<table>
<thead>
<tr>
<th>R topics documented:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>analysis_nas</td>
<td>6</td>
</tr>
<tr>
<td>analysis_outliers</td>
<td>7</td>
</tr>
<tr>
<td>as_percent</td>
<td>8</td>
</tr>
<tr>
<td>auc_value</td>
<td>8</td>
</tr>
<tr>
<td>char_cor_vars</td>
<td>9</td>
</tr>
<tr>
<td>char_to_num</td>
<td>10</td>
</tr>
<tr>
<td>checking_data</td>
<td>10</td>
</tr>
<tr>
<td>check_rules</td>
<td>11</td>
</tr>
<tr>
<td>city_variable</td>
<td>12</td>
</tr>
<tr>
<td>city_variable_process</td>
<td>12</td>
</tr>
<tr>
<td>cohort_analysis</td>
<td>13</td>
</tr>
<tr>
<td>cohort_table_plot</td>
<td>13</td>
</tr>
<tr>
<td>cor_heat_plot</td>
<td>14</td>
</tr>
<tr>
<td>cor_plot</td>
<td>15</td>
</tr>
<tr>
<td>cos_sim</td>
<td>15</td>
</tr>
<tr>
<td>cross_table</td>
<td>16</td>
</tr>
<tr>
<td>customer_segmentation</td>
<td>17</td>
</tr>
<tr>
<td>cut_equal</td>
<td>18</td>
</tr>
<tr>
<td>cv_split</td>
<td>19</td>
</tr>
<tr>
<td>data_cleansing</td>
<td>19</td>
</tr>
<tr>
<td>data_exploration</td>
<td>21</td>
</tr>
<tr>
<td>date_cut</td>
<td>22</td>
</tr>
<tr>
<td>derived_interval</td>
<td>22</td>
</tr>
<tr>
<td>derived_partial_acf</td>
<td>23</td>
</tr>
<tr>
<td>derived_pct</td>
<td>23</td>
</tr>
<tr>
<td>derived_ts_vars</td>
<td>24</td>
</tr>
<tr>
<td>de_one_hot_encoding</td>
<td>25</td>
</tr>
<tr>
<td>de_percent</td>
<td>26</td>
</tr>
<tr>
<td>digits_num</td>
<td>26</td>
</tr>
<tr>
<td>entropy_weight</td>
<td>27</td>
</tr>
<tr>
<td>entry_rate_na</td>
<td>28</td>
</tr>
<tr>
<td>euclid_dist</td>
<td>28</td>
</tr>
<tr>
<td>eval_auc</td>
<td>29</td>
</tr>
<tr>
<td>ewm_data</td>
<td>29</td>
</tr>
<tr>
<td>fast_high_cor_filter</td>
<td>30</td>
</tr>
<tr>
<td>feature_selector</td>
<td>31</td>
</tr>
<tr>
<td>fuzzy_cluster_means</td>
<td>33</td>
</tr>
<tr>
<td>gather_data</td>
<td>33</td>
</tr>
<tr>
<td>gbm_filter</td>
<td>34</td>
</tr>
<tr>
<td>gbm_params</td>
<td>35</td>
</tr>
<tr>
<td>get_auc_ks_lambda</td>
<td>37</td>
</tr>
<tr>
<td>get_bins_table_all</td>
<td>37</td>
</tr>
<tr>
<td>get_breaks_all</td>
<td>39</td>
</tr>
<tr>
<td>get_correlation_group</td>
<td>41</td>
</tr>
<tr>
<td>get_ctree_rules</td>
<td>42</td>
</tr>
<tr>
<td>get_iv_all</td>
<td>43</td>
</tr>
<tr>
<td>get_logistic_coef</td>
<td>44</td>
</tr>
<tr>
<td>get_median</td>
<td>45</td>
</tr>
</tbody>
</table>
topics documented:

- get_names .................................................. 46
- get_nas_random ........................................... 47
- get_plots .................................................... 47
- get_psi_all ............................................... 48
- get_psi_iv_all ............................................ 50
- get_psi_plots ............................................. 52
- get_score_card ........................................... 53
- get_shadow_nas ........................................... 54
- get_sim_sign_lambda ..................................... 55
- get_tree_breaks ......................................... 55
- get_x_list ................................................ 56
- high_cor_selector ....................................... 57
- is_date .................................................... 58
- knn_nas_imp ............................................... 58
- ks_table ................................................... 59
- ks_value ................................................... 60
- lasso_filter .............................................. 61
- lendingclub .............................................. 62
- lift_value ................................................ 64
- local_outlier_factor .................................... 64
- log_trans .................................................. 65
- loop_function ............................................ 65
- love_color ............................................... 66
- low_variance_filter ..................................... 67
- lr_params ................................................ 67
- lr_vif .................................................... 69
- max_min_norm ............................................ 70
- merge_category .......................................... 70
- min_max_norm ............................................ 71
- model_result_plot ....................................... 72
- multi_grid ............................................... 74
- multi_left_join ......................................... 75
- null_blank_na ........................................... 75
- one_hot_encoding ........................................ 76
- outliers_detection ...................................... 77
- partial_dependence_plot ................................ 77
- PCA_reduce ............................................... 79
- plot_bar .................................................. 79
- plot_box .................................................. 80
- plot_density .............................................. 80
- plot_distribution ........................................ 81
- plot_out_perf ............................................ 82
- plot_relative_freq_histogram ......................... 83
- plot_table ............................................... 84
- plot_theme ............................................... 85
- pred_score ............................................... 86
- pred_xgb ................................................ 87
- process_nas ............................................. 88
R topics documented:

process_outliers .................................................. 89
psi_iv_filter .......................................................... 90
p_ij ................................................................. 92
p_to_score .......................................................... 92
quick_as_df ........................................................... 93
ranking_percent_proc ............................................... 93
read_data ............................................................ 94
reduce_high_cor_filter ........................................... 95
remove_duplicated ................................................... 96
require_packages ................................................... 96
re_code .............................................................. 97
re_name ............................................................... 97
rf_params ............................................................ 98
rowAny ............................................................... 99
rules_filter .......................................................... 100
save_data ............................................................ 101
score_transfer ....................................................... 102
select_best_class ................................................... 103
sim_str ............................................................... 105
split_bins ............................................................ 106
split_bins_all ........................................................ 106
start_parallel_computing .......................................... 108
stop_parallel_computing ........................................... 108
str_match ............................................................. 109
swap_analysis ......................................................... 109
term_tfidf ............................................................ 110
time_series_proc ..................................................... 111
time_transfer ........................................................ 112
time_variable ........................................................ 112
time_vars_process ................................................... 113
tnr_value ............................................................... 113
training_model ........................................................ 114
train_lr ............................................................... 116
train_test_split ...................................................... 117
train_xgb ............................................................. 118
UCICreditCard ........................................................ 119
variable_process ..................................................... 120
var_group_proc ....................................................... 121
woe_trans_all ........................................................ 121
xgb_data .............................................................. 123
xgb_filter ............................................................. 123
xgb_params ............................................................ 125
%alike% ............................................................... 126
%islike% ............................................................... 127

Index ................................................................. 128
creditmodel-package  creditmodel: toolkit for credit modeling and data analysis

Description

creditmodel provides a highly efficient R tool suite for Credit Modeling, Analysis and Visualization. Contains infrastructure functionalities such as data exploration and preparation, missing values treatment, outliers treatment, variable derivation, variable selection, dimensionality reduction, grid search for hyper parameters, data mining and visualization, model evaluation, strategy analysis etc. This package is designed to make the development of binary classification models (machine learning based models as well as credit scorecard) simpler and faster.

Details

It has three main goals:

• creditmodel is a free and open source automated modeling R package designed to help model developers improve model development efficiency and enable many people with no background in data science to complete the modeling work in a short time. Let them focus more on the problem itself and allocate more time to decision-making.

• creditmodel covers various tools such as data preprocessing, variable processing/derivation, variable screening/dimensionality reduction, modeling, data analysis, data visualization, model evaluation, strategy analysis, etc. It is a set of customized "core" tool kit for model developers.

• ‘creditmodel’ is suitable for machine learning automated modeling of classification targets, and is more suitable for the risk and marketing data of financial credit, e-commerce, and insurance with relatively high noise and low information content.

To learn more about creditmodel, start with the vignettes: ‘browseVignettes(package = "creditmodel").’

Author(s)

Maintainer: Dongping Fan <fdp@pku.edu.cn>

address_varieble  address_varieble

Description

This function is not intended to be used by end user.

Usage

address_varieble(df, address_cols = NULL, address_pattern = NULL, parallel = TRUE)
Arguments

def        A data frame.
address_cols Variables of address,
address_pattern Regular expressions, used to match address variable names.
parallel    Logical, parallel computing. Default is TRUE.

Description

This function is not intended to be used by end user.

Usage

add_variable_process(add)

Arguments

add         A data.frame contained address variables.

Description

#' analysis_nas is for understanding the reason for missing data and understand distribution of missing data so we can categorise it as:

- missing completely at random(MCAR)
- Missing at random(MAR), or
- missing not at random, also known as IM.

Usage

analysis_nas(dat, class_var = FALSE, nas_rate = NULL, na_vars = NULL, 
mat_nas_shadow = NULL, dt_nas_random = NULL, ...)
**analysis_outliers**

**Arguments**

- **dat** A data.frame with independent variables and target variable.
- **class_var** Logical, na analysis of the nominal variables. Default is TRUE.
- **nas_rate** A list contains nas rate of each variable.
- **na_vars** Names of variables which contain nas.
- **mat_nas_shadow** A shadow matrix of variables which contain nas.
- **dt_nas_random** A data.frame with random nas imputation.
- **...** Other parameters.

**Value**

A data.frame with outliers analysis for each variable.

**Analysis**

Outliers Analysis

---

**Description**

#’ analysis_outliers is the function for outliers analysis.

**Usage**

```r
analysis_outliers(dat, target, x, lof = NULL)
```

**Arguments**

- **dat** A data.frame with independent variables and target variable.
- **target** The name of target variable.
- **x** The name of variable to process.
- **lof** Outliers of each variable detected by outliers_detection.

**Value**

A data.frame with outliers analysis for each variable.
as_percent

Percent Format

Description

as_percent is a small function for making percent format.

Usage

as_percent(x, digits = 2)

Arguments

x A numeric vector or list.
digits Number of digits. Default: 2.

Value

x with percent format.

Examples

as_percent(0.2363, digits = 2)
as_percent(1)

auc_value

auc_value auc_value is for get best lambda required in lasso_filter.
This function required in lasso_filter

Description

auc_value auc_value is for get best lambda required in lasso_filter. This function required in lasso_filter

Usage

auc_value(target, prob)

Arguments

target Vector of target.
prob A list of predict probability or score.

Value

Lanmbda value
**Description**

`char_cor_vars` is function for calculating Cramer’s V matrix between categorical variables. `char_cor` is function for calculating the correlation coefficient between variables by cremers ‘V.

**Usage**

```r
char_cor_vars(dat, x)
char_cor(dat, x_list = NULL, ex_cols = "date$", parallel = FALSE,
         note = FALSE)
```

**Arguments**

- `dat`: A data frame.
- `x`: The name of variable to process.
- `x_list`: Names of independent variables.
- `ex_cols`: A list of excluded variables. Regular expressions can also be used to match variable names. Default is NULL.
- `parallel`: Logical, parallel computing. Default is FALSE.
- `note`: Logical. Outputs info. Default is TRUE.

**Value**

A list contains correlation index of x with other variables in dat.

**Examples**

```r
## Not run:
char_x_list = get_names(dat = UCICreditCard,
types = c('factor', 'character'),
ex_cols = "ID$|date$|default.payment.next.month$", get_ex = FALSE)
char_cor(dat = UCICreditCard[char_x_list])
## End(Not run)
```
**char_to_num**  

**character to number**

**Description**

`char_to_num` is for transferring character variables which are actually numerical numbers containing strings to numeric.

**Usage**

```r
char_to_num(dat, char_list = NULL, note = TRUE, ex_cols = NULL)
```

**Arguments**

- `dat`: A data frame
- `char_list`: The list of characteristic variables that need to merge categories. Default is NULL. In case of NULL, merge categories for all variables of string type.
- `note`: Logical, outputs info. Default is TRUE.
- `ex_cols`: A list of excluded variables. Regular expressions can also be used to match variable names. Default is NULL.

**Value**

A data.frame

**Examples**

```r
dat_sub = lendingclub[c('dti_joint','emp_length')]
str(dat_sub)
# variables that are converted to numbers containing strings
dat_sub = char_to_num(dat_sub)
str(dat_sub)
```

---

**checking_data**  

**Checking Data**

**Description**

`checking_data` checking dat before processing.

**Usage**

```r
checking_data(dat = NULL, target = NULL, occur_time = NULL, note = FALSE, pos_flag = NULL)
```
**check_rules**

### Arguments

- **dat**: A data.frame with independent variables and target variable.
- **target**: The name of target variable. Default is NULL.
- **occur_time**: The name of the variable that represents the time at which each observation takes place.
- **note**: Logical. Outputs info. Default is TRUE.
- **pos_flag**: The value of positive class of target variable, default: "1".

### Value

data.frame

### Examples

```r
dat = checking_data(dat = UCICreditCard, target = "default.payment.next.month")
```

---

**check_rules**

*check rules*

### Description

This function is used to check rules.

### Usage

```r
check_rules(rules_list, test_dat, target, value = NULL)
```

### Arguments

- **rules_list**: A list of rules.
- **test_dat**: A data.frame of test.
- **target**: The name of target variable.
- **value**: The name of value to gather.

### Value

A data frame with tree rules and percent under each rule.

### See Also

`get_ctree_rules`, `rules_filter`
Examples

```r
train_test = train_test_split(UCICreditCard, split_type = "Random", prop = 0.8, save_data = FALSE)
dat_train = train_test$train
dat_test = train_test$test
dat_train$default.payment.next.month = as.numeric(dat_train$default.payment.next.month)
rules_list = get_ctree_rules(tree_fit = NULL, train_dat = dat_train[, 8:26],
target = "default.payment.next.month", test_dat = dat_test)[1:3,2]
check_rules(rules_list = rules_list, target = "default.payment.next.month",
test_dat = dat_test, value = NULL)
```

description

This function is used for city variables derivation.

Usage

city_varieble(df = df, city_cols = NULL, city_pattern = NULL,
city_class = city_class, parallel = TRUE)

Arguments

df A data.frame.
city_cols Variables of city,
city_pattern Regular expressions, used to match city variable names. Default is "city$".
city_class Class or levels of cities.
parallel Logical, parallel computing. Default is TRUE.

description

This function is not intended to be used by end user.

