= 3.1.0)

Imports data.table (>= 1.10.0), ggplot2, gridExtra, foreach, doParallel, parallel

URL https://github.com/ShichenXie/scorecard

BugReports https://github.com/ShichenXie/scorecard/issues

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 6.0.1

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2018-06-12 15:08:12 UTC

R topics documented:

`germancredit` *German Credit Data*

Description

Credit data that classifies debtors described by a set of attributes as good or bad credit risks. See source link below for detailed information.

Usage

```
data(germancredit)
```

Format

A data frame with 21 variables (numeric and factors) and 1000 observations.

Source

https://archive.ics.uci.edu/ml/datasets/Statlog+(German+Credit+Data)

Examples

```
# Load German credit data and create subset
data(germancredit)
df = germancredit[, c('creditability', 'credit.amount', 'duration.in.month',
                  'savings.account.and.bonds', 'purpose')]
# Display structure of the subset (data frame)
str(df)
```

iv *Information Value*

Description

This function calculates information value (IV) for multiple x variables.

Usage

```
iv(dt, y, x = NULL, positive = "bad|1", order = TRUE)
```

Arguments

dt	A data frame with both x (predictor/feature) and y (response/label) variables.
y	Name of y variable.
x	Name of x variables. Default is NULL. If x is NULL, then all variables except y are counted as x variables.
positive	Value of positive class, default is "bad|1".
order	Logical, default is TRUE. If it is TRUE, the output will descending order via iv.

Details

IV is a very useful concept for variable selection while developing credit scorecards. The formula for information value is shown below:

$$IV = \sum(DistributionBad_i - DistributionGood_i) * \ln(\frac{DistributionBad_i}{DistributionGood_i}).$$

The log component in information value is defined as weight of evidence (WOE), which is shown as

$$WeightofEvidence = \ln(\frac{DistributionBad_i}{DistributionGood_i}).$$

The relationship between information value and predictive power is as follows: <0.02 (useless for prediction), 0.02 to 0.1 (Weak predictor), 0.1 to 0.3 (Medium predictor), 0.3 to 0.5 (Strong predictor) and >0.5 (Suspicious or too good to be true).

Value

Information Value

Examples

```
# Load German credit data
data(germancredit)

# information values
dt_info_value = iv(germancredit, y = "creditability")
```

perf_eva *KS, ROC, Lift, PR*

Description

perf_eva provides performance evaluations, such as kolmogorov-smirnow(ks), ROC, lift and precision-recall curves, based on provided label and predicted probability values.

Usage

```
perf_eva(label, pred, title = NULL, groupnum = NULL, type = c("ks",
  "roc"), show_plot = TRUE, positive = "bad|1", seed = 186)
```

Arguments

label	Label values, such as 0s and 1s, 0 represent for good and 1 for bad.
pred	Predicted probability or score.
title	Title of plot, default is NULL.
groupnum	The group number when calculating KS. Default NULL, which means the number of sample size.
type	Types of performance plot, such as "ks", "lift", "roc", "pr". Default c("ks", "roc").
show_plot	Logical value, default is TRUE. It means whether to show plot.
positive	Value of positive class, default is "bad|1".
seed	Integer, default is 186. The specify seed is used for random sorting data.

Details

Accuracy = true positive and true negative/total cases

Error rate = false positive and false negative/total cases

TPR, True Positive Rate(Recall or Sensitivity) = true positive/total actual positive

PPV, Positive Predicted Value(Precision) = true positive/total predicted positive

TNR, True Negative Rate(Specificity) = true negative/total actual negative = 1-FPR

NPV, Negative Predicted Value = true negative/total predicted negative

Value

ks, roc, lift, pr

See Also

perf_psi

Examples

```
## Not run:
# load germancredit data
data("germancredit")

# filter variable via missing rate, iv, identical value rate
dt_sel = var_filter(germancredit, "creditability")

# woe binning ------
bins = woebin(dt_sel, "creditability")
dt_woe = woebin_ply(dt_sel, bins)

# glm ------
m1 = glm( creditability ~ ., family = binomial(), data = dt_woe)
# summary(m1)

# Select a formula-based model by AIC
m_step = step(m1, direction="both", trace=FALSE)
m2 = eval(m_step$call)
# summary(m2)

# predicted proability
dt_pred = predict(m2, type='response', dt_woe)

# performance ------
# Example I # only ks & auc values
perf_eva(dt_woe$creditability, dt_pred, show_plot=FALSE)

# Example II # ks & roc plot
perf_eva(dt_woe$creditability, dt_pred)

# Example III # ks, lift, roc & pr plot
perf_eva(dt_woe$creditability, dt_pred, type = c("ks","lift","roc","pr"))

## End(Not run)
```

