

# Package ‘scorecard’

March 26, 2019

**Version** 0.2.4

**Title** Credit Risk Scorecard

**Description** The ‘scorecard’ package makes the development of credit risk scorecard easier and efficient by providing functions for some common tasks, such as data partition, variable selection, woe binning, scorecard scaling, performance evaluation and report generation. These functions can also used in the development of machine learning models.

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.

**Depends** R (>= 3.1.0)

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

**Suggests** knitr, rmarkdown, pkgdown, testthat

**License** MIT + file LICENSE

**URL** <https://github.com/ShichenXie/scorecard>

**BugReports** <https://github.com/ShichenXie/scorecard/issues>

**LazyData** true

**VignetteBuilder** knitr

**RoxygenNote** 6.1.1

**Encoding** UTF-8

**NeedsCompilation** no

**Author** Shichen Xie [aut, cre]

**Maintainer** Shichen Xie <xie@shichen.name>

**Repository** CRAN

**Date/Publication** 2019-03-26 14:00:03 UTC

**R topics documented:**

gains_table	2
germancredit	4
iv	5
one_hot	6
perf_eva	7
perf_psi	10
report	12
scorecard	14
scorecard2	16
scorecard_ply	18
split_df	19
var_filter	20
vif	22
woebin	23
woebin_adj	25
woebin_plot	27
woebin_ply	28
<b>Index</b>	<b>30</b>

---

gains_table	<i>Gains Table</i>
-------------	--------------------

---

**Description**

gains\_table creates a data frame including distribution of total, good, bad, bad rate and approval rate by score bins. It provides both equal width and equal frequency intervals on score binning.

**Usage**

```
gains_table(score, label, bin_num = 10, bin_type = "freq",
            positive = "bad|1", ...)
```

**Arguments**

score	A list of credit score for actual and expected data samples. For example, score = list(actual = scoreA, expect = scoreE).
label	A list of label value for actual and expected data samples. For example, label = list(actual = labelA, expect = labelE).
bin_num	Integer, the number of score bins. Default is 10. If it is 'max', then individual scores are used as bins.
bin_type	The score is binning by equal frequency or equal width. Accepted values are 'freq' and 'width'. Default is 'freq'.
positive	Value of positive class, default is "bad 1".
...	Additional parameters.

**Value**

A data frame

**See Also**

[perf\\_eva](#) [perf\\_psi](#)

**Examples**

```
# data preparing -----
# load germancredit data
data("germancredit")
# filter variable via missing rate, iv, identical value rate
dt_f = var_filter(germancredit, "creditability")
# breaking dt into train and test
dt_list = split_df(dt_f, "creditability")
label_list = lapply(dt_list, function(x) x$creditability)

# woe binning -----
bins = woebin(dt_list$train, "creditability")
# converting train and test into woe values
dt_woe_list = lapply(dt_list, function(x) woebin_ply(x, bins))

# glm -----
m1 = glm(creditability ~ ., family = binomial(), data = dt_woe_list$train)
# vif(m1, merge_coef = TRUE)
# Select a formula-based model by AIC
m_step = step(m1, direction="both", trace=FALSE)
m2 = eval(m_step$call)
# vif(m2, merge_coef = TRUE)

# predicted probability
pred_list = lapply(dt_woe_list, function(x) predict(m2, type = 'response', x))

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

# credit score, only_total_score = TRUE
score_list = lapply(dt_list, function(x) scorecard_ply(x, card))
# credit score, only_total_score = FALSE
score_list2 = lapply(dt_list, function(x) scorecard_ply(x, card, only_total_score=FALSE))

##### perf_eva examples #####
# Example I, one dataset
## predicted p1
perf_eva(pred = pred_list$train, label=dt_list$train$creditability, title = 'train')
## predicted score
# perf_eva(pred = score_list$train, label=dt_list$train$creditability, title = 'train')

# Example II, multiple datasets
```

```
## predicted p1
perf_eva(pred = pred_list, label = label_list)
## predicted score
# perf_eva(score_list, label_list)

##### perf_psi examples #####
# Example I # only total psi
psi1 = perf_psi(score = score_list, label = label_list)
psi1$psi # psi data frame
psi1$pic # pic of score distribution

# Example II # both total and variable psi
psi2 = perf_psi(score = score_list, label = label_list)
# psi2$psi # psi data frame
# psi2$pic # pic of score distribution

##### gains_table examples #####
# Example I, input score and label can be a list or a vector
g1 = gains_table(score = score_list$train, label = label_list$train)
g2 = gains_table(score = score_list, label = label_list)

# Example II, specify the bins number and type
g3 = gains_table(score = score_list, label = label_list, bin_num = 20)
g4 = gains_table(score = score_list, label = label_list, bin_type = 'width')
```