Usage

city_varieble_process(df_city, x, city_class)

Arguments

df_city A data.frame.
x Variables of city,
city_class Class or levels of cities.
**cohort_analysis**

*cohort_analysis* is for cohort(vintage) analysis.

**Description**

*cohort_analysis* is for cohort(vintage) analysis.

**cohort_table**

**Usage**

```r
cohort_analysis(dat, obs_id = NULL, occur_time = NULL, MOB = NULL,
    period = "monthly", status = NULL, amount = NULL, by_out = "cnt",
    start_date = NULL, end_date = NULL, dead_status = 30)
```

```r
cohort_table(dat, obs_id = NULL, occur_time = NULL, MOB = NULL,
    period = "monthly", status = NULL, amount = NULL, by_out = "cnt",
    start_date = NULL, end_date = NULL, dead_status = 30)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dat</td>
<td>A data.frame contained id, occur_time, mob, status ...</td>
</tr>
<tr>
<td>obs_id</td>
<td>The name of ID of observations or key variable of data. Default is NULL.</td>
</tr>
<tr>
<td>occur_time</td>
<td>The name of the variable that represents the time at which each observation takes place.</td>
</tr>
<tr>
<td>MOB</td>
<td>Mobility of book</td>
</tr>
<tr>
<td>period</td>
<td>Period of event to analysis. Default is &quot;monthly&quot;</td>
</tr>
<tr>
<td>status</td>
<td>Status of observations</td>
</tr>
<tr>
<td>amount</td>
<td>The name of variable representing amount. Default is NULL.</td>
</tr>
<tr>
<td>by_out</td>
<td>Output: amount (amt) or count (cnt)</td>
</tr>
<tr>
<td>start_date</td>
<td>The earliest occurrence time of observations.</td>
</tr>
<tr>
<td>end_date</td>
<td>The latest occurrence time of observations.</td>
</tr>
<tr>
<td>dead_status</td>
<td>Status of dead observations</td>
</tr>
</tbody>
</table>

**cohort_table_plot**

*cohort_table_plot* is for plotting cohort(vintage) analysis table.

**Description**

This function is not intended to be used by end user.
Usage

cohort_table_plot(cohort_dat)

cohort_plot(cohort_dat)

Arguments

cohort_dat: A data.frame generated by cohort_analysis.

cor_heat_plot

Correlation Heat Plot

description

cor_heat_plot is for plotting correlation matrix

Usage

cor_heat_plot(cor_mat, low_color = love_color("deep_red"),
               high_color = love_color("light_cyan"), title = "Correlation Matrix")

Arguments

cor_mat: A correlation matrix.

low_color: color of the lowest correlation between variables.

high_color: color of the highest correlation between variables.

title: title of plot.

Examples

train_test <- train_test_split(UCICreditCard,
split_type = "Random", prop = 0.8, save_data = FALSE)
dat_train = train_test$train
dat_test = train_test$test
cor_mat = cor(dat_train[,8:12],use = "complete.obs")
cor_heat_plot(cor_mat)
**cor_plot**

**Correlation Plot**

**Description**

cor_plot is for plotting correlation matrix

**Usage**

cor_plot(dat, dir_path = tempfile(), x_list = NULL, gtitle = NULL, save_data = FALSE, plot_show = FALSE)

**Arguments**

dat A data.frame with independent variables and target variable.
dir_path The path for periodically saved graphic files. Default is "/model/LR"
x_list Names of independent variables.
gtitle The title of the graph & The name for periodically saved graphic file. Default is "_correlation_of_variables".save_data Logical, save results in locally specified folder. Default is TRUEplot_show Logical, show graph in current graphic device.

**Examples**

```r
train_test <- train_test_split(UCICreditCard,
    split_type = "Random", prop = 0.8, save_data = FALSE)
dat_train = train_test$train
dat_test = train_test$test
cor_plot(dat_train[,8:12], plot_show = TRUE)
```

**cos_sim**

**cos_sim**

**Description**

This function is not intended to be used by end user.

**Usage**

cos_sim(x, y, cos_margin = 1)

**Arguments**

x A list of numbers

y A list of numbers

`cos_margin` Margin of matrix, 1 for rows and 2 for cols, Default is 1.
**Value**

A number of cosin similarity

**Description**

`cross_table` is for cross table analysis.

**Usage**

```r
cross_table(dat, cross_x, cross_y, target = NULL, value = NULL, 
cross_type = "total_sum")
```

**Arguments**

- `dat` A data.frame with independent variables.
- `cross_x` Names of variables to cross.
- `cross_y` Names of variables to cross.
- `target` The name of target variable.
- `value` The name of the variable to sum. When this parameter is NULL, the default statistics is to sum frequency.
- `cross_type` Output form of the result of crosstable. Provide these four forms: "total_sum","total_pct","bad_sum","bad_pct".

**Value**

A cross table.

**Examples**

```r
cross_table(dat = UCICreditCard, cross_x = "SEX",cross_y = "AGE", 
target = "default.payment.next.month", cross_type = "bad_pct",value = "LIMIT_BAL")
cross_table(dat = UCICreditCard, cross_x = c("SEX", "MARRIAGE"), cross_y = "AGE", 
target = "default.payment.next.month", cross_type = "bad_pct",value = "LIMIT_BAL")
```
customer_segmentation

Description

customer_segmentation is a function for clustering and find the best segment variable.

Usage

customer_segmentation(dat, x_list = NULL, ex_cols = NULL,
cluster_control = list(meth = "Kmeans", kc = 2, nstart = 1, epsm =
1e-06, sf = 2, max_iter = 100), tree_control = list(cv_folds = 5,
maxdepth = kc + 1, minbucket = nrow(dat)/(kc + 1)), save_data = FALSE,
file_name = NULL, dir_path = tempdir())

Arguments

dat A data.frame contained only predict variables.
x_list A list of x variables.
ex_cols A list of excluded variables. Default is NULL.
cluster_control A list controls cluster. kc is the number of cluster center (default is 2), nstart
is the number of random groups (default is 1), max_iter max iteration num-
ber (default is 100).
  • meth Method of clustering. Provides two methods,"Kmeans" and "FCM(Fuzzy
Cluster Means)"(default is "Kmeans").
  • kc Number of cluster center (default is 2).
  • nstart Number of random groups (default is 1).
  • max_iter Max iteration number (default is 100).
tree_control A list of controls for desison tree to find the best segment variable.
  • cv_folds Number of cross-validations (default is 5).
  • maxdepth Maximum depth of a tree (default is kc +1).
  • minbucket Minimum percent of observations in any terminal <leaf> node
(default is nrow(dat) / (kc + 1)).
save_data Logical. If TRUE, save outliers analysis file to the specified folder at dir_path
file_name The name for periodically saved segmentation file. Default is NULL.
dir_path The path for periodically saved segmentation file.

Value

A "data.frame" object contains cluster results.
References


Examples

```r
clust <- customer_segmentation(dat = lendingclub[1:10000,20:30],
    x_list = NULL, ex_cols = "id$|loan_status",
    cluster_control = list(meth = "FCM", kc = 2), save_data = FALSE,
    tree_control = list(minbucket = round(nrow(lendingclub) / 10)),
    file_name = NULL, dir_path = tempdir())
```

---

cut_equal  Generating Initial Equal Size Sample Bins

Description

cut_equal is used to generate initial breaks for equal frequency binning.

Usage

cut_equal(dat_x, g = 10, sp_values = NULL, cut_bin = "equal_depth")

Arguments

dat_x  A vector of an variable x.
g  numeric, number of initial bins for equal_bins.
sp_values  a list of special value. Default: list(-1, "missing")
cut_bin  A string, 'equal_depth' or 'equal_width', default is 'equal_depth'.

See Also

get_breaks, get_breaks_all, get_tree_breaks

Examples

```r
#equal sample size breaks
equ_breaks = cut_equal(dat = UCICreditCard[, "PAY_AMT2"], g = 10)
```
### cv_split

**Description**

this function creates stratified folds for cross validation.

**Usage**

```r
cv_split(dat, k = 5, occur_time = NULL, seed = 46)
```

**Arguments**

- `dat`: A data.frame.
- `k`: k is an integer specifying the number of folds.
- `occur_time`: time variable for creating OOT folds. Default is NULL.
- `seed`: A seed. Default is 46.

**Value**

a list of indices

**Examples**

```r
sub = cv_split(UCICreditCard, k = 30)[[1]]
dat = UCICreditCard[sub,]
```

### data_cleansing

**Description**

The data_cleansing function is a simpler wrapper for data cleaning functions, such as delete variables that values are all NAs; checking dat and target format. delete low variance variables replace null or NULL or blank with NA; encode variables which NAs & miss value rate is more than 95 encode variables which unique value rate is more than 95 merge categories of character variables that is more than 10; transfer time variables to dateformation; remove duplicated observations; process outliers; process NAs.

**Usage**

```r
data_cleansing(dat, target = NULL, obs_id = NULL, occur_time = NULL, pos_flag = NULL, x_list = NULL, ex_cols = NULL, miss_values = NULL, remove_dup = TRUE, outlier_proc = TRUE, missing_proc = "median", low_var = 0.999, missing_rate = 0.98, merge_cat = 30, note = TRUE, parallel = FALSE, save_data = FALSE, file_name = NULL, dir_path = tempdir())
```
Arguments

dat        A data frame with x and target.
target     The name of target variable.
obs_id     The name of ID of observations. Default is NULL.
occur_time The name of occur time of observations. Default is NULL.
pos_flag  The value of positive class of target variable, default: "1".
x_list     A list of x variables.
ex_cols    A list of excluded variables. Default is NULL.
miss_values Other extreme value might be used to represent missing values, e.g: -9999, -9998. These miss_values will be encoded to -1 or "missing".
remove_dup Logical, if TRUE, remove the duplicated observations.
outlier_proc Logical, process outliers or not. Default is TRUE.
missing_proc If logical, process missing values or not. If "median", then Nas imputation with k neighbors median. If "avg_dist", the distance weighted average method is applied to determine the NAs imputation with k neighbors. If "default", assigning the missing values to -1 or "missing", otherwise, processing the missing values according to the results of missing analysis.
low_var    The maximum percent of unique values (including NAs) for filtering low variance variables.
missing_rate The maximum percent of missing values for recoding values to missing and non_missing.
merge_cat  The minimum number of categories for merging categories of character variables.
note      Logical. Outputs info. Default is TRUE.
parallel  Logical, parallel computing or not. Default is FALSE.
save_data Logical, save the result or not. Default is FALSE.
file_name  The name for periodically saved data file. Default is NULL.
dir_path   The path for periodically saved data file. Default is tempdir().

Value

A preprocessed data.frame

See Also

remove_duplicated, null_blank_na, entry_rate_na, low_variance_filter, process_nas, process_outliers
data_exploration

Examples

```r
# data cleaning
dat_cl <- data_cleansing(dat = UCICreditCard[1:2000,],
target = "default.payment.next.month",
x_list = NULL,
obs_id = "ID",
occur_time = "apply_date",
ex_cols = c("PAY_6|BILL_"),
outlier_proc = TRUE,
missing_proc = TRUE,
low_var = TRUE,
save_data = FALSE)
```

---

data_exploration Data Exploration

Description

The `data_exploration` includes both univariate and bivariate analysis and ranges from univariate statistics and frequency distributions, to correlations, cross-tabulation and characteristic analysis.

Usage

```r
data_exploration(dat, save_data = FALSE, file_name = NULL,
dir_path = tempdir())
```

Arguments

- **dat** A data.frame with x and target.
- **save_data** Logical. If TRUE, save files to the specified folder at `dir_path`
- **file_name** The file name for periodically saved outliers analysis file. Default is NULL.
- **dir_path** The path for periodically saved outliers analysis file. Default is `tempdir()`.

Value

A list contains both category and numeric variable analysis.

Examples

```r
data_ex = data_exploration(dat = UCICreditCard[1:1000,])
```
date_cut  

*Date Time Cut Point*

**Description**

date_cut is a small function to get date point.

**Usage**

date_cut(dat_time, pct = 0.7)

**Arguments**

dat_time  
time vectors.

pct  
the percent of cutting. Default: 0.7.

**Value**

A Date.

**Examples**

date_cut(dat_time = lendingclub$issue_d, pct = 0.8)  
#"2018-08-01"

---

derived_interval  

**Description**

This function is not intended to be used by end user.

**Usage**

derived_interval(dat_s, interval_type = c("cnt_interval", "time_interval"))

**Arguments**

dat_s  
A data.frame contained only predict variables.

interval_type  
Available of c("cnt_interval", "time_interval")
**derived_partial_acf**

---

### Description

This function is not intended to be used by end user.

### Usage

```r
derived_partial_acf(dat_s)
```

### Arguments

- **dat_s**
  
  A data.frame

---

**derived_pct**

---

### Description

This function is not intended to be used by end user.

### Usage

```r
derived_pct(dat_s, pct_type = "total_pct")
```

### Arguments

- **dat_s**
  
  A data.frame contained only predict variables.

- **pct_type**
  
  Available of "total_pct"
**derived_ts_vars**  

*Derivation of Behavioral Variables*

**Description**

This function is used for derivating behavioral variables and is not intended to be used by end user.

**Usage**

```r
derived_ts_vars(dat, grx = NULL, td = NULL, ID = NULL,  
ex_cols = NULL, x_list = NULL, der = c("cvs", "sums", "means",  
"maxs", "max_mins", "time_intervals", "cnt_intervals", "total_pcts",  
"cum_pcts", "partial_acfs"), parallel = TRUE, note = TRUE)
```

