Package ‘scorecard’

October 17, 2019

Version 0.2.7

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:

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 2019-10-17 18:40:02 UTC
R topics documented:

- gains_table
- germancredit
- iv
- one_hot
- perf_cv
- perf_eva
- perf_psi
- report
- scorecard
- scorecard2
- scorecard_ply
- split_df
- var_filter
- vif
- woebin
- woebin_adj
- woebin_plot
- woebin_ply

Index

- gains_table

Gains Table

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, method = "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. Defaults to 10. If it is 'max', then individual scores are used as bins.
- method: The score is binning by equal frequency or equal width. Accepted values are 'freq' and 'width'. Defaults to 'freq'.
- positive: Value of positive class, Defaults to "bad|1".
- ...: Additional parameters.
gains_table

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

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

# predicted proability
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 datsets
## 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)
p1 = 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, method = '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

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

# structure of germancredit
str(germancredit)

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

### 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

```r
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. Defaults to NULL. If x is NULL, then all columns except y are counted as x variables.
- **positive**: Value of positive class, Defaults to "bad|1".
- **order**: Logical, Defaults to 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:

\[
IV = \sum (\text{DistributionBad}_i - \text{DistributionGood}_i) * \ln\left(\frac{\text{DistributionBad}_i}{\text{DistributionGood}_i}\right).
\]

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

\[
\text{Weight of Evidence} = \ln\left(\frac{\text{DistributionBad}_i}{\text{DistributionGood}_i}\right).
\]

The relationship between information value and predictive power is as follows:
Information Value Predictive Power

<table>
<thead>
<tr>
<th>Information Value</th>
<th>Predictive Power</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; 0.02</td>
<td>useless for prediction</td>
</tr>
<tr>
<td>0.02 to 0.1</td>
<td>Weak predictor</td>
</tr>
<tr>
<td>0.1 to 0.3</td>
<td>Medium predictor</td>
</tr>
<tr>
<td>&gt; 0.3</td>
<td>Strong predictor</td>
</tr>
</tbody>
</table>

Value

A data frame with columns for variable and info_value

Examples

```r
# 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**

```r
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. Defaults to NULL.
- `var_encode` Name of categorical variables to be one-hot encoded. Defaults to 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. Defaults to 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. Defaults to NULL, which means no missing values will be replaced.
Value

A data frame

Examples

# load germancredit data
data(germancredit)

library(data.table)

dat = rbind(
   setDT(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_cv**

Cross Validation

Description

perf_cv provides cross validation on logistic regression and other binomial classification models.

Usage

perf_cv(dt, y, x = NULL, no_folds = 5, seeds = NULL,
  binomial_metric = "ks", positive = "bad|1", breaks_list = NULL,
  ...)


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. Defaults to NULL. If x is NULL, then all columns except y are counted as x variables.
- **no_folds**: Number of folds for K-fold cross-validation. Defaults to 5.
- **seeds**: The seeds to create multiple random splits of the input dataset into training and validation data by using `split_df` function. Defaults to NULL.
- **binomial_metric**: Defaults to ks.
- **positive**: Value of positive class, defaults to "bad1".
- **breaks_list**: List of break points, defaults to NULL. If it is NULL, then using original values of the input data to fitting model, otherwise converting into woe values based on training data.
- **...**: Additional parameters.

Value

A list of data frames of binomial metrics for each datasets.

Examples

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

dt = var_filter(germancredit, y = 'creditability')
bins = woebin(dt, y = 'creditability')
dt_woe = woebin_ply(dt, bins)

perf1 = perf_cv(dt_woe, y = 'creditability', no_folds = 5)

perf2 = perf_cv(dt_woe, y = 'creditability', no_folds = 5,
                seeds = sample(1000, 10))

perf3 = perf_cv(dt_woe, y = 'creditability', no_folds = 5,
                binomial_metric = c('ks', 'auc'))

## End(Not run)
```
Description

`perf_eva` calculates metrics to evaluate the performance of binomial classification model. It can also creates confusion matrix and model performance graphics.