---

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

## Examples

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

# structure of germancredit
str(germancredit)

# summary of germancredit
# lapply(germancredit, summary)
```

---

iv *Information Value*

---

## Description

This function calculates information value (IV) for multiple x variables. It treats each unique value in x variables as a group. If there is a zero number of y class, it will be replaced by 0.99 to make sure woe/iv is calculable.

## 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 columns 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:

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

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

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

The relationship between information value and predictive power is as follows:

Information Value      Predictive Power

< 0.02	useless for prediction
0.02 to 0.1	Weak predictor
0.1 to 0.3	Medium predictor
> 0.3	Strong predictor

**Value**

A data frame with columns for variable and info\_value

**Examples**

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

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

str(info_value)
```

---

one\_hot

*One Hot Encoding*

---

**Description**

One-hot encoding on categorical variables and replace missing values. It is not needed when creating a standard scorecard model, but required in models that without doing woe transformation.

**Usage**

```
one_hot(dt, var_skip = NULL, var_encode = NULL, nacol_rm = FALSE,
        replace_na = NULL)
```

**Arguments**

dt	A data frame.
var_skip	Name of categorical variables that will skip for one-hot encoding. Default is NULL.
var_encode	Name of categorical variables to be one-hot encoded, default is NULL. If it is NULL, then all categorical variables except in var_skip are counted.
nacol_rm	Logical. One-hot encoding on categorical variable contains missing values, whether to remove the column generated to indicate the presence of NAs. Default is FALSE.
replace_na	Replace missing values with a specified value such as -1, or the mean/median value for numeric variable and mode value for categorical variable. Default is NULL, which means no missing values will be replaced.

**Value**

A data frame

**Examples**

```
# load germancredit data
data(germancredit)

library(data.table)
dat = rbind(
  germancredit[, c(sample(20,3),21)],
  data.table(creditability=sample(c("good","bad"),10,replace=TRUE)),
  fill=TRUE)

# one hot encoding
## keep na columns from categorical variable
dat_onehot1 = one_hot(dat, var_skip = 'creditability', nacol_rm = FALSE) # default
str(dat_onehot1)
## remove na columns from categorical variable
dat_onehot2 = one_hot(dat, var_skip = 'creditability', nacol_rm = TRUE)
str(dat_onehot2)

## one hot and replace NAs
dat_onehot3 = one_hot(dat, var_skip = 'creditability', replace_na = -1)
str(dat_onehot3)

# replace missing values only
## replace with -1
dat_repna1 = one_hot(dat, var_skip = names(dat), replace_na = -1)
## replace with median for numeric, and mode for categorical
dat_repna2 = one_hot(dat, var_skip = names(dat), replace_na = 'median')
## replace with to mean for numeric, and mode for categorical
dat_repna3 = one_hot(dat, var_skip = names(dat), replace_na = 'mean')
```

---

perf\_eva

*Binomial Metrics*

---

**Description**

perf\_eva calculates metrics to evaluate the performance of binomial classification model. It can also creates confusion matrix and model performance graphics.