```r
derived_ts(dat, grx_x = NULL, x_list = NULL, td = NULL, ID = NULL,  
ex_cols = NULL, der = c("cvs", "sums", "means", "maxs", "max_mins",  
"time_intervals", "cnt_intervals", "total_pcts", "cum_pcts",  
"partial_acfs"))
```

**Arguments**

- **dat**  
  A data.frame contained only predict variables.
- **grx**  
  Regular expressions used to match variable names.
- **td**  
  Number of variables to derivate.
- **ID**  
  The name of ID of observations or key variable of data. Default is NULL.
- **ex_cols**  
  A list of excluded variables. Regular expressions can also be used to match variable names. Default is NULL.
- **x_list**  
  Names of independent variables.
- **der**  
  Variables to derivate
- **parallel**  
  Logical, parallel computing. Default is FALSE.
- **note**  
  Logical, outputs info. Default is TRUE.
- **grx_x**  
  Regular expression used to match a group of variable names.

**Details**

The key to creating a good model is not the power of a specific modelling technique, but the breadth and depth of derived variables that represent a higher level of knowledge about the phenomena under examination.
de_one_hot_encoding

Recovery One-Hot Encoding

Description

de_one_hot_encoding is for one-hot encoding recovery processing

Usage

de_one_hot_encoding(dat_one_hot, cat_vars = NULL, na_act = TRUE, note = FALSE)

Arguments

dat_one_hot A dat frame with the one hot encoding variables

cat_vars variables to be recovery processed, default is null, if null, find these variables through regular expressions.

na_act Logical. If true, the missing value is assigned as "missing", if FALSE missing value is omitted, the default is TRUE.

note Logical. Outputs info. Default is TRUE.

Value

A dat frame with the one hot encoding recovery character variables

See Also

one_hot_encoding

Examples

#one hot encoding
dat1 = one_hot_encoding(dat = UCICreditCard, cat_vars = c("SEX","MARRIAGE"), merge_cat = TRUE, na_act = TRUE)
#de one hot encoding
dat2 = de_one_hot_encoding(dat_one_hot = dat1, cat_vars = c("SEX","MARRIAGE"), na_act = FALSE)
de_percent

**Recovery Percent Format**

**Description**

`de_percent` is a small function for recovering percent format.

**Usage**

```
de_percent(x, digits = 2)
```

**Arguments**

- **x**: Character with percent format.
- **digits**: Number of digits. Default: 2.

**Value**

`x` without percent format.

**Examples**

```
de_percent("24%")
```

---

digits_num

**Number of digits**

**Description**

`digits_num` is for calculating optimal digits number for numeric variables.

**Usage**

```
digits_num(dat_x)
```

**Arguments**

- **dat_x**: A numeric variable.

**Value**

A number of digits
entropy_weight

Examples

```r
## Not run:
digits_num(lendingclub[,"dti"])
# 7
## End(Not run)
```

---

## entropy_weight

### Entropy Weight Method

#### Description

`entropy_weight` is for calculating Entropy Weight.

#### Usage

```r
entropy_weight(dat, ID = NULL, pos_vars, neg_vars)
```

#### Arguments

- `dat`: A data.frame with independent variables.
- `ID`: The name of ID variable.
- `pos_vars`: Names or index of positive direction variables, the bigger the better.
- `neg_vars`: Names or index of negative direction variables, the smaller the better.

#### Details

Step1 Raw data normalization
Step2 Find out the total amount of contributions of all samples to the index Xj
Step3 Each element of the step generated matrix is transformed into the product of each element and the LN (element), and the information entropy is calculated. Step4 Calculate redundancy. Step5 Calculate the weight of each index.

#### Value

A data.frame with weights of each variable.

#### Examples

```r
entropy_weight(dat = ewm_data, ID = "ID",
               pos_vars = -c(7,11),
               neg_vars = c(7,11))
```
entry_rate_na  

**Description**

entry_rate_na is the function to recode variables with missing values up to a certain percentage with missing and non_missing.

**Usage**

`entry_rate_na(dat, nr = 0.98, note = FALSE)`

**Arguments**

- **dat**  
  A data frame with x and target.
- **nr**  
  The maximum percent of NAs.
- **note**  
  Logical. Outputs info. Default is TRUE.

**Value**

A data.frame

**Examples**

```r
 datss = entry_rate_na(dat = lendingclub[1:1000, ], nr = 0.98)
```

---

euclid_dist  

**Description**

This function is not intended to be used by end user.

**Usage**

`euclid_dist(x, y, cos_margin = 1)`

**Arguments**

- **x**  
  A list
- **y**  
  A list
- **cos_margin**  
  rows or cols
**eval_auc**

### Functions of xgboost feval

**Description**

eval_auc, eval_ks, eval_lift, eval_tnr is for getting best params of xgboost.

**Usage**

eval_auc(preds, dtrain)
eval_ks(preds, dtrain)
eval_tnr(preds, dtrain)
eval_lift(preds, dtrain)

**Arguments**

- **preds**  
  A list of predict probability or score.
- **dtrain**  
  Matrix of x predictors.

**Value**

List of best value

---

**ewm.data**

### Entropy Weight Method Data

**Description**

This data is for Entropy Weight Method examples.

**Format**

A data frame with 10 rows and 13 variables.
Description

fast_high_cor_filter In a highly correlated variable group, select the variable with the highest IV.

high_cor_filter In a highly correlated variable group, select the variable with the highest IV.

Usage

fast_high_cor_filter(dat, p = 0.95, x_list = NULL, com_list = NULL, ex_cols = NULL, save_data = FALSE, cor_class = TRUE, vars_name = TRUE, parallel = FALSE, note = FALSE, file_name = NULL, dir_path = tempdir(), ...)

high_cor_filter(dat, com_list = NULL, x_list = NULL, ex_cols = NULL, onehot = TRUE, parallel = FALSE, p = 0.7, file_name = NULL, dir_path = tempdir(), save_data = FALSE, note = FALSE, ...)

Arguments

dat A data.frame with independent variables.
p Threshold of correlation between features. Default is 0.95.
x_list Names of independent variables.
com_list A data.frame with important values of each variable. eg : IV_list
ex_cols A list of excluded variables. Regular expressions can also be used to match variable names. Default is NULL.
save_data Logical, save results in locally specified folder. Default is FALSE.
cor_class Calculate catagery variables’s correlation matrix. Default is FALSE.
vars_name Logical, output a list of filtered variables or table with detailed compared value of each variable. Default is TRUE.
parallel Logical, parallel computing. Default is FALSE.
note Logical. Outputs info. Default is TRUE.
file_name The name for periodically saved results files. Default is “Feature_selected_COR”.
dir_path The path for periodically saved results files. Default is “./variable”.
... Additional parameters.
onehot one-hot-encoding independent variables.

Value

A list of selected variables.
feature_selector

See Also

get_correlation_group, high_cor_selector, char_cor_vars

Examples

# calculate iv for each variable.
iv_list = feature_selector(dat_train = UCICreditCard[1:1000,], dat_test = NULL,
target = "default.payment.next.month",
occur_time = "apply_date",
filter = c("IV"), cv_folds = 1, iv_cp = 0.01,
ex_cols = "ID$|date$|default.payment.next.month$",
save_data = FALSE, vars_name = FALSE)
fast_high_cor_filter(dat = UCICreditCard[1:1000,],
com_list = iv_list, save_data = FALSE,
ex_cols = "ID$|date$|default.payment.next.month$",
p = 0.9, cor_class = FALSE ,var_name = FALSE)

Description

feature_selector This function uses four different methods (IV, PSI, correlation, xgboost) in order to select important features. The correlation algorithm must be used with IV.

Usage

feature_selector(dat_train, dat_test = NULL, x_list = NULL,
target = NULL, pos_flag = NULL, occur_time = NULL,
ex_cols = NULL, filter = c("IV", "PSI", "XGB", "COR"),
cv_folds = 1, iv_cp = 0.01, psi_cp = 0.1, xgb_cp = 0,
cor_cp = 0.98, breaks_list = NULL, hopper = FALSE,
vars_name = TRUE, parallel = FALSE, note = TRUE, seed = 46,
save_data = FALSE, file_name = NULL, dir_path = tempdir(), ...)

Arguments

dat_train A data.frame with independent variables and target variable.
dat_test A data.frame of test data. Default is NULL.
x_list Names of independent variables.
target The name of target variable.
pos_flag The value of positive class of target variable, default: "1".
occur_time The name of the variable that represents the time at which each observation takes place.
ex_cols A list of excluded variables. Regular expressions can also be used to match variable names. Default is NULL.
feature_selector

filter The methods for selecting important and stable variables.
cv_folds Number of cross-validations. Default: 5.
iv_cp The minimum threshold of IV. 0 < iv_i; 0.01 to 0.1 usually work. Default: 0.02
psi_cp The maximum threshold of PSI. 0 <= psi_i <=1; 0.05 to 0.2 usually work. Default: 0.1
xgb_cp Threshold of XGB feature’s Gain. 0 <= xgb_cp <=1. Default is 1/number of independent variables.
cor_cp Threshold of correlation between features. 0 <= cor_cp <=1; 0.7 to 0.98 usually work. Default is 0.98.
breaks_list A table containing a list of splitting points for each independent variable. Default is NULL.
hopper Logical.Filtering screening. Default is FALSE.
vars_name Logical, output a list of filtered variables or table with detailed IV and PSI value of each variable. Default is FALSE.
parallel Logical, parallel computing. Default is FALSE.
ote Logical.Outputs info. Default is TRUE.
seed Random number seed. Default is 46.
save_data Logical, save results in locally specified folder. Default is FALSE.
file_name The name for periodically saved results files. Default is "select_vars".
dir_path The path for periodically saved results files. Default is "/variable"

Value
A list of selected features

See Also
psi_iv_filter, xgb_filter, gbm_filter

Examples
feature_selector(dat_train = UCICreditCard[1:1000,c(2,8:12,26)],
    dat_test = NULL, target = "default.payment.next.month",
    occur_time = "apply_date", filter = c("IV", "PSI"),
    cv_folds = 1, iv_cp = 0.01, psi_cp = 0.1, xgb_cp = 0, cor_cp = 0.98,
    vars_name = FALSE, note = FALSE)
**fuzzy_cluster_means**  
*Fuzzy Cluster means.*

**Description**

This function is used for Fuzzy Clustering.

**Usage**

```r
fuzzy_cluster_means(dat, kc = 2, sf = 2, nstart = 1,  
                   max_iter = 100, epsm = 1e-06)
```

```r
fuzzy_cluster(dat, kc = 2, init_centers, sf = 3, max_iter = 100,  
             epsm = 1e-06)
```

**Arguments**

- `dat`: A data.frame contained only predict variables.
- `kc`: The number of cluster center (default is 2).
- `sf`: Default is 2.
- `nstart`: The number of random groups (default is 1).
- `max_iter`: Max iteration number (default is 100).
- `epsm`: Default is 1e-06.
- `init_centers`: Initial centers of obs.

**References**

Bezdek, James C. "FCM: The fuzzy c-means clustering algorithm". Computers & Geosciences (0098-3004), [https://doi.org/10.1016/0098-3004(84)90020-7](https://doi.org/10.1016/0098-3004(84)90020-7)

---

**gather_data**  
*gather or aggregate data*

**Description**

This function is used for gathering or aggregating data.

**Usage**

```r
gather_data(dat, x_list = NULL, ID = NULL, FUN = sum_x)
```
Arguments

dat A data.frame contained only predict variables.
x_list The names of variables to gather.
ID The name of ID of observations or key variable of data. Default is NULL.
FUN The function of gathering method.

Details

The key to creating a good model is not the power of a specific modelling technique, but the breadth
and depth of derived variables that represent a higher level of knowledge about the phenomena under
examination.

Examples

dat = data.frame(id = c(1,1,2,2,3,3,4,4,4,4,4,4,5,5,6,7,7,8,8,8,1,4,3,4,8,2,7,1,
3,4,1,8,7,2,5,7,8,8,2,1,5,7,2,7,3)),
terms = c('a','b','c','a','c','d','d','a','c','b','c','d','a','c','c','b','c','f','b','c','f','b','c','h','h','i','c','d','g','k','k'),
time = c(8,3,1,9,6,1,4,9,1,3,4,8,2,7,1,
3,4,1,8,7,2,5,7,8,8,2,1,5,7,2,7,3))
gather_data(dat = dat, x_list = "time", ID = 'id', FUN = sum_x)

Description

gbm_filter is for selecting important features using GBM.

Usage

gbm_filter(dat, target = NULL, x_list = NULL, ex_cols = NULL,
pos_flag = NULL, GBM.params = gbm_params(), cores_num = 2,
vars_name = TRUE, note = TRUE, save_data = FALSE,
file_name = NULL, dir_path = tempdir(), seed = 46, ...)

Arguments

dat A data.frame with independent variables and target variable.
target The name of target variable.
x_list Names of independent variables.
ex_cols A list of excluded variables. Regular expressions can also be used to match
variable names. Default is NULL.
gbm_params

Description

gbm_params is the list of parameters to train a GBM using in training_model.

Usage

gbm_params(n.trees = 1000, interaction.depth = 6, shrinkage = 0.01, 
bag.fraction = 0.5, train.fraction = 0.7, n.minobsinnode = 30, 
cv.folds = 5, ...)

Value

Selected variables.

See Also

psi.iv.filter, xgb.filter, feature_selector

Examples

GBM.params = gbm_params(n.trees = 2, interaction.depth = 2, shrinkage = 0.1, 
  bag.fraction = 1, train.fraction = 1, 
  n.minobsinnode = 30, 
  cv.folds = 2)
  ## Not run:
  features <- gbm_filter(dat = UCICreditCard[1:1000, c(8:12, 26)], 
    target = "default.payment.next.month", 
    occur_time = "apply_date", 
    GBM.params = GBM.params 
    , vars_name = FALSE)
  ## End(Not run)

gbm_params | GBM Parameters

pos.flag The value of positive class of target variable, default: "1".
GBM.params Parameters of GBM.
cores.num The number of CPU cores to use.
vars.name Logical, output a list of filtered variables or table with detailed IV and PSI value
  of each variable. Default is TRUE.
note Logical, outputs info. Default is TRUE.
save.data Logical, save results results in locally specified folder. Default is FALSE.
file.name The name for periodically saved results files. Default is "Feature_importance_GBDT".
dir.path The path for periodically saved results files. Default is "/variable".
seed Random number seed. Default is 46.
... Other parameters to pass to gbdt_params.
Arguments

n.trees  Integer specifying the total number of trees to fit. This is equivalent to the number of iterations and the number of basis functions in the additive expansion. Default is 100.

depth  Integer specifying the maximum depth of each tree (i.e., the highest level of variable interactions allowed). A value of 1 implies an additive model, a value of 2 implies a model with up to 2-way interactions, etc. Default is 1.

shrinkage  a shrinkage parameter applied to each tree in the expansion. Also known as the learning rate or step-size reduction; 0.001 to 0.1 usually work, but a smaller learning rate typically requires more trees. Default is 0.1.

bag.fraction  the fraction of the training set observations randomly selected to propose the next tree in the expansion. This introduces randomness into the model fit. If bag.fraction < 1 then running the same model twice will result in similar but different fits. gbm uses the R random number generator so set.seed can ensure that the model can be reconstructed. Preferably, the user can save the returned gbm.object using save. Default is 0.5.

train.fraction  The first train.fraction * nrows(data) observations are used to fit the gbm and the remainder are used for computing out-of-sample estimates of the loss function.

n.minobsinnode  Integer specifying the minimum number of observations in the terminal nodes of the trees. Note that this is the actual number of observations, not the total weight.

cv.folds  Number of cross-validation folds to perform. If cv.folds > 1 then gbm, in addition to the usual fit, will perform a cross-validation, calculate an estimate of generalization error returned in cv.error.

...  Other parameters

Details

See details at: gbm

Value

A list of parameters.

See Also

training_model, lr_params, xgb_params, rf_params
**get_auc_ks_lambda**

description

get_auc_ks_lambda is for get best lambda required in lasso_filter. This function required in lasso_filter

Usage