`perf_psi` *PSI*

Description

`perf_psi` calculates population stability index (PSI) and provides credit score distribution based on credit score datasets.

Usage

```
perf_psi(score, label = NULL, title = NULL, x_limits = NULL,
  x_tick_break = 50, show_plot = TRUE, seed = 186,
  return_distr_dat = FALSE)
```

Arguments

score	A list of credit score for actual and expected data samples. For example, score = list(actual = score_A, expect = score_E), both score_A and score_E are dataframes with the same column names.
label	A list of label value for actual and expected data samples. The default is NULL. For example, label = list(actual = label_A, expect = label_E), both label_A and label_E are vectors or dataframes. The label values should be 0s and 1s, 0 represent for good and 1 for bad.
title	Title of plot, default is NULL.
x_limits	x-axis limits, default is NULL.
x_tick_break	x-axis ticker break, default is 50.
show_plot	Logical, default is TRUE. It means whether to show plot.
seed	Integer, default is 186. The specify seed is used for random sorting data.
return_distr_dat	Logical, default is FALSE.

Details

The population stability index (PSI) formula is displayed below:

$$PSI = \sum((Actual\% - Expected\%) * (\ln(\frac{Actual\%}{Expected\%}))).$$

The rule of thumb for the PSI is as follows: Less than 0.1 inference insignificant change, no action required; 0.1 - 0.25 inference some minor change, check other scorecard monitoring metrics; Greater than 0.25 inference major shift in population, need to delve deeper.

Value

a dataframe of psi & plots of credit score distribution

See Also

perf_eva

Examples

```
## Not run:
# load germancredit data
data("germancredit")

# filter variable via missing rate, iv, identical value rate
dt_sel = var_filter(germancredit, "creditability")

# breaking dt into train and test ------
dt_list = split_df(dt_sel, "creditability", ratio = 0.6, seed=21)
dt_train = dt_list$train; dt_test = dt_list$test
```

```
# woe binning ------
bins = woebin(dt_train, "creditability")

# converting train and test into woe values
train = woebin_ply(dt_train, bins)
test = woebin_ply(dt_test, bins)

# glm ------
m1 = glm(creditability ~ ., family = binomial(), data = train)
# summary(m1)

# Select a formula-based model by AIC
m_step = step(m1, direction="both", trace=FALSE)
m2 = eval(m_step$call)
# summary(m2)

# predicted proability
train_pred = predict(m2, type='response', train)
test_pred = predict(m2, type='response', test)

# # ks & roc plot
# perf_eva(train$creditability, train_pred, title = "train")
# perf_eva(test$creditability, test_pred, title = "test")

#' # scorecard
card = scorecard(bins, m2)

# credit score, only_total_score = TRUE
train_score = scorecard_ply(dt_train, card)
test_score = scorecard_ply(dt_test, card)

# Example I # psi
psi = perf_psi(
  score = list(train = train_score, test = test_score),
  label = list(train = train$creditability, test = test$creditability)
)
# psi$psi  # psi dataframe
# psi$pic  # pic of score distribution

# Example II # specifying score range
psi_s = perf_psi(
  score = list(train = train_score, test = test_score),
  label = list(train = train$creditability, test = test$creditability),
  x_limits = c(200, 750),
  x_tick_break = 50
  )

# Example III # credit score, only_total_score = FALSE
train_score2 = scorecard_ply(dt_train, card, only_total_score=FALSE)
test_score2 = scorecard_ply(dt_test, card, only_total_score=FALSE)

# psi
psi2 = perf_psi(
```

```
  score = list(train = train_score2, test = test_score2),
  label = list(train = train$creditability, test = test$creditability)
)
# psi2$psi  # psi dataframe
# psi2$pic  # pic of score distribution

## End(Not run)
```

---

scorecard *Creating a Scorecard*

Description

scorecard creates a scorecard based on the results from woebin and glm.