**Usage**

```
perf_eva(pred, label, title = NULL, binomial_metric = c("mse", "rmse",
  "logloss", "r2", "ks", "auc", "gini"), confusion_matrix = TRUE,
```

```
threshold = NULL, show_plot = c("ks", "roc"), positive = "bad|1",
...)
```

### Arguments

pred	A list or vector of predicted probability or score.
label	A list or vector of label values.
title	The title of plot. Default is NULL.
binomial_metric	Default is c('mse', 'rmse', 'logloss', 'r2', 'ks', 'auc', 'gini'). If it is NULL, then no metric will be calculated.
confusion_matrix	Logical, whether to create a confusion matrix. Default is TRUE.
threshold	Confusion matrix threshold. Default is the pred on maximum F1.
show_plot	Default is c('ks', 'roc'). Accepted values including c('ks', 'lift', 'gain', 'roc', 'lz', 'pr', 'fl', 'density').
positive	Value of positive class, default is "bad 1".
...	Additional parameters.

### 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

A list of binomial metric, confusion matrix and graphics

### See Also

[perf\\_psi](#)

### Examples

```
# data preparing -----
# load germancredit data
data("germancredit")
# filter variable via missing rate, iv, identical value rate
dt_f = var_filter(germancredit, "creditability")
# breaking dt into train and test
dt_list = split_df(dt_f, "creditability")
label_list = lapply(dt_list, function(x) x$creditability)
```



```

# woe binning -----
bins = woebin(dt_list$train, "creditability")
# converting train and test into woe values
dt_woe_list = lapply(dt_list, function(x) woebin_ply(x, bins))

# glm -----
m1 = glm(creditability ~ ., family = binomial(), data = dt_woe_list$train)
# vif(m1, merge_coef = TRUE)
# Select a formula-based model by AIC
m_step = step(m1, direction="both", trace=FALSE)
m2 = eval(m_step$call)
# vif(m2, merge_coef = TRUE)

# predicted probability
pred_list = lapply(dt_woe_list, function(x) predict(m2, type = 'response', x))

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

# credit score, only_total_score = TRUE
score_list = lapply(dt_list, function(x) scorecard_ply(x, card))
# credit score, only_total_score = FALSE
score_list2 = lapply(dt_list, function(x) scorecard_ply(x, card, only_total_score=FALSE))

##### perf_eva examples #####
# Example I, one dataset
## predicted p1
perf_eva(pred = pred_list$train, label=dt_list$train$creditability, title = 'train')
## predicted score
# perf_eva(pred = score_list$train, label=dt_list$train$creditability, title = 'train')

# Example II, multiple datasets
## predicted p1
perf_eva(pred = pred_list, label = label_list)
## predicted score
# perf_eva(score_list, label_list)

##### perf_psi examples #####
# Example I # only total psi
psi1 = perf_psi(score = score_list, label = label_list)
psi1$psi # psi data frame
psi1$pic # pic of score distribution

# Example II # both total and variable psi
psi2 = perf_psi(score = score_list, label = label_list)
# psi2$psi # psi data frame
# psi2$pic # pic of score distribution

##### gains_table examples #####

```

```
# Example I, input score and label can be a list or a vector
g1 = gains_table(score = score_list$train, label = label_list$train)
g2 = gains_table(score = score_list, label = label_list)

# Example II, specify the bins number and type
g3 = gains_table(score = score_list, label = label_list, bin_num = 20)
g4 = gains_table(score = score_list, label = label_list, bin_type = 'width')
```

perf\_psi

*PSI***Description**

perf\_psi calculates population stability index (PSI) for both total credit score and variables. It can also create graphics to display score distribution and bad rate trends.