```r
get_auc_ks_lambda(lasso_model, x_test, y_test, save_data = FALSE,
plot_show = TRUE, file_name = NULL, dir_path = tempdir())
```

Arguments

- **lasso_model**: A lasso model generated by glmnet.
- **x_test**: A matrix of test dataset with x.
- **y_test**: A matrix of y test dataset with y.
- **save_data**: Logical, save results in locally specified folder. Default is FALSE
- **plot_show**: Logical, if TRUE plot the results. Default is TRUE.
- **file_name**: The name for periodically saved results files. Default is NULL.
- **dir_path**: The path for periodically saved results files.

Value

Lambdva values with max K-S and AUC.

See Also

- `lasso_filter`, `get_sim_sign_lambda`

**get_bins_table_all**

*Table of Binning*

description

get_bins_table is used to generates summary information of variables. get_bins_table_all can generates bins table for all specified independent variables.
Usage

get_bins_table_all(dat, x_list = NULL, target = NULL, pos_flag = NULL, dat_test = NULL, ex_cols = NULL, breaks_list = NULL, parallel = FALSE, note = FALSE, bins_total = TRUE, save_data = FALSE, file_name = NULL, dir_path = tempdir())

get_bins_table(dat, x, target = NULL, pos_flag = NULL, dat_test = NULL, breaks = NULL, breaks_list = NULL, bins_total = TRUE, note = FALSE)

Arguments

dat A data.frame with independent variables and target variable.

x_list Names of independent variables.

target The name of target variable.

pos_flag Value of positive class. Default is "1".

dat_test A data.frame of test data. Default is NULL.

ex_cols A list of excluded variables. Regular expressions can also be used to match variable names. Default is NULL.

breaks_list A table containing a list of splitting points for each independent variable. Default is NULL.

parallel Logical, parallel computing. Default is FALSE.

note Logical, outputs info. Default is TRUE.

bins_total Logical, total sum for each columns.

save_data Logical, save results in locally specified folder. Default is FALSE.

file_name The name for periodically saved bins table file. Default is "bins_table".

dir_path The path for periodically saved bins table file. Default is "/variable".

x The name of an independent variable.

breaks Splitting points for an independent variable. Default is NULL.

See Also

get_iv, get_iv_all, get_psi, get_psi_all

Examples

breaks_list = get_breaks_all(dat = UCICreditCard, x_list = names(UCICreditCard)[3:4], target = "default.payment.next.month", equal_bins = TRUE, best = FALSE, g = 5, ex_cols = "ID|apply_date", save_data = FALSE)
get_bins_table_all(dat = UCICreditCard, breaks_list = breaks_list, target = "default.payment.next.month")
get_breaks_all

**Generate Best Breaks for Binning**

**Description**

get_breaks is for generating optimal binning for numerical and nominal variables. The get_breaks_all is a simpler wrapper for get_breaks.

**Usage**

```r
get_breaks_all(dat, target = NULL, x_list = NULL, ex_cols = NULL,
    pos_flag = NULL, occur_time = NULL, oot_pct = 0.7, best = TRUE,
    equal_bins = FALSE, cut_bin = "equal_depth", g = 10,
    sp_values = NULL, tree_control = list(p = 0.05, cp = 1e-06, xval = 5,
        maxdepth = 10), bins_control = list(bins_num = 10, bins_pct = 0.05,
        b_chi = 0.05, b_odds = 0.1, b_psi = 0.05, b_or = 0.15, mono = 0.3,
        odds_psi = 0.2, kc = 1), parallel = FALSE, note = FALSE,
    save_data = FALSE, file_name = NULL, dir_path = tempdir(), ...)
```

```r
get_breaks(dat, x, target = NULL, pos_flag = NULL, best = TRUE,
    equal_bins = FALSE, cut_bin = "equal_depth", g = 10,
    sp_values = NULL, occur_time = NULL, oot_pct = 0.7,
    tree_control = NULL, bins_control = NULL, note = FALSE, ...)
```

**Arguments**

- **dat**
  A data frame with x and target.
- **target**
  The name of target variable.
- **x_list**
  A list of x variables.
- **ex_cols**
  A list of excluded variables. Default is NULL.
- **pos_flag**
  The value of positive class of target variable, default: "1".
- **occur_time**
  The name of the variable that represents the time at which each observation takes place.
- **oot_pct**
  Percentage of observations retained for overtime test (especially to calculate PSI). Default is 0.7
- **best**
  Logical, if TRUE, merge initial breaks to get optimal breaks for binning.
- **equal_bins**
  Logical, if TRUE, equal sample size initial breaks generates.If FALSE , tree breaks generates using desision tree.
- **cut_bin**
  A string, if equal_bins is TRUE, 'equal_depth' or 'equal_width', default is 'equal_depth'.
- **g**
  Integer, number of initial bins for equal_bins.
- **sp_values**
  A list of missing values.
- **tree_control**
  the list of tree parameters.
• \( p \) the minimum percent of observations in any terminal <leaf> node. \( 0 < p < 1 \); 0.01 to 0.1 usually work.
• \( cp \) complexity parameter. the larger, the more conservative the algorithm will be. \( 0 < cp < 1 \); 0.0001 to 0.000001 usually work.
• \( xval \) number of cross-validations. Default: 5
• \( max\_depth \) maximum depth of a tree. Default: 10

**bins_control** the list of parameters.

• \( bins\_num \) The maximum number of bins. 5 to 10 usually work. Default: 10
• \( bins\_pct \) The minimum percent of observations in any bins. \( 0 < bins\_pct < 1 \); 0.01 to 0.1 usually work. Default: 0.02
• \( b\_chi \) The minimum threshold of chi-square merge. \( 0 < b\_chi < 1 \); 0.01 to 0.1 usually work. Default: 0.02
• \( b\_odds \) The minimum threshold of odds merge. \( 0 < b\_odds < 1 \); 0.05 to 0.2 usually work. Default: 0.1
• \( b\_psi \) The maximum threshold of PSI in any bins. \( 0 < b\_psi < 1 \); 0 to 0.1 usually work. Default: 0.05
• \( b\_or \) The maximum threshold of G/B index in any bins. \( 0 < b\_or < 1 \); 0.05 to 0.3 usually work. Default: 0.15
• \( odds\_psi \) The maximum threshold of Training and Testing G/B index PSI in any bins. \( 0 < odds\_psi < 1 \); 0.01 to 0.3 usually work. Default: 0.1
• \( mono \) Monotonicity of all bins, the larger, the more nonmonotonic the bins will be. \( 0 < mono < 0.5 \); 0.2 to 0.4 usually work. Default: 0.2
• \( kc \) number of cross-validations. 1 to 5 usually work. Default: 1

**parallel** Logical, parallel computing or not. Default is FALSE.
**note** Logical. Outputs info. Default is TRUE.
**save_data** Logical, save results in locally specified folder. Default is TRUE
**file_name** File name that save results in locally specified folder. Default is "breaks_list".
**dir_path** Path to save results. Default is "/variable"

... Additional parameters.

\( x \) The Name of an independent variable.

**Value**
A table containing a list of splitting points for each independent variable.

**See Also**

`get_tree_breaks, cut_equal, select_best_class, select_best_breaks`

**Examples**

```r
#controls
tree_control = list(p = 0.02, cp = 0.000001, xval = 5, maxdepth = 10)
bins_control = list(bins_num = 10, bins_pct = 0.02, b_chi = 0.02, b_odds = 0.1,
```
get_correlation_group

get_correlation_group

Description

get_correlation_group is function for obtaining highly correlated variable groups. select_cor_group is function for selecting highly correlated variable group. select_cor_list is function for selecting highly correlated variable list.

Usage

get_correlation_group(cor_mat, p = 0.8)

select_cor_group(cor_vars)

select_cor_list(cor_vars_list)

Arguments

cor_mat A correlation matrix of independent variables.
p Threshold of correlation between features. Default is 0.7.
cor_vars Correlated variables.
cor_vars_list List of correlated variable

Value

A list of selected variables.
get_ctree_rules

Parse decision tree rules

Description

get_ctree_rules This function is used to desision tree rules and percentage of 1 under each rule.

Usage

get_ctree_rules(tree_fit = NULL, train_dat = NULL, target = NULL, test_dat = NULL, tree_control = list(p = 0.05, cp = 1e-04, xval = 1, maxdepth = 10), seed = 46)

Arguments

tree_fit  A tree model object.
train_dat  A data.frame of train.
target    The name of target variable.
test_dat  A data.frame of test.
tree_control  the list of parameters to control cutting initial breaks by decision tree.
seed  Random number seed. Default is 46.

Value

A data frame with tree rules and 1 percent under each rule.

See Also

rules_filter, check_rules

Examples

train_test <- train_test_split(UCICreditCard, split_type = "Random", prop = 0.8, save_data = FALSE)

dat_train = train_test$train

dat_test = train_test$test

dat_train$default.payment.next.month = as.numeric(dat_train$default.payment.next.month)

get_ctree_rules(tree_fit = NULL, train_dat = dat_train[, 8:26], target = "default.payment.next.month", test_dat = dat_test)
Calculate Information Value (IV) get_iv is used to calculate Information Value (IV) of an independent variable. get_iv_all can loop through IV for all specified independent variables.

**Usage**

```r
get_iv_all(dat, x_list = NULL, ex_cols = NULL, breaks_list = NULL, target = NULL, pos_flag = NULL, best = TRUE, equal_bins = FALSE, tree_control = NULL, bins_control = NULL, g = 10, parallel = FALSE, note = FALSE)
```

```r
get_iv(dat, x, target = NULL, pos_flag = NULL, breaks = NULL, breaks_list = NULL, best = TRUE, equal_bins = FALSE, tree_control = NULL, bins_control = NULL, g = 10, note = FALSE)
```

**Arguments**

- `dat`: A data.frame with independent variables and target variable.
- `x_list`: Names of independent variables.
- `ex_cols`: A list of excluded variables. Regular expressions can also be used to match variable names. Default is NULL.
- `breaks_list`: A table containing a list of splitting points for each independent variable. Default is NULL.
- `target`: The name of target variable. Default is NULL.
- `pos_flag`: Value of positive class, Default is "1".
- `best`: Logical, merge initial breaks to get optimal breaks for binning.
- `equal_bins`: Logical, generates initial breaks for equal frequency binning.
- `tree_control`: Parameters of using Decision Tree to segment initial breaks. See details: `get_tree_breaks`
- `bins_control`: Parameters used to control binning. See details: `select_best_class`, `select_best_breaks`
- `g`: Number of initial breakpoints for equal frequency binning.
- `parallel`: Logical, parallel computing. Default is FALSE.
- `note`: Logical, outputs info. Default is TRUE.
- `x`: The name of an independent variable.
- `breaks`: Splitting points for an independent variable. Default is NULL.
get_logistic_coef

Details
IV Rules of Thumb for evaluating the strength a predictor. Less than 0.02: unpredictable. 0.02 to 0.1: weak. 0.1 to 0.3: medium. 0.3+: strong.

References
Information Value Statistic: Bruce Lund, Magnify Analytics Solutions, a Division of Marketing Associates, Detroit, MI (Paper AA - 14 - 2013)

See Also
get_iv, get_iv_all, get_psi, get_psi_all

Examples
get_iv_all(dat = UCICreditCard,
x_list = names(UCICreditCard)[3:10],
equal_bins = TRUE, best = FALSE,
target = "default.payment.next.month",
ex_cols = "ID|apply_date")
get_iv(UCICreditCard, x = "PAY_3",
equal_bins = TRUE, best = FALSE,
target = "default.payment.next.month")

Description
get_logistic_coef is for getting logistic coefficients.

Usage
get_logistic_coef(lg_model, file_name = NULL, dir_path = tempdir(),
save_data = FALSE)

Arguments
lg_model An object of logistic model.
file_name The name for periodically saved coefficient file. Default is "LR_coef".
dir_path The Path for periodically saved coefficient file. Default is "/model".
save_data Logical, save the result or not. Default is FALSE.

Value
A data.frame with logistic coefficients.
get_median

get central value.

Examples

# dataset splitting
sub = cv_split(UCICreditCard, k = 30)[[1]]
dat = UCICreditCard[sub,]
# rename the target variable
dat = re_name(dat, "default.payment.next.month", "target")
dat = data_cleansing(dat, target = "target", obs_id = "ID",
occur_time = "apply_date", miss_values = list("", -1))
# train test splitting
train_test <- train_test_split(dat, split_type = "OOT", prop = 0.7,
occur_time = "apply_date")
dat_train = train_test$train
dat_test = train_test$test
# get breaks of all predictive variables
x_list = c("PAY_0", "LIMIT_BAL", "PAY_AMT5", "EDUCATION", "PAY_3", "PAY_2")
breaks_list <- get_breaks_all(dat = dat_train, target = "target",
x_list = x_list, occur_time = "apply_date", ex_cols = "ID",
save_data = FALSE, note = FALSE)
# WOE transforming
train_woe = woe_trans_all(dat = dat_train,
target = "target",
breaks_list = breaks_list,
woe_name = FALSE)
test_woe = woe_trans_all(dat = dat_test,
target = "target",
breaks_list = breaks_list,
note = FALSE)
Formula = as.formula(paste("target", paste(x_list, collapse = ' + '), sep = ' ~ '))
set.seed(46)
lr_model = glm(Formula, data = train_woe[, c("target", x_list)], family = binomial(logit))
# get LR coefficient
dt_imp_LR = get_logistic_coef(lg_model = lr_model, save_data = FALSE)
bins_table = get_bins_table_all(dat = dat_train, target = "target",
x_list = x_list, dat_test = dat_test,
breaks_list = breaks_list, note = FALSE)
# score card
LR_score_card <- get_score_card(lg_model = lr_model, bins_table, target = "target")
# scoring
train_pred = dat_train[, c("ID", "apply_date", "target")]
test_pred = dat_test[, c("ID", "apply_date", "target")]
train_pred$post LR = score_transfer(model = lr_model,
tbl_woe = train_woe,
save_data = TRUE)[, "score"]
test_pred$post LR = score_transfer(model = lr_model,
tbl_woe = test_woe, save_data = FALSE)[, "score"]
**get_names**

**Description**

This function is not intended to be used by end user.

**Usage**

\[
\text{get_median}(x, \text{ weight_avg } = \text{ NULL})
\]

**Arguments**

- \(x\): A vector or list.
- \(\text{weight_avg}\): avg weight to calculate means.

---

**get_names**

*Get Variable Names*

**Description**

get_names is for getting names of particular classes of variables

**Usage**

\[
\text{get_names}(\text{dat}, \text{ types } = \text{ c('logical', 'factor', 'character', 'numeric', 'integer64', 'integer', 'double', 'Date', 'POSIXlt', 'POSIXct', 'POSIXt')}, \text{ ex_cols } = \text{ NULL}, \text{ get_ex } = \text{ FALSE})
\]

**Arguments**

- \(\text{dat}\): A data.frame with independent variables and target variable.
- \(\text{types}\): The class or types of variables which names to get. Default: \(\text{c('numeric', 'integer', 'double')}\)
- \(\text{ex_cols}\): A list of excluded variables. Regular expressions can also be used to match variable names. Default is NULL.
- \(\text{get_ex}\): Logical, if TRUE, return a list contains names of excluded variables.

**Value**

A list contains names of variables

**See Also**

- get_x_list

**Examples**

\[
\text{x_list} = \text{get_names}(\text{dat} = \text{UCICreditCard}, \text{ types } = \text{ c('factor', 'character')}, \\
\text{ex_cols} = \text{ c('default.payment.next.month', 'ID$|_date$')}, \text{ get_ex } = \text{ FALSE})
\]

\[
\text{x_list} = \text{get_names}(\text{dat} = \text{UCICreditCard}, \text{ types } = \text{ c('numeric', 'character', 'integer')}, \\
\text{ex_cols} = \text{ c('default.payment.next.month', 'ID$|SEX ')}), \text{ get_ex } = \text{ FALSE})
\]
### Description

This function is not intended to be used by end user.

### Usage