Usage

```
scorecard(bins, model, points0 = 600, odds0 = 1/19, pdo = 50,
  basepoints_eq0 = FALSE)
```

Arguments

bins	Binning information generated from woebin function.
model	A glm model object.
points0	Target points, default 600.
odds0	Target odds, default 1/19. Odds = p/(1-p).
pdo	Points to Double the Odds, default 50.
basepoints_eq0	Logical, default is FALSE. If it is TRUE, the basepoints will equally distribute to each variable.

Value

scorecard

See Also

scorecard_ply

Examples

```
## Not run:
# load germancredit data
data("germancredit")

# filter variable via missing rate, iv, identical value rate
dt_sel = var_filter(germancredit, "creditability")
```

```
# woe binning ------
bins = woebin(dt_sel, "creditability")
dt_woe = woebin_ply(dt_sel, bins)

# glm ------
m = glm(creditability ~ ., family = binomial(), data = dt_woe)
# summary(m)

# Select a formula-based model by AIC
m_step = step(m, direction="both", trace=FALSE)
m = eval(m_step$call)
# summary(m)

# predicted proability
# dt_pred = predict(m, type='response', dt_woe)

# performace
# ks & roc plot
# perf_eva(dt_woe$creditability, dt_pred)

# scorecard
# Example I # creat a scorecard
card = scorecard(bins, m)

# credit score
# Example I # only total score
score1 = scorecard_ply(dt, card)

# Example II # credit score for both total and each variable
score2 = scorecard_ply(dt, card, only_total_score = F)

## End(Not run)
```

`scorecard_ply` *Score Transformation*

Description

`scorecard_ply` calculates credit score using the results from `scorecard`.

Usage

```
scorecard_ply(dt, card, only_total_score = TRUE, print_step = 1L)
```

Arguments

`dt`	Original data
`card`	Scorecard generated from `scorecard`.

only_total_score
: Logical, default is TRUE. If it is TRUE, then the output includes only total credit score; Otherwise, if it is FALSE, the output includes both total and each variable's credit score.

print_step
: A non-negative integer. Default is 1. If print_step>0, print variable names by each print_step-th iteration. If print_step=0, no message is print.

Value

Credit score

See Also

scorecard

Examples

```
## Not run:
# load germancredit data
data("germancredit")

# filter variable via missing rate, iv, identical value rate
dt_sel = var_filter(germancredit, "creditability")

# woe binning ------
bins = woebin(dt_sel, "creditability")
dt_woe = woebin_ply(dt_sel, bins)

# glm ------
m = glm(creditability ~ ., family = binomial(), data = dt_woe)
# summary(m)

# Select a formula-based model by AIC
m_step = step(m, direction="both", trace=FALSE)
m = eval(m_step$call)
# summary(m)

# predicted proability
# dt_pred = predict(m, type='response', dt_woe)

# performace
# ks & roc plot
# perf_eva(dt_woe$creditability, dt_pred)

# scorecard
# Example I # creat a scorecard
card = scorecard(bins, m)

# credit score
# Example I # only total score
score1 = scorecard_ply(dt, card)
```

```
# Example II # credit score for both total and each variable
score2 = scorecard_ply(dt, card, only_total_score = F)

## End(Not run)
```

`split_df` *Split a dataset*

Description

Split a dataset

Usage

```
split_df(dt, y = NULL, ratio = 0.7, seed = 186)
```

Arguments

`dt`	A data frame.
`y`	Name of y variable, default is NULL. The input data will split based on the predictor y, if it is provide.
`ratio`	A numeric value, default is 0.7. It indicates the ratio of total rows contained in one split, must less than 1.
`seed`	A random seed, default is 186.

Examples

```
# Load German credit data
data(germancredit)

dt_list = split_df(germancredit, y="creditability")
train = dt_list$train
test = dt_list$test
```

`var_filter` *Variable Filter*

Description

This function filter variables base on specified conditions, such as information value, missing rate, identical value rate.