**Usage**

```
perf_psi(score, label = NULL, title = NULL, show_plot = TRUE,
         positive = "bad|1", threshold_variable = 20, var_skip = NULL, ...)
```

**Arguments**

score	A list of credit score for actual and expected data samples. For example, score = list(actual = scoreA, expect = scoreE).
label	A list of label value for actual and expected data samples. For example, label = list(actual = labelA, expect = labelE). Default is NULL.
title	Title of plot, default is NULL.
show_plot	Logical. Default is TRUE.
positive	Value of positive class, default is "bad 1".
threshold_variable	Integer. Default is 20. If the number of unique values > threshold_variable, the provided score will be counted as total credit score, otherwise, it is variable score.
var_skip	Name of variables that are not score, such as id column. It should be the same with the var_kp in scorecard_ply function. Default is NULL.
...	Additional parameters.

**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 data frame of psi and graphics of credit score distribution

**See Also**

[perf\\_eva](#) [gains\\_table](#)

**Examples**

```
# data preparing -----
# load germancredit data
data("germancredit")
# filter variable via missing rate, iv, identical value rate
dt_f = var_filter(germancredit, "creditability")
# breaking dt into train and test
dt_list = split_df(dt_f, "creditability")
label_list = lapply(dt_list, function(x) x$creditability)

# woe binning -----
bins = woebin(dt_list$train, "creditability")
# converting train and test into woe values
dt_woe_list = lapply(dt_list, function(x) woebin_ply(x, bins))

# glm -----
m1 = glm(creditability ~ ., family = binomial(), data = dt_woe_list$train)
# vif(m1, merge_coef = TRUE)
# Select a formula-based model by AIC
m_step = step(m1, direction="both", trace=FALSE)
m2 = eval(m_step$call)
# vif(m2, merge_coef = TRUE)

# predicted probability
pred_list = lapply(dt_woe_list, function(x) predict(m2, type = 'response', x))

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

# credit score, only_total_score = TRUE
score_list = lapply(dt_list, function(x) scorecard_ply(x, card))
# credit score, only_total_score = FALSE
score_list2 = lapply(dt_list, function(x) scorecard_ply(x, card, only_total_score=FALSE))

##### perf_eva examples #####
# Example I, one dataset
## predicted p1
perf_eva(pred = pred_list$train, label=dt_list$train$creditability, title = 'train')
## predicted score
# perf_eva(pred = score_list$train, label=dt_list$train$creditability, title = 'train')

# Example II, multiple datasets
```

```

## predicted p1
perf_eva(pred = pred_list, label = label_list)
## predicted score
# perf_eva(score_list, label_list)

##### perf_psi examples #####
# Example I # only total psi
psi1 = perf_psi(score = score_list, label = label_list)
psi1$psi # psi data frame
psi1$pic # pic of score distribution
# modify colors
# perf_psi(score = score_list, label = label_list,
#           line_color='#FC8D59', bar_color=c('#FFFFBF', '#99D594'))

# Example II # both total and variable psi
psi2 = perf_psi(score = score_list, label = label_list)
# psi2$psi # psi data frame
# psi2$pic # pic of score distribution

##### gains_table examples #####
# Example I, input score and label can be a list or a vector
g1 = gains_table(score = score_list$train, label = label_list$train)
g2 = gains_table(score = score_list, label = label_list)

# Example II, specify the bins number and type
g3 = gains_table(score = score_list, label = label_list, bin_num = 20)
g4 = gains_table(score = score_list, label = label_list, bin_type = 'width')

```

---

report

*Scorecard Modeling Report*


---

## Description

report creates a scorecard modeling report and save it as a xlsx file.