```r
get_nas_random(dat)
```

### Arguments

- **dat**  
  A data.frame contained only predict variables.

---

### Description

You can use the `plot_vars` to produce plots that characterize the frequency or the distribution of your data. `get_plots` can loop through plots for all specified independent variables.

### Usage

```r
get_plots(dat_train, dat_test = NULL, x_list = NULL, target = NULL,
          ex_cols = NULL, breaks_list = NULL, pos_flag = NULL,
          equal_bins = FALSE, cut_bin = "equal_depth", best = TRUE, g = 20,
          tree_control = NULL, bins_control = NULL, plot_show = TRUE,
          save_data = FALSE, file_name = NULL, parallel = FALSE,
          g_width = 8, dir_path = tempdir())
```

```r
plot_vars(dat_train, x, target, dat_test = NULL, g_width = 8,
         breaks_list = NULL, breaks = NULL, pos_flag = list("1", 1, "bad",
         "positive"), equal_bins = TRUE, cut_bin = "equal_depth",
         best = FALSE, g = 10, tree_control = NULL, bins_control = NULL,
         plot_show = TRUE, save_data = FALSE, dir_path = tempdir())
```

### Arguments

- **dat_train**  
  A data.frame with independent variables and target variable.
- **dat_test**  
  A data.frame of test data. Default is NULL.
- **x_list**  
  Names of independent variables.
- **target**  
  The name of target variable.
get_psi_all

**Description**

Calculate Population Stability Index (PSI) get_psi is used to calculate Population Stability Index (PSI) of an independent variable. get_psi_all can loop through PSI for all specified independent variables.

**Examples**

```r
train_test <- train_test_split(UCICreditCard[1:1000,], split_type = "Random",
                            prop = 0.8, save_data = FALSE)
dat_train = train_test$train
dat_test = train_test$test
get_plots(dat_train[, c(8, 26)], dat_test = dat_test[, c(8, 26)],
             target = "default.payment.next.month")
```
Usage

get_psi_all(dat, x_list = NULL, target = NULL, dat_test = NULL,
breaks_list = NULL, occur_time = NULL, start_date = NULL,
cut_date = NULL, oot_pct = 0.7, pos_flag = NULL,
parallel = FALSE, ex_cols = NULL, as_table = FALSE, g = 10,
bins_no = TRUE, note = FALSE)

get_psi(dat, x, target = NULL, dat_test = NULL, occur_time = NULL,
start_date = NULL, cut_date = NULL, pos_flag = NULL,
breaks = NULL, breaks_list = NULL, oot_pct = 0.7, g = 10,
as_table = TRUE, note = FALSE, bins_no = TRUE)

Arguments

dat          A data.frame with independent variables and target variable.
x_list       Names of independent variables.
target       The name of target variable.
dat_test     A data.frame of test data. Default is NULL.
breaks_list  A table containing a list of splitting points for each independent variable. Default is NULL.
occur_time   The name of the variable that represents the time at which each observation takes place.
start_date   The earliest occurrence time of observations.
cut_date     Time points for splitting data sets, e.g.: splitting Actual and Expected data sets.
oot_pct      Percentage of observations retained for overtime test (especially to calculate PSI). Default is 0.7
pos_flag     Value of positive class, Default is "1".
parallel     Logical, parallel computing. Default is FALSE.
ex_cols      Names of excluded variables. Regular expressions can also be used to match variable names. Default is NULL.
as_table     Logical, output results in a table. Default is TRUE.
g           Number of initial breakpoints for equal frequency binning.
bins_no      Logical, add serial numbers to bins. Default is TRUE.
note         Logical, outputs info. Default is TRUE.
x           The name of an independent variable.
breaks       Splitting points for an independent variable. Default is NULL.

Details

PSI Rules for evaluating the stability of a predictor
Less than 0.02: Very stable
0.02 to 0.1: Stable
0.1 to 0.2: Unstable
0.2 to 0.5: Change more than 0.5: Great change
get_psi_iv_all

See Also

get_iv, get_iv_all, get_psi, get_psi_all

Examples

# dat_test is null
get_psi(dat = UCICreditCard, x = "PAY_3", occur_time = "apply_date")
# dat_test is not all
# train_test split
train_test = train_test_split(dat = UCICreditCard, prop = 0.7, split_type = "OOT",
  occur_time = "apply_date", start_date = NULL, cut_date = NULL,
  save_data = FALSE, note = FALSE)
dat_ex = train_test$train
dat_ac = train_test$test
# generate psi table
get_psi(dat = dat_ex, dat_test = dat_ac, x = "PAY_3",
  occur_time = "apply_date", bins_no = TRUE)

get_psi_iv_all Calculate IV & PSI

Description

get_iv_psi is used to calculate Information Value (IV) and Population Stability Index (PSI) of an independent variable. get_iv_psi_all can loop through IV & PSI for all specified independent variables.

Usage

get_psi_iv_all(dat, dat_test = NULL, x_list = NULL, target,
ex_cols = NULL, pos_flag = NULL, breaks_list = NULL,
  occur_time = NULL, oot_pct = 0.7, equal_bins = FALSE,
cut_bin = "equal_depth", tree_control = NULL, bins_control = NULL,
  bins_total = FALSE, best = TRUE, g = 10, as_table = TRUE,
  note = FALSE, parallel = FALSE, bins_no = TRUE)
get_psi_iv(dat, dat_test = NULL, x, target, pos_flag = NULL,
  breaks = NULL, breaks_list = NULL, occur_time = NULL,
  oot_pct = 0.7, equal_bins = FALSE, cut_bin = "equal_depth",
tree_control = NULL, bins_control = NULL, bins_total = FALSE,
  best = TRUE, g = 10, as_table = TRUE, note = FALSE,
  bins_no = TRUE)

Arguments

dat A data.frame with independent variables and target variable.
dat_test A data.frame of test data. Default is NULL.
get_psi_iv_all

x_list Names of independent variables.
target The name of target variable.
ex_cols A list of excluded variables. Regular expressions can also be used to match variable names. Default is NULL.
pos_flag The value of positive class of target variable, default: "1".
breaks_list A table containing a list of splitting points for each independent variable. Default is NULL.
occur_time The name of the variable that represents the time at which each observation takes place.
oot_pct Percentage of observations retained for overtime test (especially to calculate PSI). Default is 0.7.
equal_bins Logical, generates initial breaks for equal frequency or width binning.
cut_bin A string, if equal_bins is TRUE, 'equal_depth' or 'equal_width', default is 'equal_depth'.
tree_control Parameters of using Decision Tree to segment initial breaks. See details: get_tree_breaks
bins_control Parameters used to control binning. See details: select_best_class, select_best_breaks
bins_total Logical, total sum for each variable.
best Logical, merge initial breaks to get optimal breaks for binning.
g Number of initial breakpoints for equal frequency binning.
as_table Logical, output results in a table. Default is TRUE.
note Logical, outputs info. Default is TRUE.
parallel Logical, parallel computing. Default is FALSE.
bins_no Logical, add serial numbers to bins. Default is FALSE.
x The name of an independent variable.
breaks Splitting points for an independent variable. Default is NULL.

See Also
get_iv, get_iv_all, get_psi, get_psi_all

Examples

iv_list = get_psi_iv_all(dat = UCICreditCard[,1:1000],
x_list = names(UCICreditCard)[3:5], equal_bins = TRUE,
target = "default.payment.next.month", ex_cols = "ID|apply_date")
get_psi_iv(UCICreditCard, x = "PAY_3",
target = "default.payment.next.month", bins_total = TRUE)
get_psi_plots

Plot PSI (Population Stability Index)

Description

You can use the psi_plot to plot PSI of your data. get_psi_plots can loop through plots for all specified independent variables.

Usage

get_psi_plots(dat_train, dat_test = NULL, x_list = NULL, ex_cols = NULL, breaks_list = NULL, occur_time = NULL, g = 10, plot_show = TRUE, save_data = FALSE, file_name = NULL, parallel = FALSE, g_width = 8, dir_path = tempdir())

psi_plot(dat_train, x, dat_test = NULL, occur_time = NULL, g_width = 8, breaks_list = NULL, breaks = NULL, g = 10, plot_show = TRUE, save_data = FALSE, dir_path = tempdir())

Arguments

dat_train A data.frame with independent variables.
dat_test A data.frame of test data. Default is NULL.
x_list Names of independent variables.
ex_cols A list of excluded variables. Regular expressions can also be used to match variable names. Default is NULL.
brakes_list A table containing a list of splitting points for each independent variable. Default is NULL.
occur_time The name of occur time.
g Number of initial breakpoints for equal frequency binning.
plot_show Logical, show model performance in current graphic device. Default is FALSE.
save_data Logical, save results in locally specified folder. Default is FALSE.
file_name The name for periodically saved data file. Default is NULL.
parallel Logical, parallel computing. Default is FALSE.
g_width The width of graphs.
dir_path The path for periodically saved graphic files.
x The name of an independent variable.
brakes Splitting points for a continues variable.
get_score_card

Examples

```r
train_test <- train_test_split(UCICreditCard[1:1000,], split_type = "Random", prop = 0.8, save_data = FALSE)
dat_train = train_test$train
dat_test = train_test$test
get_psi_plots(dat_train[, c(8, 9)], dat_test = dat_test[, c(8, 9)])
```

get_score_card

Score Card

Description

get_score_card is for generating a standard scorecard

Usage

```r
get_score_card(lg_model, target, bins_table, a = 600, b = 50,
file_name = NULL, dir_path = tempdir(), save_data = FALSE)
```

Arguments

- `lg_model`: An object of glm model.
- `target`: The name of target variable.
- `bins_table`: a data.frame generated by `get_bins_table`
- `a`: Base line of score.
- `b`: Numeric. Increased scores from doubling Odds.
- `file_name`: The name for periodically saved scorecard file. Default is "LR_Score_Card".
- `dir_path`: The path for periodically saved scorecard file. Default is "/model".
- `save_data`: Logical, save results in locally specified folder. Default is FALSE.

Value

scorecard

Examples

```r
# dataset splitting
sub = cv_split(UCICreditCard, k = 30)[[1]]
dat = UCICreditCard[sub,]
# rename the target variable
dat = re_name(dat, "default.payment.next.month", "target")
dat = data_cleaning(dat, target = "target", obs_id = "ID",
occur_time = "apply_date", miss_values = list("", -1))
# train test splitting
train_test <- train_test_split(dat, split_type = "OOT", prop = 0.7,
occur_time = "apply_date")
```
dat_train = train_test$train
dat_test = train_test$test

# get breaks of all predictive variables
x_list = c("PAY_0", "LIMIT_BAL", "PAY_AMT5", "EDUCATION", "PAY_3", "PAY_2")
breaks_list <- get_breaks_all(dat = dat_train, target = "target",
                            x_list = x_list, occur_time = "apply_date", ex_cols = "ID",
                            save_data = FALSE, note = FALSE)

# woe transforming
train_woe = woe_trans_all(dat = dat_train,
                          target = "target",
                          breaks_list = breaks_list,
                          woe_name = FALSE)
test_woe = woe_trans_all(dat = dat_test,
                          target = "target",
                          breaks_list = breaks_list,
                          note = FALSE)

Formula = as.formula(paste("target", paste(x_list, collapse = " + "), sep = " - "))
set.seed(46)
lr_model = glm(Formula, data = train_woe[, c("target", x_list)], family = binomial(logit))

# get LR coefficient
dt_imp_LR = get_logistic_coef(lg_model = lr_model, save_data = FALSE)
bins_table = get_bins_table_all(dat = dat_train, target = "target",
                                dat_test = dat_test,
                                x_list = x_list,
                                breaks_list = breaks_list, note = FALSE)

# score card
LR_score_card <- get_score_card(lg_model = lr_model, bins_table, target = "target")

# scoring
train_pred = dat_train[, c("ID", "apply_date", "target")]
test_pred = dat_test[, c("ID", "apply_date", "target")]
train_pred$pred_LR = score_transfer(model = lr_model,
                                    tbl_woe = train_woe,
                                    save_data = FALSE)[, "score"]
test_pred$pred_LR = score_transfer(model = lr_model,
                                   tbl_woe = test_woe, save_data = FALSE)[, "score"]

---

get_shadow_nas

Description

This function is not intended to be used by end user.

Usage

get_shadow_nas(dat)

Arguments

dat A data.frame contained only predict variables.
get_sim_sign_lambda

Description

get_sim_sign_lambda is for get Best lambda required in lasso_filter. This function required in lasso_filter.

Usage

get_sim_sign_lambda(lasso_model, sim_sign = "negative")

Arguments

lasso_model: A lasso model generated by glmnet.
sim_sign: Default is "negative". This is related to pos_flag. If pos_flag equals 1 or 1, the value must be set to negative. If pos_flag equals 0 or 0, the value must be set to positive.

Details

lambda.sim_sign give the model with the same positive or negative coefficients of all variables.

Value

Lambda value

get_tree_breaks

Description

get_tree_breaks is for generating initial breaks by decision tree for a numerical or nominal variable. The get_breaks function is a simpler wrapper for get_tree_breaks.

Usage

get_tree_breaks(dat, x, target, pos_flag = NULL, tree_control = list(p = 0.02, cp = 1e-06, xval = 5, maxdepth = 10), sp_values = NULL)
get_x_list

**Arguments**

- **dat** A data frame with x and target.
- **x** name of variable to cut breaks by tree.
- **target** The name of target variable.
- **pos_flag** The value of positive class of target variable, default: "1".
- **tree_control** the list of parameters to control cutting initial breaks by decision tree.
  - p the minimum percent of observations in any terminal <leaf> node. 0 < p< 1; 0.01 to 0.1 usually work.
  - cp complexity parameter. the larger, the more conservative the algorithm will be. 0 < cp< 1 ; 0.0001 to 0.0000001 usually work.
  - xval number of cross-validations.Default: 5
  - max_depth maximum depth of a tree. Default: 10
- **sp_values** A list of special value. Default: NULL.

**See Also**

get_breaks, get_breaks_all

**Examples**