Usage

```
var_filter(dt, y, x = NULL, iv_limit = 0.02, missing_limit = 0.95,
  identical_limit = 0.95, var_rm = NULL, var_kp = NULL,
  return_rm_reason = FALSE, positive = "bad|1")
```

Arguments

dt	A data frame with both x (predictor/feature) and y (response/label) variables.
y	Name of y variable.
x	Name of x variables. Default is NULL. If x is NULL, then all variables except y are counted as x variables.
iv_limit	The information value of kept variables should >= iv_limit. The default is 0.02.
missing_limit	The missing rate of kept variables should <= missing_limit. The default is 0.95.
identical_limit	The identical value rate (excluding NAs) of kept variables should <= identical_limit. The default is 0.95.
var_rm	Name of force removed variables, default is NULL.
var_kp	Name of force kept variables, default is NULL.
return_rm_reason	Logical, default is FALSE.
positive	Value of positive class, default is "bad|1".

Value

A data.table with y and selected x variables and a data.table with the reason of removed x variable if return_rm_reason == TRUE.

Examples

```
# Load German credit data
data(germancredit)

# variable filter
dt_sel = var_filter(germancredit, y = "creditability")
```

---

woebin *WOE Binning*

Description

woebin generates optimal binning for numerical, factor and categorical variables using methods including tree-like segmentation or chi-square merge. woebin can also customizing breakpoints if the breaks_list was provided.

Usage

```
woebin(dt, y, x = NULL, breaks_list = NULL, special_values = NULL,
  min_perc_fine_bin = 0.02, min_perc_coarse_bin = 0.05, stop_limit = 0.1,
  max_num_bin = 8, positive = "bad|1", no_cores = NULL, print_step = 0L,
  method = "tree")
```

Arguments

dt	A data frame with both x (predictor/feature) and y (response/label) variables.
y	Name of y variable.
x	Name of x variables. Default is NULL. If x is NULL, then all variables except y are counted as x variables.
breaks_list	List of break points, default is NULL. If it is not NULL, variable binning will based on the provided breaks.
special_values	the values specified in special_values will be in separate bins. Default is NULL.
min_perc_fine_bin	The minimum percentage of initial binning class number over total. Accepted range: 0.01-0.2; default is 0.02, which means initial binning into 50 fine bins for continuous variables.
min_perc_coarse_bin	The minimum percentage of final binning class number over total. Accepted range: 0.01-0.2; default is 0.05.
stop_limit	Stop binning segmentation when information value gain ratio less than the stop_limit, or stop binning merge when the minimum of chi-square less than 'qchisq(1-stoplimit, 1)'. Accepted range: 0-0.5; default is 0.1.
max_num_bin	Integer. The maximum number of binning.
positive	Value of positive class, default "bad|1".
no_cores	Number of CPU cores for parallel computation. Defaults NULL. If no_cores is NULL, the no_cores will set as 1 if length of x variables less than 10, and will set as the number of all CPU cores if the length of x variables greater than or equal to 10.
print_step	A non-negative integer. Default is 1. If print_step>0, print variable names by each print_step-th iteration. If print_step=0 or no_cores>1, no message is print.
method	Optimal binning method, it should be "tree" or "chimerge". Default is "tree".

Value

Optimal or customized binning information.

See Also

woebin_ply, woebin_plot, woebin_adj

Examples

```
# load germancredit data
data(germancredit)

# Example I
# binning of two variables in germancredit dataset
bins_2var = woebin(germancredit, y = "creditability", x = c("credit.amount", "purpose"))

## Not run:
# Example II
# binning of the germancredit dataset
bins_germ = woebin(germancredit, y = "creditability")
# converting bins_germ into a dataframe
# bins_germ_df = data.table::rbindlist(bins_germ)

# Example III
# customizing the breakpoints of binning
library(data.table)
dat = rbind(
  germancredit,
  data.table(creditability=sample(c("good","bad"),10,replace=TRUE)),
  fill=TRUE)

breaks_list = list(
  age.in.years = c(26, 35, 37, "Inf%,%missing"),
  housing = c("own", "for free%,%rent")
)

special_values = list(
  credit.amount = c(2600, 9960, "6850%,%missing"),
  purpose = c("education", "others%,%missing")
)

bins_cus_brk = woebin(dat, y="creditability",
  x=c("age.in.years","credit.amount","housing","purpose"),
  breaks_list=breaks_list, special_values=special_values)


## End(Not run)
```

`woebin_adj` *WOE Binning Adjustment*

Description

`woebin_adj` interactively adjust the binning breaks.