## Usage

```
report(dt, y, x, breaks_list, special_values = NULL, seed = 618,
       save_report = "report", positive = "bad|1", ...)
```

## Arguments

**dt** A data frame or a list of data frames that have both x (predictor/feature) and y (response/label) variables. If there are multiple data frames are provided, only the first data frame would be used for training, and the others would be used for testing/validation.

y	Name of y variable.
x	Name of x variables. Default is NULL. If x is NULL, then all columns except y are counted as x variables.
breaks_list	A list of break points. It can be extracted from woebin and woebin_adj via the argument save_breaks_list.
special_values	The values specified in special_values will be in separate bins. Default is NULL.
seed	A random seed to split input data frame. Default is 618. If it is NULL, input dt will not split into two datasets.
save_report	The name of xlsx file where the report is to be saved. Default is 'report'.
positive	Value of positive class, default "bad1".
...	Additional paramters.

### Examples

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

y = 'creditability'
x = c(
  "status.of.existing.checking.account",
  "duration.in.month",
  "credit.history",
  "purpose",
  "credit.amount",
  "savings.account.and.bonds",
  "present.employment.since",
  "installment.rate.in.percentage.of.disposable.income",
  "personal.status.and.sex",
  "property",
  "age.in.years",
  "other.installment.plans",
  "housing"
)

special_values=NULL
breaks_list=list(
  status.of.existing.checking.account=c("... < 0 DM,%0 <= ... < 200 DM",
    "... >= 200 DM / salary assignments for at least 1 year", "no checking account"),
  duration.in.month=c(8, 16, 34, 44),
  credit.history=c(
    "no credits taken/ all credits paid back duly,%all credits at this bank paid back duly",
    "existing credits paid back duly till now", "delay in paying off in the past",
    "critical account/ other credits existing (not at this bank)"),
  purpose=c("retraining%,%car (used)", "radio/television",
    "furniture/equipment%,%domestic appliances%,%business%,%repairs",
    "car (new)%,%others%,%education"),
  credit.amount=c(1400, 1800, 4000, 9200),
  savings.account.and.bonds=c("... < 100 DM", "100 <= ... < 500 DM",
    "500 <= ... < 1000 DM%,%... >= 1000 DM%,%unknown/ no savings account"),
  present.employment.since=c("unemployed%,%... < 1 year", "1 <= ... < 4 years",
```

```

    "4 <= ... < 7 years", "... >= 7 years"),
  installment.rate.in.percentage.of.disposable.income=c(2, 3),
  personal.status.and.sex=c("female : divorced/separated/married", "male : single",
    "male : married/widowed"),
  property=c("real estate", "building society savings agreement/ life insurance",
    "car or other, not in attribute Savings account/bonds", "unknown / no property"),
  age.in.years=c(26, 28, 35, 37),
  other.installment.plans=c("bank%,%stores", "none"),
  housing=c("rent", "own", "for free")
)

# Example I
# input dt is a data frame
# split input data frame into two
report(germancredit, y, x, breaks_list, special_values, seed=618, save_report='report1',
  show_plot = c('ks', 'lift', 'gain', 'roc', 'lz', 'pr', 'f1', 'density'))

# donot split input data
report(germancredit, y, x, breaks_list, special_values, seed=NULL, save_report='report2')

# Example II
# input dt is a list
# only one dataset
report(list(dt=germancredit), y, x,
  breaks_list, special_values, seed=NULL, save_report='report3')

# multiple datasets
report(list(dt1=germancredit[sample(1000,500)],
  dt2=germancredit[sample(1000,500)]), y, x,
  breaks_list, special_values, seed=NULL, save_report='report4')

# multiple datasets
report(list(dt1=germancredit[sample(1000,500)],
  dt2=germancredit[sample(1000,500)],
  dt3=germancredit[sample(1000,500)]), y, x,
  breaks_list, special_values, seed=NULL, save_report='report5')

## End(Not run)

```

**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**

A scorecard data frame

**See Also**

[scorecard2](#) [scorecard\\_ply](#)

**Examples**

```
# 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 probability
# 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)
card2 = scorecard2(bins=bins, dt=germancredit, y='creditability',
  x=c("status.of.existing.checking.account", "duration.in.month", "credit.history",
    "purpose", "credit.amount", "savings.account.and.bonds",
    "present.employment.since", "installment.rate.in.percentage.of.disposable.income",
    "personal.status.and.sex", "other.debtors.or.guarantors", "property",
    "age.in.years", "other.installment.plans", "housing"))