```r
#tree breaks
tree_control = list(p = 0.02, cp = 0.000001, xval = 5, maxdepth = 10)
tree_breaks = get_tree_breaks(dat = UCICreditCard, x = "MARRIAGE", target = "default.payment.next.month", tree_control = tree_control)
```

---

get_x_list

**Get X List.**

**Description**

get_x_list is for getting intersect names of x_list, train and test.

**Usage**

get_x_list(dat_train = NULL, dat_test = NULL, x_list = NULL, ex_cols = NULL, note = FALSE)

**Arguments**

- **dat_train** A data.frame with independent variables.
- **dat_test** Another data.frame.
- **x_list** Names of independent variables.
- **ex_cols** A list of excluded variables. Regular expressions can also be used to match variable names. Default is NULL.
- **note** Logical. Outputs info. Default is TRUE.
high_cor_selector

Value

A list contains names of variables

See Also

generate_names

Examples

x_list = get_x_list(x_list = NULL, dat_train = UCICreditCard,
ex_cols = c("default.payment.next.month","ID$\_date$"))

Description

high_cor_selector is function for comparing the two highly correlated variables, select a variable
with the largest IV value.

Usage

high_cor_selector(cor_mat, p = 0.95, x_list = NULL, com_list = NULL,
retain = TRUE)

Arguments

cor_mat A correlation matrix.
p The threshold of high correlation.
x_list Names of independent variables.
com_list A data.frame with important values of each variable. eg: IV_list.
retain Logical, output selected variables, if FALSE, output filtered variables.

Value

A list of selected variables.
is_date

is_date

Description

is_date is a small function for distinguishing time formats

Usage

is_date(x)

Arguments

x list or vectors

Value

A Date.

Examples

is_date(lendingclub$issue_d)

knn_nas_imp

Imputate nas using KNN

Description

This function is not intended to be used by end user.

Usage

knn_nas_imp(dat, x, nas_rate = NULL, mat_nas_shadow = NULL,
dt_nas_random = NULL, k = 10, scale = FALSE, method = "median")

Arguments

dat A data.frame with independent variables.
x The name of variable to process.
nas_rate A list contains nas rate of each variable.
mat_nas_shadow A shadow matrix of variables which contain nas.
dt_nas_random A data.frame with random nas imputation.
k Number of neighbors of each obs which x is missing.
scale Logical.Standardization of variable.
method The methods of imputation by knn. "median" is knn imputation with k neighbors median, "avg_dist" is knn imputation with k neighbors of distance weighted mean.
ks_table

ks_table & plot

Description

ks_table is for generating a model performance table. ks_table_plot is for plotting the table generated by ks_table. ks_psi_plot is for K-S & PSI distribution plotting.

Usage

ks_table(train_pred, test_pred = NULL, target = NULL, score = NULL, g = 10, breaks = NULL, pos_flag = list("1", "1", "Bad", 1))

ks_table_plot(train_pred, test_pred, target = "target", score = "score", g = 10, plot_show = TRUE, g_width = 12, file_name = NULL, save_data = FALSE, dir_path = tempdir(), gtitle = NULL)

ks_psi_plot(train_pred, test_pred, target = "target", score = "score", gtitle = NULL, plot_show = TRUE, g_width = 12, save_data = FALSE, breaks = NULL, g = 10, dir_path = tempdir())

model_key_index(tb_pred)

Arguments

train_pred   A data frame of training with predicted prob or score.

test_pred   A data frame of validation with predict prob or score.

target   The name of target variable.

score   The name of prob or score variable.

g   Number of breaks for prob or score.

breaks   Splitting points of prob or score.

pos_flag   The value of positive class of target variable, default: "1".

plot_show   Logical, show model performance in current graphic device. Default is FALSE.

g_width   Width of graphs.

file_name   The name for periodically saved data file. Default is NULL.

save_data   Logical, save results in locally specified folder. Default is FALSE.

dir_path   The path for periodically saved graphic files.

gtitle   The title of the graph & The name for periodically saved graphic file. Default is "_ks_psi_table".

tb_pred   A table generated by codeks_table
Examples

```r
sub = cv_split(UCICreditCard, k = 30)[[1]]
dat = UCICreditCard[sub,]
dat = re_name(dat, "default.payment.next.month", "target")
dat = data_cleansing(dat, target = "target", obs_id = "ID",
occurs_time = "apply_date", miss_values = list("", -1))

train_test <- train_test_split(dat, split_type = "OOT", prop = 0.7,
   occurs_time = "apply_date")
dat_train = train_test$train
dat_test = train_test$test
x_list = c("PAY_0", "LIMIT_BAL", "PAY_AMT5", "PAY_3", "PAY_2")
formula = as.formula(paste("target", paste(x_list, collapse = '+'), sep = '-'))
set.seed(46)
lr_model = glm(formula, data = dat_train[, c("target", x_list)], family = binomial(logit))

dat_train$pred_LR = round(predict(lr_model, dat_train[, x_list], type = "response"), 5)
dat_test$pred_LR = round(predict(lr_model, dat_test[, x_list], type = "response"), 5)

# model evaluation
ks_psi_plot(train_pred = dat_train, test_pred = dat_test,
score = "pred_LR", target = "target",
   plot_show = TRUE)
tb_pred <- ks_table_plot(train_pred = dat_train, test_pred = dat_test,
score = "pred_LR", target = "target",
g = 10, g_width = 13, plot_show = FALSE)
key_index = model_key_index(tb_pred)
```

ks_value

---

### Description

**ks_value** is for get K-S value for a prob or score.

### Usage

```r
ks_value(target, prob)
```

### Arguments

- `target` Vector of target.
- `prob` A list of predict probability or score.

### Value

KS value
Description

lasso_filter filter variables by lasso.

Usage

lasso_filter(dat_train, dat_test = NULL, target = NULL, x_list = NULL, pos_flag = NULL, ex_cols = NULL, sim_sign = "negtive", best_lambda = "lambda.auc", save_data = FALSE, plot.it = TRUE, seed = 46, file_name = NULL, dir_path = tempdir(), note = FALSE)

Arguments

dat_train  A data.frame with independent variables and target variable.
dat_test A data.frame of test data. Default is NULL.
target The name of target variable.
x_list Names of independent variables.
pos_flag The value of positive class of target variable, default: "1".
ex_cols A list of excluded variables. Regular expressions can also be used to match variable names. Default is NULL.
sim_sign The coefficients of all variables should be all negetive or positive, after turning to woe. Default is "negtive" for pos_flag is "1".
best_lambda Methods of best lambda standards using to filter variables by LASSO. There are 3 methods: ("lambda.auc", "lambda.ks", "lambda.sim_sign") . Default is "lambda.auc".
save_data Logical, save results in locally specified folder. Default is FALSE
plot.it Logical, shrinkage plot. Default is TRUE.
seed Random number seed. Default is 46.
file_name The name for periodically saved results files. Default is "Feature_selected_LASSO".
dir_path The path for periodically saved results files. Default is "/variable".
note Logical, outputs info. Default is FALSE.

Value

A list of filtered x variables by lasso.
Examples

```r
sub = cv_split(UCICreditCard, k = 40)[[1]]
dat = UCICreditCard[sub,]
dat = re_name(dat, "default.payment.next.month", "target")
dat_train = data_cleansing(dat, target = "target", obs_id = "ID", occur_time = "apply_date",
miss_values = list("", -1))
dat_train = process_nas(dat_train, default_miss = TRUE)
# get breaks of all predictive variables
x_list = c("PAY_0", "LIMIT_BAL", "PAY_AMT5", "EDUCATION", "PAY_3", "PAY_2")
breaks_list <- get_breaks_all(dat = dat_train, target = "target",
  x_list = x_list, occur_time = "apply_date", ex_cols = "ID",
save_data = FALSE, note = FALSE)
# woe transform
train_woe = woe_trans_all(dat = dat_train,
  target = "target",
  breaks_list = breaks_list,
  woe_name = FALSE)
lasso_filter(dat_train = train_woe,
  target = "target", x_list = x_list,
  save_data = FALSE, plot.it = FALSE)
```

---

**lendingclub**  
**Lending Club data**

**Description**

This data contains complete loan data for all loans issued through the time period stated, including the current loan status (Current, Late, Fully Paid, etc.) and latest payment information. The data containing loan data through the “present” contains complete loan data for all loans issued through the previous completed calendar quarter (time period: 2018Q1:2018Q4).

**Format**

A data frame with 63532 rows and 145 variables.

**Details**

- id: A unique LC assigned ID for the loan listing.
- issue_d: The month which the loan was funded.
- loan_status: Current status of the loan.
- addr_state: The state provided by the borrower in the loan application.
- acc_open_past_24mths: Number of trades opened in past 24 months.
- all_util: Balance to credit limit on all trades.
- annual_inc: The self-reported annual income provided by the borrower during registration.
- avg_cur_bal: Average current balance of all accounts.
- bc_open_to_buy: Total open to buy on revolving bankcards.
• bc_util: Ratio of total current balance to high credit/credit limit for all bankcard accounts.
• dti: A ratio calculated using the borrower's total monthly debt payments on the total debt obligations, excluding mortgage and the requested LC loan, divided by the borrower's self-reported monthly income.
• dti_joint: A ratio calculated using the co:borrowers' total monthly payments on the total debt obligations, excluding mortgages and the requested LC loan, divided by the co:borrowers' combined self:reported monthly income.
• emp_length: Employment length in years. Possible values are between 0 and 10 where 0 means less than one year and 10 means ten or more years.
• emp_title: The job title supplied by the Borrower when applying for the loan.
• funded_amnt_inv: The total amount committed by investors for that loan at that point in time.
• grade: LC assigned loan grade
• inq_last_12m: Number of credit inquiries in past 12 months
• installment: The monthly payment owed by the borrower if the loan originates.
• max_bal_bc: Maximum current balance owed on all revolving accounts
• mo_sin_old_il_acct: Months since oldest bank installment account opened
• mo_sin_old_rev_tl_op: Months since oldest revolving account opened
• mo_sin_rcnt_rev_tl_op: Months since most recent revolving account opened
• mo_sin_rcnt_tl: Months since most recent account opened
• mort_acc: Number of mortgage accounts.
• pct_tl_nvr_dlq: Percent of trades never delinquent
• percent_bc_gt_75: Percentage of all bankcard accounts > 75
• purpose: A category provided by the borrower for the loan request.
• sub_grade: LC assigned loan subgrade
• term: The number of payments on the loan. Values are in months and can be either 36 or 60.
• tot_cur_bal: Total current balance of all accounts
• tot_hi_crd_limit: Total high credit/credit limit
• total_acc: The total number of credit lines currently in the borrower’s credit file
• total_bal_ex_mort: Total credit balance excluding mortgage
• total_bc_limit: Total bankcard high credit/credit limit
• total_cu_tl: Number of finance trades
• total_il_high_credit_limit: Total installment high credit/credit limit
• verification_status_joint: Indicates if the co:borrowers’ joint income was verified by LC, not verified, or if the income source was verified
• zip_code: The first 3 numbers of the zip code provided by the borrower in the loan application.

See Also
UCICreditCard
lift_value

**Description**

*lift_value* is for getting max lift value for a prob or score.

**Usage**

```r
lift_value(target, prob)
```

**Arguments**

- `target` Vector of target.
- `prob` A list of predict probability or score.

**Value**

Max lift value

---

local_outlier_factor

**Description**

*local_outlier_factor* is function for calculating the lof factor for a data set using knn. This function is not intended to be used by end user.

**Usage**

```r
local_outlier_factor(dat, k = 10)
```

**Arguments**

- `dat` A data.frame contained only predict variables.
- `k` Number of neighbors for LOF. Default is 10.
log_trans

Description

log_trans is for logarithmic transformation.

Usage

log_trans(dat, target, x_list = NULL, cor_dif = 0.01, ex_cols = NULL, note = TRUE)

log_vars(dat, x_list = NULL, target = NULL, cor_dif = 0.01, ex_cols = NULL)

Arguments

dat A data.frame.
target The name of target variable.
x_list A list of x variables.
cor_dif The correlation coefficient difference with the target of logarithm transformed variable and original variable.
ex_cols Names of excluded variables. Regular expressions can also be used to match variable names. Default is NULL.
note Logical, outputs info. Default is TRUE.

Value

Log transformed data.frame.

Examples

dat = log_trans(dat = UCICreditCard, target = "default.payment.next.month", x_list = NULL, cor_dif = 0.01, ex_cols = "ID", note = TRUE)

loop_function

Loop Function. #’ loop_function is an iterator to loop through

Description

Loop Function. #’ loop_function is an iterator to loop through
Usage

```r
loop_function(func = NULL, args = list(data = NULL), x_list = NULL,
    bind = "rbind", parallel = TRUE, as_list = FALSE)
```

Arguments

- `func`: A function.
- `args`: A list of arguments required by function.
- `x_list`: Names of objects to loop through.
- `bind`: Compile results, "rbind" & "cbind" are available.
- `parallel`: Logical, parallel computing.
- `as_list`: Logical, whether outputs to be a list.

Value

A data.frame or list

Examples

```r
dat = UCICreditCard[24:26]
num_x_list = get_names(dat = dat, types = c("numeric", "integer", "double"),
    ex_cols = NULL, get_ex = FALSE)
dat[,num_x_list] = loop_function(func = outliers_kmeans_lof, x_list = num_x_list,
    args = list(dat = dat),
    bind = "cbind", as_list = FALSE, parallel = FALSE)
```

**Description**

`love_color` is for get plots for a variable.

Usage