Usage

```r
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. Defaults to NULL.
- `binomial_metric`: Defaults to `c("mse", "rmse", "logloss", "r2", "ks", "auc", "gini")`. If it is NULL, then no metric will calculated.
- `confusion_matrix`: Logical, whether to create a confusion matrix. Defaults to TRUE.
- `threshold`: Confusion matrix threshold. Defaults to the pred on maximum F1.
- `show_plot`: Defaults to `c("ks", "roc")`. Accepted values including `c("ks", "lift", "gain", "roc", "lz", "pr", "11", "density")`.
- `positive`: Value of positive class, Defaults to "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

```r
# 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)

# woebin ------
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 datsets
## predicted p1
perf_eva(pred = pred_list, label = label_list)
## predicted score
# perf_eva(score_list, label_list)
```
perf_psi

##### 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, method = 'width')

---

**perf_psi**

<table>
<thead>
<tr>
<th><strong>PSI</strong></th>
</tr>
</thead>
</table>

**Description**

`perf_psi` calculates population stability index (PSI) for both total credit score and variables. It can also creates graphics to display score distribution and bad rate trends.

**Usage**

```r
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)`. Defaults to NULL.
- `title`: Title of plot, Defaults to NULL.
- `show_plot`: Logical. Defaults to TRUE.
- `positive`: Value of positive class, Defaults to "bad\|1".
threshold_variable

Integer. Defaults to 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. Defaults to NULL.

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 proability
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, method = 'width')
Scorecard Modeling Report

Description

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

Usage

```r
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. Defaults to 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. Defaults to NULL.
- **seed**: A random seed to split input data frame. Defaults to 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. Defaults to 'report'.
- **positive**: Value of positive class, default "bad|1".
- **...**: Additional parameters.

Examples

```r
## 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",
    "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",
    "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", "... >= 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')
scorecard

**Creating a Scorecard**

**Description**

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

**Usage**

```r
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, Defaults to 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 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)
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)
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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bins</td>
<td>Binning information generated from woebin function.</td>
</tr>
<tr>
<td>dt</td>
<td>A data frame with both x (predictor/feature) and y (response/label) variables.</td>
</tr>
<tr>
<td>y</td>
<td>Name of y variable.</td>
</tr>
<tr>
<td>x</td>
<td>Name of x variables. If it is NULL, then all variables in bins are used. Defaults to NULL.</td>
</tr>
<tr>
<td>points0</td>
<td>Target points, default 600.</td>
</tr>
<tr>
<td>odds0</td>
<td>Target odds, default 1/19. Odds = p/(1-p).</td>
</tr>
<tr>
<td>pdo</td>
<td>Points to Double the Odds, default 50.</td>
</tr>
<tr>
<td>basepoints_eq0</td>
<td>Logical, Defaults to FALSE. If it is TRUE, the basepoints will equally distribute to each variable.</td>
</tr>
<tr>
<td>positive</td>
<td>Value of positive class, default &quot;bad</td>
</tr>
<tr>
<td>...</td>
<td>Additional parameters.</td>
</tr>
</tbody>
</table>

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 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)
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, Defaults to 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. Defaults to 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. Defaults to TRUE. This argument should be the same with woebin’s.

var_kp

Name of force kept variables, such as id column. Defaults to 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 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)
```
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 = c(0.7, 0.3), seed = 618, no_dfs = 2,
    name_dfs = c("train", "test"))

Arguments

dt A data frame.
y Name of y variable, Defaults to NULL. The input data will split based on the predictor y, if it is provide.
ratio A numeric value, Defaults to 0.7. It indicates the ratio of total rows contained in one split, must less than 1.
seed A random seed, Defaults to 618.
no_dfs Number of returned data frames. Defaults to 2.
name_dfs Name of returned data frames. If its length is not equal with no_dfs, then the names will seted as 'dX'. Defaults to train and test.