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

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

```

---

scorecard2

*Creating a Scorecard*


---

## Description

scorecard2 creates a scorecard based on the results from woebin. It has the same function with scorecard, but without model object input.

## Usage

```
scorecard2(bins, dt, y, x = NULL, points0 = 600, odds0 = 1/19,
  pdo = 50, basepoints_eq0 = FALSE, positive = "bad|1", ...)
```

## Arguments

bins	Binning information generated from woebin function.
dt	A data frame with both x (predictor/feature) and y (response/label) variables.
y	Name of y variable.
x	Name of x variables. If it is NULL, then all variables in bins are used. Default is NULL.
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.
positive	Value of positive class, default "bad 1".
...	Additional parameters.



**Value**

A scorecard data frame

**See Also**

[scorecard](#) [scorecard\\_ply](#)

**Examples**

```
# 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 probability
# 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)
card2 = scorecard2(bins=bins, dt=germancredit, y='creditability',
  x=c("status.of.existing.checking.account", "duration.in.month", "credit.history",
    "purpose", "credit.amount", "savings.account.and.bonds",
    "present.employment.since", "installment.rate.in.percentage.of.disposable.income",
    "personal.status.and.sex", "other.debtors.or.guarantors", "property",
    "age.in.years", "other.installment.plans", "housing"))

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

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

---

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 = 0L,
  replace_blank_na = TRUE, var_kp = NULL)
```

### Arguments

dt	A data frame, which is the original dataset for training model.
card	The scorecard generated from the function 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.
replace_blank_na	Logical. Replace blank values with NA. Default is TRUE. This argument should be the same with woebin's.
var_kp	Name of force kept variables, such as id column. Default is NULL.

### Value

A data frame in score values

### See Also

[scorecard](#) [scorecard2](#)

### Examples

```
# 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 probability
# 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)
card2 = scorecard2(bins=bins, dt=germancredit, y='creditability',
  x=c("status.of.existing.checking.account", "duration.in.month", "credit.history",
    "purpose", "credit.amount", "savings.account.and.bonds",
    "present.employment.since", "installment.rate.in.percentage.of.disposable.income",
    "personal.status.and.sex", "other.debtors.or.guarantors", "property",
    "age.in.years", "other.installment.plans", "housing"))

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

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

```

---

split\_df

*Split a dataset*


---

### Description

Split a dataset into train and test

### Usage

```
split_df(dt, y = NULL, ratio = 0.7, seed = 618)
```

**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 618.

**Value**

A list of data frames

**Examples**

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

# Example I
dt_list = split_df(germancredit, y="creditability")
train = dt_list[[1]]
test = dt_list[[2]]

# dimensions of train and test datasets
lapply(dt_list, dim)

# Example II
dt_list2 = split_df(germancredit, y="creditability", ratio = c(0.5, 0.2))
lapply(dt_list2, dim)
```

---

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 columns except y are counted as x variables.
iv_limit	The information value of kept variables should $\geq$ iv_limit. The default is 0.02.
missing_limit	The missing rate of kept variables should $\leq$ missing_limit. The default is 0.95.
identical_limit	The identical value rate (excluding NAs) of kept variables should $\leq$ 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 "bad1".

**Value**

A data frame with columns for y and selected x variables, and a data frame with columns for remove reason if return\_rm\_reason == TRUE.