Usage

```
woebin_adj(dt, y, bins, adj_all_var = TRUE, special_values = NULL,
  method = "tree")
```

Arguments

dt	A data frame.
y	Name of y variable.
bins	A list or data frame. Binning information generated from woebin.
adj_all_var	Logical, default is TRUE. If it is TRUE, all variables need to adjust binning breaks, otherwise, only include the variables that have more then one inflection point.
special_values	the values specified in special_values will in separate bins. Default is NULL.
method	optimal binning method, it should be "tree" or "chimerge". Default is "tree".

See Also

woebin, woebin_ply, woebin_plot

Examples

```
## Not run:
# Load German credit data
data(germancredit)

# Example I
dt = germancredit[, c("creditability", "age.in.years", "credit.amount")]
bins = woebin(dt, y="creditability")
breaks_adj = woebin_adj(dt, y="creditability", bins)
bins_final = woebin(dt, y="creditability",
                    breaks_list=breaks_adj)

# Example II
binsII = woebin(germancredit, y="creditability")
breaks_adjII = woebin_adj(germancredit, "creditability", binsII)
bins_finalII = woebin(germancredit, y="creditability",
                    breaks_list=breaks_adjII)

## End(Not run)
```

---

woebin_plot *WOE Binning Visualization*

Description

woebin_plot create plots of count distribution and bad probability for each bin. The binning informations are generates by woebin.

Usage

```
woebin_plot(bins, x = NULL, title = NULL, show_iv = TRUE)
```

Arguments

bins	A list or data frame. Binning information generated by woebin.
x	Name of x variables. Default is NULL. If x is NULL, then all variables except y are counted as x variables.
title	String added to the plot title. Default is NULL.
show_iv	Logical. Default is TRUE, which means show information value in the plot title.

Value

List of binning plot

See Also

woebin, woebin_ply, woebin_adj

Examples

```
# Load German credit data
data(germancredit)

# Example I
dt1 = germancredit[, c("creditability", "credit.amount")]

bins1 = woebin(dt1, y="creditability")
p1 = woebin_plot(bins1)

## Not run:
# Example II
bins = woebin(germancredit, y="creditability")
plotlist = woebin_plot(bins)

# # save binning plot
# for (i in 1:length(plotlist)) {
#   ggplot2::ggsave(
#      paste0(names(plotlist[i]), ".png"), plotlist[[i]],
#      width = 15, height = 9, units="cm" )
#   }

## End(Not run)
```

woebin_ply *WOE Transformation*

Description

woebin_ply converts original input data into woe values based on the binning information generated from woebin.

Usage

```
woebin_ply(dt, bins, no_cores = NULL, print_step = 0L)
```

Arguments

dt	A data frame.
bins	Binning information generated from woebin.
no_cores	Number of CPU cores for parallel computation. Defaults NULL. If no_cores is NULL, the no_cores will set as 1 if length of x variables less than 10, and will set as the number of all CPU cores if the length of x variables greater than or equal to 10.
print_step	A non-negative integer. Default is 1. If print_step>0, print variable names by each print_step-th iteration. If print_step=0 or no_cores>1, no message is print.

Value

Binning information

See Also

woebin, woebin_plot, woebin_adj

Examples

```
# load germancredit data
data(germancredit)

# Example I
dt = germancredit[, c("creditability", "credit.amount", "purpose")]

# binning for dt
bins = woebin(dt, y = "creditability")

# converting original value to woe
dt_woe = woebin_ply(dt, bins=bins)

## Not run:
# Example II
# binning for germancredit dataset
```

```
bins_germancredit = woebin(germancredit, y="creditability")

# converting the values in germancredit to woe
# bins is a list which generated from woebin()
germancredit_woe = woebin_ply(germancredit, bins_germancredit)

# bins is a dataframe
bins_df = data.table::rbindlist(bins_germancredit)
germancredit_woe = woebin_ply(germancredit, bins_df)

## End(Not run)
```

Index