```r
love_color(color = NULL, type = NULL, all = FALSE)
```

Arguments

- `color`: The name of colors.
- `type`: The type of colors, "deep".
- `all`: all of colors.

Examples

```r
love_color(color="dark_cyan")
```
low_variance_filter  Filtering Low Variance Variables

Description

low_variance_filter is for removing variables with repeated values up to a certain percentage.

Usage

low_variance_filter(dat, lvp = 0.97, note = FALSE, ex_cols = NULL)

Arguments

dat A data frame with x and target.
lvp The maximum percent of unique values (including NAs).
note Logical. Outputs info. Default is TRUE.
ex_cols A list of excluded variables. Default is NULL.

Value

A data.frame

Examples

dat = low_variance_filter(lendingclub[1:1000, ], lvp = 0.9)

lr_params Logistic Regression & Scorecard Parameters

Description

lr_params is the list of parameters to train a LR model or Scorecard using in training_model. lr_params_search is for searching the optimal parameters of logistic regression, if any parameters of params in lr_params is more than one.

Usage

lr_params(tree_control = list(p = 0.02, cp = 1e-08, xval = 5, maxdepth = 10), bins_control = list(bins_num = 10, bins_pct = 0.05, b_chi = 0.02, b_odds = 0.1, b_psi = 0.03, b_or = 0.15, mono = 0.2, odds_psi = 0.15, kc = 1), f_eval = "ks", best_lambda = "lambda.ks", method = "random_search", iters = 10, lasso = TRUE, step_wise = TRUE, score_card = TRUE, sp_values = NULL, forced_in = NULL, obsweight = c(1, 1), thresholds = list(cor_p =

---

---

---
\[0.8, \text{iv}_i = 0.02, \psi_i = 0.1, \cos_i = 0.5), \ldots\]

```
lr_params_search(method = "random_search", dat_train, target,
dat_test = NULL, occur_time = NULL, x_list = NULL, prop = 0.7,
iters = 10, tree_control = list(p = 0.02, cp = 0, xval = 1, maxdepth
= 10), bins_control = list(bins_num = 10, bins_pct = 0.02, b_chi =
0.02, b_odds = 0.1, b_psi = 0.05, b_or = 0.1, mono = 0.1, odds_psi =
0.03, kc = 1), thresholds = list(cor_p = 0.8, iv_i = 0.02, psi_i = 0.1,
\cos_i = 0.6), step_wise = FALSE, lasso = FALSE, f_eval = "ks")
```

**Arguments**

- `tree_control` the list of parameters to control cutting initial breaks by decision tree. See details at: `get_tree_breaks`
- `bins_control` the list of parameters to control merging initial breaks. See details at: `select_best_breaks`, `select_best_class`
- `f_eval` Customized evaluation function, "ks" & "auc" are available.
- `best_lambda` Methods of best lambda standards using to filter variables by LASSO. There are 3 methods: ("lambda.auc", "lambda.ks", "lambda.sim_sign") . Default is "lambda.auc".
- `method` Method of searching optimal parameters. "random_search", "grid_search", "local_search" are available.
- `iters` Number of iterations of "random_search" optimal parameters.
- `lasso` Logical, if TRUE, variables filtering by LASSO. Default is TRUE.
- `step_wise` Logical, stepwise method. Default is TRUE.
- `score_card` Logical, transfer woe to a standard scorecard. If TRUE, Output scorecard, and score prediction, otherwise output probability. Default is TRUE.
- `sp_values` Values will be in separate bins.e.g. list(-1, "missing") means that -1 & missing as special values. Default is NULL.
- `forced_in` Names of forced input variables. Default is NULL.
- `obsweight` An optional vector of 'prior weights' to be used in the fitting process. Should be NULL or a numeric vector. If you oversample or cluster different datasets to training the LR model, you need to set this parameter to ensure that the probability of logistic regression output is the same as that before oversampling or segmentation. e.g.: There are 10,000 0 obs and 500 1 obs before oversampling or under-sampling, 5,000 0 obs and 3,000 1 obs after oversampling. Then this parameter should be set to c(10000/5000, 500/3000). Default is NULL.
- `thresholds` Thresholds for selecting variables.
  - `cor_p` The maximum threshold of correlation. Default: 0.8.
  - `iv_i` The minimum threshold of IV. 0.01 to 0.1 usually work. Default: 0.02
  - `psi_i` The maximum threshold of PSI. 0.1 to 0.3 usually work. Default: 0.1.
  - `cos_i` cos_similarity of positive rate of train and test. 0.7 to 0.9 usually work. Default: 0.5.
- `...` Other parameters
**Description**

`lr_vif` is for calculating Variance-Inflation Factors.

**Usage**

`lr_vif(lr_model)`

**Arguments**

- `lr_model` An object of logistic model.

**Examples**

```r
sub = cv_split(UCICreditCard, k = 30)[[1]]
x_list = c("PAY_0", "LIMIT_BAL", "PAY_AMT5", "PAY_3", "PAY_2")
dat = re_name(UCICreditCard[sub], "default.payment.next.month", "target")
dat = dat[,c("target",x_list)]
dat = data_cleansing(dat, miss_values = list("", -1))

train_test <- train_test_split(dat, prop = 0.7)
dat_train = train_test$train
dat_test = train_test$test

Formula = as.formula(paste("target", paste(x_list, collapse = ' + '), sep = ' - '))
set.seed(46)
lr_model = glm(Formula, data = dat_train[, c("target", x_list)], family = binomial(logit))
```
merge_category

```r
lr_vif(lr_model)
get_logistic_coef(lr_model)
class(dat)
mod = lr_model
lr_vif(lr_model)
```

---

### max_min_norm

**Max Min Normalization**

**Description**

`max_min_norm` is for normalizing each column vector of matrix 'x' using max_min normalization.

**Usage**

```r
max_min_norm(x)
```

**Arguments**

- `x` Vector

**Value**

Normalized vector

**Examples**

```r
dat_s = apply(UCICreditCard[,12:14], 2, max_min_norm)
```

---

### merge_category

**Merge Category**

**Description**

`merge_category` is for merging category of nominal variables which number of categories is more than m or percent of samples in any categories is less than p.

**Usage**

```r
merge_category(dat, char_list = NULL, ex_cols = NULL, m = 30, note = TRUE)
```
Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dat</td>
<td>A data frame with x and target.</td>
</tr>
<tr>
<td>char_list</td>
<td>The list of characteristic variables that need to merge categories. Default is NULL. In case of NULL, merge categories for all variables of string type.</td>
</tr>
<tr>
<td>ex_cols</td>
<td>A list of excluded variables. Default is NULL.</td>
</tr>
<tr>
<td>m</td>
<td>The minimum number of categories.</td>
</tr>
<tr>
<td>note</td>
<td>Logical, outputs info. Default is TRUE.</td>
</tr>
</tbody>
</table>

Value

A data frame with merged category variables.

Examples

```
#merge category
dat = merge_category(lendingclub, ex_cols = "id\$\_d\$")
char_list = get_names(dat = dat, types = c("factor", "character"),
ex_cols = "id\$\_d\$", get_ex = FALSE)
str(dat[,char_list])
```

---

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>min_max_norm</td>
<td>Min Max Normalization</td>
</tr>
</tbody>
</table>

Description

`min_max_norm` is for normalizing each column vector of matrix 'x' using min_max normalization.

Usage

```
min_max_norm(x)
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>Vector</td>
</tr>
</tbody>
</table>

Value

Normalized vector

Examples

```
dat_s = apply(UCICreditCard[,12:14], 2, min_max_norm)
```
model_result_plot is a wrapper of following:

perf_table is for generating a model performance table.
ks_plot is for K-S.
roc_plot is for ROC.
lift_plot is for Lift Chart.
score_distribution_plot is for plotting the score distribution.

Description

model_result_plot is a wrapper of following: perf_table is for generating a model performance table. ks_plot is for K-S. roc_plot is for ROC. lift_plot is for Lift Chart. score_distribution_plot is for plotting the score distribution.

Usage

model_result_plot(train_pred, score, target, test_pred = NULL,
gtitle = NULL, perf_dir_path = NULL, save_data = FALSE,
plot_show = TRUE, total = TRUE, g = 10, cut_bin = "equal_depth")

perf_table(train_pred, test_pred = NULL, target = NULL, score = NULL,
g = 10, cut_bin = "equal_depth", breaks = NULL,
pos_flag = list("1", "1", "Bad", 1), total = FALSE)

ks_plot(train_pred, test_pred = NULL, target = NULL, score = NULL,
gtitle = NULL, breaks = NULL, g = 10, cut_bin = "equal_depth",
perf_tb = NULL)

lift_plot(train_pred, test_pred = NULL, target = NULL, score = NULL,
gtitle = NULL, breaks = NULL, g = 10, cut_bin = "equal_depth",
perf_tb = NULL)

roc_plot(train_pred, test_pred = NULL, target = NULL, score = NULL,
gtitle = NULL)

score_distribution_plot(train_pred, test_pred, target, score,
gtitle = NULL, breaks = NULL, g = 10, cut_bin = "equal_depth",
perf_tb = NULL)

Arguments

train_pred A data frame of training with predicted prob or score.
score          The name of prob or score variable.
target         The name of target variable.
test_pred       A data frame of validation with predict prob or score.
gtitle          The title of the graph & The name for periodically saved graphic file.
perf_dir_path   The path for periodically saved graphic files.
save_data      Logical, save results in locally specified folder. Default is FALSE.
pplot_show      Logical, show model performance in current graphic device. Default is TRUE.
total          Whether to summarize the table. default: TRUE.
g              Number of breaks for prob or score.
cut.bin        A string, if equal_bins is TRUE, 'equal_depth' or 'equal_width', default is 'equal_depth'.
b breaks        Splitting points of prob or score.
pos_flag       The value of positive class of target variable, default: "1".
perf_tb        Performance table.

Examples

```r
sub = cv_split(UCICreditCard, k = 30)[1]
dat = UCICreditCard[sub,]
dat = re_name(dat, "default.payment.next.month", "target")
x_list = c("PAY_0", "LIMIT_BAL", "PAY_AMT5", "PAY_3", "PAY_2")
dat = data_cleansing(dat, target = "target", obs_id = "ID", x_list = x_list,
   occur_time = "apply_date", miss_values = list("", -1))
dat = process_nas(dat, default_miss = TRUE)
train_test <- train_test_split(dat, split_type = "OOT", prop = 0.7,
   occur_time = "apply_date")

dat_train = train_test$train
dat_test = train_test$test
Formula = as.formula(paste("target", paste(x_list, collapse = ' + '), sep = ' - '))
set.seed(46)

# Fit the model
lr_model = glm(Formula, data = dat_train[, c("target", x_list)], family = binomial(logit))

dat_train$pred_LR = round(predict(lr_model, dat_train[, x_list], type = "response"), 5)
dat_test$pred_LR = round(predict(lr_model, dat_test[, x_list], type = "response"), 5)

# model evaluation
perf_table(train_pred = dat_train, test_pred = dat_test, target = "target", score = "pred_LR")
ks_plot(train_pred = dat_train, test_pred = dat_test, target = "target", score = "pred_LR")
roc_plot(train_pred = dat_train, test_pred = dat_test, target = "target", score = "pred_LR")
#lift_plot(train_pred = dat_train, test_pred = dat_test, target = "target", score = "pred_LR")
#score_distribution_plot(train_pred = dat_train, test_pred = dat_test,
#target = "target", score = "pred_LR")
#model_result_plot(train_pred = dat_train, test_pred = dat_test,
#target = "target", score = "pred_LR")
```
multi_grid

Arrangement of multiple plots into a grid

Description

Plot multiple ggplot-objects as a grid-arranged single plot.

Usage

\texttt{multi\_grid(..., grobs = list(...), nrow = NULL, ncol = NULL)}

Arguments

\ldots{} Other parameters.

grobs A list of ggplot-objects to be arranged into the grid.

nrow Number of rows in the plot grid.

ncol Number of columns in the plot grid.

Details

This function takes a list of ggplot-objects as argument. Plotting functions of this package that produce multiple plot objects (e.g., when there is an argument \texttt{facet.grid}) usually return multiple plots as list.

Value

An object of class \texttt{gtable}.

Examples

\begin{verbatim}
library(ggplot2)
sub = cv_split(UCICreditCard, k = 30)[[1]]
dat = UCICreditCard[sub,]
dat = re_name(dat, "default.payment.next.month", "target")
dat = data_cleansing(dat, target = "target", obs_id = "ID",
occurs_time = "apply_date", miss_values = list("", -1))
dat = process_nas(dat, default_miss = TRUE)
train_test <- train_test_split(dat, split_type = "OOT", prop = 0.7,
occur_time = "apply_date")
dat_train = train_test$train
dat_test = train_test$test
x_list = c("PAY_0", "LIMIT_BAL", "PAY_AMT5", "PAY_3", "PAY_2")
Formula = as.formula(paste("target", paste(x_list, collapse = ' + '), sep = ' - '))
set.seed(46)
lr_model = glm(Formula, data = dat_train[, c("target", x_list)], family = binomial(logit))

dat_train$pred_LR = round(predict(lr_model, dat_train[, x_list], type = "response"), 5)
dat_test$pred_LR = round(predict(lr_model, dat_test[, x_list], type = "response"), 5)
\end{verbatim}
# model evaluation
p1 = ks_plot(train_pred = dat_train, test_pred = dat_test, target = "target", score = "pred_LR")
p2 = roc_plot(train_pred = dat_train, test_pred = dat_test, target = "target", score = "pred_LR")
p3 = lift_plot(train_pred = dat_train, test_pred = dat_test, target = "target", score = "pred_LR")
p4 = score_distribution_plot(train_pred = dat_train, test_pred = dat_test, target = "target", score = "pred_LR")
p_plots = multi_grid(p1, p2, p3, p4)
plot(p_plots)

---

### multi_left_join

**Description**

multi_left_join is for left join a list of datasets fast.

**Usage**

multi_left_join(..., df_list = list(...), key_dt = NULL, by = NULL)

**Arguments**

- `...`: Datasets need join
- `df_list`: A list of datasets.
- `key_dt`: Name or index of Key table to left join.
- `by`: Name of Key columns to join.

**Examples**

multi_left_join(UCICreditCard[1:10, 1:10], UCICreditCard[1:10, c(1,8:14)], UCICreditCard[1:10, c(1,20:25)], by = "ID")

---

### null_blank_na

**Encode NAs**

**Description**

null_blank_na is the function to replace null, NULL, blank or other missing values with NA.