**Examples**

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

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

# return the reason of variable removed
dt_sel2 = var_filter(germancredit, y = "creditability", return_rm_reason = TRUE)
lapply(dt_sel2, dim)

str(dt_sel2$dt)
str(dt_sel2$rm)

# keep columns manually, such as rowid
germancredit$rowid = row.names(germancredit)
dt_sel3 = var_filter(germancredit, y = "creditability", var_kp = 'rowid')

# remove columns manually
dt_sel4 = var_filter(germancredit, y = "creditability", var_rm = 'rowid')
```

---

**vif***Variance Inflation Factors*

---

**Description**

vif calculates variance-inflation and generalized variance-inflation factors for linear, generalized linear.

**Usage**

```
vif(model, merge_coef = FALSE)
```

**Arguments**

model	A model object.
merge_coef	Logical, whether to merge with coefficients of model summary matrix. Default is FALSE.

**Value**

A data frame with columns for variable and gvif, or additional columns for df and  $gvif^{1/(2*df)}$  if provided model uses factor variable.

**See Also**

<https://cran.r-project.org/package=car>

**Examples**

```
data(germancredit)

# Example I
fit1 = glm(creditability~ age.in.years + credit.amount +
  present.residence.since, family = binomial(), data = germancredit)
vif(fit1)
vif(fit1, merge_coef=TRUE)

# Example II
fit2 = glm(creditability~ status.of.existing.checking.account +
  credit.history + credit.amount, family = binomial(), data = germancredit)
vif(fit2)
vif(fit2, merge_coef=TRUE)
```

---

woebin	<i>WOE Binning</i>
--------	--------------------

---

**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. The default woe is defined as  $\ln(\text{Bad}_i/\text{Good}_i)$ . If you prefer  $\ln(\text{Good}_i/\text{Bad}_i)$ , please set the argument positive as negative value, such as '0' or 'good'. If there is a zero frequency class when calculating woe, the zero will replaced by 0.99 to make the woe calculable.

**Usage**

```
woebin(dt, y, x = NULL, var_skip = NULL, breaks_list = NULL,
       special_values = NULL, stop_limit = 0.1, count_distr_limit = 0.05,
       bin_num_limit = 8, positive = "bad|1", no_cores = NULL,
       print_step = 0L, method = "tree", save_breaks_list = NULL,
       ignore_const_cols = TRUE, ignore_datetime_cols = TRUE,
       check_cate_num = TRUE, replace_blank_na = 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 columns except y and var_skip are counted as x variables.
var_skip	Name of variables that will skip for binning. Default is NULL.
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.
stop_limit	Stop binning segmentation when information value gain ratio less than the 'stop_limit' if using tree method; or stop binning merge when the chi-square of each neighbor bins are larger than the threshold under significance level of 'stop_limit' and freedom degree of 1 if using chimerge method. Accepted range: 0-0.5; default is 0.1.
count_distr_limit	The minimum count distribution percentage. Accepted range: 0.01-0.2; default is 0.05.
bin_num_limit	Integer. The maximum number of binning. Default is 8.
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	Four methods are provided, "tree" and "chimerge" for optimal binning that support both numerical and categorical variables, and 'width' and 'freq' for equal binning that support numerical variables only. Default is "tree".
save_breaks_list	A string. The file name to save breaks_list. Default is None.
ignore_const_cols	Logical. Ignore constant columns. Default is TRUE.
ignore_datetime_cols	Logical. Ignore datetime columns. Default is TRUE.
check_cate_num	Logical. Check whether the number of unique values in categorical columns larger than 50. It might make the binning process slow if there are too many unique categories. Default is TRUE.
replace_blank_na	Logical. Replace blank values with NA. Default is TRUE.
...	Additional parameters.

**Value**

A list of data frames include binning information for each x variables.