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. Defaults to NULL. If x is NULL, then all columns except y are counted as x variables.
iv_limit The information value of kept variables should >= iv_limit. The Defaults to 0.02.
missing_limit The missing rate of kept variables should <= missing_limit. The Defaults to 0.95.
identical_limit The identical value rate (excluding NAs) of kept variables should <= identical_limit. The Defaults to 0.95.
var_rm Name of force removed variables, Defaults to NULL.
vif

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. Defaults to 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

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. The default woe is defined as ln(Bad_i/Good_i). If you prefer ln(Good_i/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.
woebin

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_inf = 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. Defaults to 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. Defaults to NULL.
breaks_list List of break points. Defaults to 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. Defaults to 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; Defaults to 0.1. If it is 'N', each x value is a bin.
count_distr_limit The minimum count distribution percentage. Accepted range: 0.01-0.2; Defaults to 0.05.
bin_num_limit Integer. The maximum number of binning. Defaults to 8.
positive Value of positive class, defaults to "bad|1".
no_cores Number of CPU cores for parallel computation. Defaults to 90 percent of total cpu cores.
print_step A non-negative integer. Defaults to 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. Defaults to "tree".
save_breaks_list A string. The file name to save breaks_list. Defaults to None.
ignore_const_cols Logical. Ignore constant columns. Defaults to TRUE.
ignore_datetime_cols Logical. Ignore datetime columns. Defaults to 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. Defaults to TRUE.
replace_blank_inf
    Logical. Replace blank values with NA and infinite with -1. Defaults to 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
    # setting of stop_limit
    # stop_limit = 0.1 (by default)
    bins_x1 = woebin(germancredit, y = 'creditability', x = 'foreign.worker', stop_limit = 0.1)
    # stop_limit = 'N', each x value is a bin
    bins_x1_N = woebin(germancredit, y = 'creditability', x = 'foreign.worker', stop_limit = 'N')

    # Example III
    # 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)
woebin_adj

# Example IV
# customizing the breakpoints of binning
library(data.table)
dat = rbind(
  setDT(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 V
# 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.
woebin_adj

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

adj_all_var Logical, whether to show variables have monotonic woe trends. Defaults to TRUE.

special_values The values specified in special_values will in separate bins. Defaults to NULL.

method Optimal binning method, it should be "tree" or "chimerge". Defaults to "tree".

save_breaks_list A string. The file name to save breaks_list. Defaults to None.

count_distr_limit The minimum count distribution percentage. Accepted range: 0.01-0.2; Defaults to 0.05. This argument should be the same with woebin’s.

... Additional parameters.

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_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

```r
woebin_plot(bins, x = NULL, title = NULL, show_iv = TRUE,
            line_value = "badprob", ...)
```

Arguments

- `bins`: A list of data frames. Binning information generated by `woebin`.
- `x`: Name of x variables. Defaults to NULL. If x is NULL, then all columns except y are counted as x variables.
- `title`: String added to the plot title. Defaults to NULL.
- `show_iv`: Logical. Defaults to TRUE, which means show information value in the plot title.
- `line_value`: The value displayed as line. Accepted values are 'badprob' and 'woe'. Defaults to bad probability.
- `...`: Additional parameters

Value

A list of binning graphics.

See Also

`woebin, woebin_ply, woebin_adj`

Examples

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

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

pl = woebin_plot(bins1)
print(pl)

# modify line value
pl_w = woebin_plot(bins1, line_value = 'woe')
print(pl_w)
```
# 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**

```r
woebin_ply(dt, bins, no_cores = NULL, print_step = 0L,
            replace_blank_inf = TRUE, ...)
```

**Arguments**

- `dt` A data frame.
- `bins` Binning information generated from woebin.
- `no_cores` Number of CPU cores for parallel computation. Defaults to 90 percent of total cpu cores.
- `print_step` A non-negative integer. Defaults to 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_inf` Logical. Replace blank values with NA and infinite with -1. Defaults to TRUE. This argument should be the same with woebin's.
- `...` Additional parameters.
woebin_ply

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, 12
germancredit, 4

iv, 5
one_hot, 6

perf_cv, 7
perf_eva, 3, 9, 12
perf_psi, 3, 10, 11

report, 14

scorecard, 16, 18, 20
scorecard2, 16, 17, 20
scorecard_ply, 16, 18, 19
split_df, 21

var_filter, 22
vif, 23

woebin, 24, 28, 29, 31
woebin_adj, 26, 27, 29, 31
woebin_plot, 26, 28, 29, 31
woebin_ply, 26, 28, 29, 30