**See Also**

[woebin\\_ply](#), [woebin\\_plot](#), [woebin\\_adj](#)

**Examples**

```
# load germancredit data
data(germancredit)

# Example I
# binning of two variables in germancredit dataset
# using tree method
bins2_tree = woebin(germancredit, y="creditability",
  x=c("credit.amount", "housing"), method="tree")
bins2_tree

## Not run:
# using chimerge method
bins2_chi = woebin(germancredit, y="creditability",
  x=c("credit.amount", "housing"), method="chimerge")

# binning in equal freq/width # only supports numerical variables
numeric_cols = c("duration.in.month", "credit.amount",
  "installment.rate.in.percentage.of.disposable.income", "present.residence.since",
  "age.in.years", "number.of.existing.credits.at.this.bank",
  "number.of.people.being.liable.to.provide.maintenance.for")
bins_freq = woebin(germancredit, y="creditability", x=numeric_cols, method="freq")
```



```

bins_width = woebin(germancredit, y="creditability", x=numeric_cols, method="width")

# y can be NULL if no label column in dataset
bins_freq_noy = woebin(germancredit, y=NULL, x=numeric_cols)

# Example II
# binning of the germancredit dataset
bins_germ = woebin(germancredit, y = "creditability")
# converting bins_germ into a data frame
# 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)

# Example IV
# save breaks_list as a R file
bins2 = woebin(germancredit, y="creditability",
  x=c("credit.amount", "housing"), save_breaks_list='breaks_list')

## 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", save_breaks_list = NULL, count_distr_limit = 0.05)
```

**Arguments**

**dt** A data frame.

**y** Name of y variable.

**bins** A list of data frames. Binning information generated from woebin.

**adj\_all\_var** Logical, whether to show variables have monotonic woe trends. Default is TRUE

**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".

**save\_breaks\_list** A string. The file name to save breaks\_list. Default is None.

**count\_distr\_limit** The minimum count distribution percentage. Accepted range: 0.01-0.2; default is 0.05. This argument should be the same with woebin's.

**Value**

A list of modified break points of each x variables.

**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_finalIII = woebin(germancredit, y="creditability",
                       breaks_list=breaks_adjII)

## End(Not run)
```

---

woebin_plot	<i>WOE Binning Visualization</i>
-------------	----------------------------------

---

### Description

woebin\_plot create plots of count distribution and bad probability for each bin. The binning information are generated by woebin.

### Usage

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

### Arguments

bins	A list of data frames. Binning information generated by woebin.
x	Name of x variables. Default is NULL. If x is NULL, then all columns 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.
...	Additional parameters

### Value

A list of binning graphics.

### See Also

[woebin](#), [woebin\\_ply](#), [woebin\\_adj](#)

### Examples

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

# Example I
bins1 = woebin(germancredit, y="creditability", x="credit.amount")

p1 = woebin_plot(bins1)
print(p1)
# modify colors
p1_c = woebin_plot(bins1, line_color='#FC8D59', bar_color=c('#FFFFFF', '#99D594'))
print(p1_c)

# Example II
bins = woebin(germancredit, y="creditability")
plotlist = woebin_plot(bins)
print(plotlist$credit.amount)
```

```

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

```

---

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,
  replace_blank_na = TRUE, ...)

```

### 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.
replace_blank_na	Logical. Replace blank values with NA. Default is TRUE. This argument should be the same with woebin's.
...	Additional parameters.

### Value

A data frame with columns for variables converted into woe values.

### 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)
str(dt_woe)

# 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 data frame
bins_df = data.table::rbindlist(bins_germancredit)
germancredit_woe = woebin_ply(germancredit, bins_df)

# return value is bin but not woe
germancredit_bin = woebin_ply(germancredit, bins_germancredit, value = 'bin')
```

# Index

## \*Topic **data**

germancredit, 4

gains\_table, 2, 11

germancredit, 4

iv, 5

one\_hot, 6

perf\_eva, 3, 7, 11

perf\_psi, 3, 8, 10

report, 12

scorecard, 14, 17, 18

scorecard2, 15, 16, 18

scorecard\_ply, 15, 17, 18

split\_df, 19

var\_filter, 20

vif, 22

woebin, 23, 26–28

woebin\_adj, 24, 25, 27, 28

woebin\_plot, 24, 26, 27, 28

woebin\_ply, 24, 26, 27, 28