**Usage**

null_blank_na(dat, miss_values = NULL, note = FALSE)
Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dat</td>
<td>A data frame with x and target.</td>
</tr>
<tr>
<td>miss_values</td>
<td>Other extreme value might be used to represent missing values, e.g: -9999, -9998. These miss_values will be encoded to -1 or “missing”.</td>
</tr>
<tr>
<td>note</td>
<td>Logical. Outputs info. Default is TRUE.</td>
</tr>
</tbody>
</table>

Value

A data.frame

Examples

datss = null_blank_na(dat = UCICreditCard[1:1000, ], miss_values = list(-1,-2))

---

one_hot_encoding  One-Hot Encoding

Description

one_hot_encoding is for converting the factor or character variables into multiple columns

Usage

one_hot_encoding(dat, cat_vars = NULL, ex_cols = NULL, merge_cat = TRUE, na_act = TRUE, note = FALSE)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dat</td>
<td>A data frame.</td>
</tr>
<tr>
<td>cat_vars</td>
<td>The name or Column index list to be one_hot encoded.</td>
</tr>
<tr>
<td>ex_cols</td>
<td>Variables to be excluded, use regular expression matching</td>
</tr>
<tr>
<td>merge_cat</td>
<td>Logical. If TRUE, to merge categories greater than 8, default is TRUE.</td>
</tr>
<tr>
<td>na_act</td>
<td>Logical. If true, the missing value is processed, if FALSE missing value is omitted.</td>
</tr>
<tr>
<td>note</td>
<td>Logical. Outputs info. Default is TRUE.</td>
</tr>
</tbody>
</table>

Value

A data frame with the one hot encoding applied to all the variables with type as factor or character.

See Also

de_one_hot_encoding
Examples

```r
dat1 = one_hot_encoding(dat = UCICreditCard, 
cat_vars = c("SEX", "MARRIAGE"), 
merge_cat = TRUE, na_act = TRUE)
dat2 = de_one_hot_encoding(dat_one_hot = dat1, 
cat_vars = c("SEX","MARRIAGE"), na_act = FALSE)
```

Description

Outliers Detection `outliers_detection` is for outliers detecting using Kmeans and Local Outlier Factor (lof)

Usage

```r
outliers_detection(dat, x, kc = 3, kn = 5)
```

Arguments

- `dat`: A data.frame with independent variables.
- `x`: The name of variable to process.
- `kc`: Number of clustering centers for Kmeans.
- `kn`: Number of neighbors for LOF.

Value

Outliers of each variable.

Description

`outliers_detection` is for outliers detecting using Kmeans and Local Outlier Factor (lof).

Usage

```r
outliers_detection(dat, x, kc = 3, kn = 5)
```

Arguments

- `dat`: A data.frame with independent variables.
- `x`: The name of variable to process.
- `kc`: Number of clustering centers for Kmeans.
- `kn`: Number of neighbors for LOF.

Value

Outliers of each variable.

Description

`partial_dependence_plot` is for generating a partial dependence plot. `get_partial_dependence_plots` is for ploting partial dependence of all variables in `x_list`. 

```r
def one_hot_encoding(dat = UCICreditCard, 
cat_vars = c("SEX", "MARRIAGE"), 
merge_cat = TRUE, na_act = TRUE)
def de_one_hot_encoding(dat_one_hot = dat1, 
cat_vars = c("SEX","MARRIAGE"), na_act = FALSE)
```
Usage

partial_dependence_plot(model, x, x_train, n.trees = NULL)

get_partial_dependence_plots(model, x_train, x_list, n.trees = NULL,
   dir_path = getwd(), save_data = TRUE, plot_show = FALSE,
   parallel = FALSE)

Arguments

model  A data frame of training with predicted prob or score.
x  The name of an independent variable.
x_train  A data frame with independent variables.
n.trees  Number of trees for best.iter of gbm.
x_list  Names of independent variables.
dir_path  The path for periodically saved graphic files.
save_data  Logical, save results in locally specified folder. Default is FALSE.
plot_show  Logical, show model performance in current graphic device. Default is FALSE.
parallel  Logical, parallel computing. Default is FALSE.

Examples

sub = cv_split(UCICreditCard, k = 30)[[1]]
dat = UCICreditCard[sub,]
dat = re_name(dat, "default.payment.next.month", "target")
dat = data_cleansing(dat, target = "target", obs_id = "ID",
   occur_time = "apply_date", miss_values = list("", -1))

train_test <- train_test_split(dat, split_type = "OOT", prop = 0.7,
   occur_time = "apply_date")
dat_train = train_test$train
dat_test = train_test$test
x_list = c("PAY_0", "LIMIT_BAL", "PAY_AMT5", "PAY_3", "PAY_2")
Formula = as.formula(paste("target", paste(x_list, collapse = ' + '), sep = ' - '))
set.seed(46)
lr_model = glm(Formula, data = dat_train[, c("target", x_list)], family = binomial(logit))
#plot partial dependency of one variable
partial_dependence_plot(model = lr_model, x ="LIMIT_BAL", x_train = dat_train)
#plot partial dependency of all variables
pd_list = get_partial_dependence_plots(model = lr_model, x_list = x_list[1:2],
   x_train = dat_train, save_data = FALSE,plot_show = TRUE)
PCA_reduce

Description

PCA_reduce is used for PCA reduction of high dimension data.

Usage

PCA_reduce(train = train, test = NULL, mc = 0.9)

Arguments

- train: A data.frame with independent variables and target variable.
- test: A data.frame of test data.
- mc: Threshold of cumulative imp.

Examples

```r
## Not run:
num_x_list = get_names(dat = UCICreditCard, types = c('numeric'),
ex_cols = "ID$|date$|default.payment.next.month$", get_ex = FALSE)
PCA_dat = PCA_reduce(train = UCICreditCard[num_x_list])
## End(Not run)
```

plot_bar

Description

You can use the plot_bar to produce the barplot.

Usage

plot_bar(dat, x, breaks = NULL, g = 10, cut_bin = "equal_width")

Arguments

- dat: A data.frame with independent variables and target variable.
- x: The name of an independent variable.
- breaks: Breaks points of x.
- g: Number of initial breakpoints for equal frequency binning.
- cut_bin: 'equal_width' or 'equal_depth'
plot_density

Examples

plot_bar(dat = lendingclub, x = "grade")

plot_box

Plot Box

Description

You can use the plot_box to produce boxplot.

Usage

plot_box(dat, y, x = NULL, g = 5, colors_x = c(love_color(type = "deep"), love_color(type = "light"), love_color(type = "shallow"), love_color(type = "dark")))

Arguments

dat  A data.frame with independent variables and target variable.
y  The name of target variable.
x  The name of an independent variable.
g  Number of initial breakpoints for equal frequency binning.
colors_x  colors of x.

Examples

plot_box(lendingclub, x = "grade", y = "installment", g = 7)

plot_density

Plot Density

Description

You can use the plot_density to produce plots that characterize the density.

Usage

plot_density(dat, x, y = NULL, m = 3, g = 5, y_breaks = NULL, binwidth = NULL, hist = TRUE, colors_y = c(love_color(type = "deep"), love_color(type = "light"), love_color(type = "shallow")))
**plot_distribution**

**Arguments**

- `dat`: A data.frame with independent variables and target variable.
- `x`: The name of an independent variable.
- `y`: The name of target variable.
- `m`: The outlier cutoff.
- `g`: Number of initial breakpoints for equal frequency binning.
- `y_breaks`: Breaks points of y.
- `binwidth`: Width of bins for histogram.
- `hist`: If plot the histogram.
- `colors_y`: Colors of y.

**Examples**

```r
plot_density(dat = lendingclub, x = "annual_inc", y = "emp_length", m = 0, hist = FALSE)
plot_density(dat = lendingclub, x = "annual_inc", m = 2,
            colors_y = love_color(type = "line")[c(1,3)])
```

**Description**

You can use the `plot_distribution_x` to produce the distribution plot of a variable. You can use the `plot_distribution` to produce the plots of distributions of all variables.

**Usage**

```r
plot_distribution(dat, x_list = NULL, dir_path = tempdir(),
                  breaks_list = NULL, g = 10, m = 3, cut_bin = "equal_width")
plot_distribution_x(dat, x, breaks = NULL, g = 10, m = 3,
                    cut_bin = "equal_width", binwidth = NULL)
```

**Arguments**

- `dat`: A data.frame with independent variables and target variable.
- `x_list`: Names of independent variables.
- `dir_path`: The path for periodically saved graphic files.
- `breaks_list`: A table containing a list of splitting points for each independent variable. Default is NULL.
- `g`: Number of initial breakpoints for equal frequency binning.
- `m`: The outlier cutoff.
cut_bin  A string, if equal_bins is TRUE, 'equal_depth' or 'equal_width', default is 'equal_depth'.
x  The name of an independent variable.
breaks  Splitting points for an independent variable. Default is NULL.
binwidth  Width of bins for histogram.

Examples

plot_distribution_x(dat = lendingclub, x = "max_bal_bc", g = 10,
cut_bin = 'equal_width')
plot_distribution(dat = lendingclub, x_list = c("max_bal_bc", "installment"),
g = 10, dir_path = tempdir(),
cut_bin = 'equal_width')

plot_oot_perf  plot_oot_perf is for plotting performance of cross time samples in the future

Description

plot_oot_perf is for plotting performance of cross time samples in the future

Usage

plot_oot_perf(dat_test, x, occur_time, target, k = 3, g = 10,
period = "month", best = FALSE, equal_bins = TRUE, pl = "rate",
breaks = NULL, cut_bin = "equal_depth", gtitle = NULL,
perf_dir_path = NULL, save_data = FALSE, plot_show = TRUE)

Arguments

dat_test  A data frame of testing dataset with predicted prob or score.
x  The name of prob or score variable.
occur_time  The name of the variable that represents the time at which each observation takes place.
target  The name of target variable.
k  If period is NULL, number of equal frequency samples.
g  Number of breaks for prob or score.
period  OOT period, 'weekly' and 'month' are available. If NULL, use k equal frequency samples.
best  Logical, merge initial breaks to get optimal breaks for binning.
equal_bins  Logical, generates initial breaks for equal frequency or width binning.
pl  'lift' is for lift chart plot, 'rate' is for positive rate plot.
breaks  Splitting points of prob or score.
plot_relative_freq_histogram

Plot Relative Frequency Histogram

Description
You can use the plot_relative_freq_histogram to produce the relative frequency histogram plots.

Usage
plot_relative_freq_histogram(dat, x, y = NULL, x_breaks = NULL, y_breaks = NULL, g = 10, cut_bin = "equal_width")

Arguments
- dat: A data.frame with independent variables and target variable.
- x: The name of an independent variable.
- y: The name of target variable. Default is NULL.
plot_table

x_breaks  Breaks points of x.
y_breaks  Breaks points of y.
g  Number of initial breakpoints for equal frequency binning.
cut_bin  'equal_width' or 'equal_depth' to produce the breaks points.

Examples

plot_relative_freq_histogram(dat = lendingclub, x = "grade", y = "dti_joint", g = 7,
cut_bin = 'equal_width')

plot_table

Description

plot_table is for table visualization.

Usage

plot_table(grid_table, theme = c("cyan", "grey", "green", "red", "blue", "purple"), title = NULL, title.size = 12, title.color = "black", title.face = "bold", title.position = "middle", subtitle = NULL, subtitle.size = 8, subtitle.color = "black", subtitle.face = "plain", subtitle.position = "middle", tile.color = "white", tile.size = 1, colname.size = 3, colname.color = "white", colname.face = "bold", colname.fill.color = love_color("dark_cyan"), text.size = 3, text.color = love_color("dark_grey"), text.face = "plain", text.fill.color = c("white", love_color("pale_grey")))

Arguments

grid_table  A data.frame or table
theme  The theme of color, "cyan", "grey", "green", "red", "blue", "purple" are available.
title  The title of table
title.size  The title size of plot.
title.color  The title color.
title.face  The title face, such as "plain", "bold".
title.position  The title position, such as "left", "middle", "right".
subtitle  The subtitle of table
subtitle.size  The subtitle size.
subtitle.color  The subtitle color.
subtitle.face  The subtitle face, such as "plain", "bold", default is "bold".
subtitle.position
The subtitle position, such as "left", "middle", "right", default is "middle".

tile.color
The color of table lines, default is 'white'.

tile.size
The size of table lines, default is 1.

colname.size
The size of colnames, default is 3.

colname.color
The color of colnames, default is 'white'.

colname.face
The face of colnames, default is 'bold'.

colname.fill.color
The fill color of colnames, default is love_color("dark_cyan").

text.size
The size of text, default is 3.

text.color
The color of text, default is love_color("dark_grey").

text.face
The face of text, default is 'plain'.

text.fill.color
The fill color of text, default is c('white', love_color("pale_grey")).

Examples

```r
iv_list = get_psi_iv_all(dat = UCICreditCard[1:1000, ],
    x_list = names(UCICreditCard)[3:5], equal_bins = TRUE,
    target = "default.payment.next.month", ex_cols = "ID|apply_date")
iv_dt = get_psi_iv(UCICreditCard, x = "PAY_3",
    target = "default.payment.next.month", bins_total = TRUE)
plot_table(iv_dt)
```

Description

`plot_theme` is a simper wrapper of theme for ggplot2.

Usage

```r
plot_theme(legend.position = "top", angle = 30, legend_size = 7,
    axis_size_y = 8, axis_size_x = 8, axis_title_size = 10,
    title_size = 11, title_vjust = 0, title_hjust = 0,
    linetype = "dotted", face = "bold")
```
pred_score

Arguments

- `legend.position` (see details at: codelegend.position)
- `angle` (see details at: codeaxis.text.x)
- `legend_size` (see details at: codelegend.text)
- `axis_size_y` (see details at: codeaxis.text.y)
- `axis_size_x` (see details at: codeaxis.text.x)
- `axis_title_size` (see details at: codeaxis.title.x)
- `title_size` (see details at: codeplot.title)
- `title_vjust` (see details at: codeplot.title)
- `title_hjust` (see details at: codeplot.title)
- `linetype` (see details at: codepanel.grid.major)
- `face` (see details at: codeaxis.title.x)

Details

- see details at: codetheme

Description

pred_score is for using logistic regression model model to predict new data.

Usage

```r
pred_score(model, dat, x_list = NULL, bins_table = NULL,
            obs_id = NULL, miss_values = list(-1, "-1", "NULL", "-1", "-9999",
                                           "-9996", "-9997", "-9995", "-9998",
                                           -9999, -9998, -9997, -9996, -9995),
            woe_name = TRUE)
```

Arguments

- `model` Logistic Regression Model generated by `training_model`.
- `dat` Dataframe of new data.
- `x_list` Into the model variables.
- `bins_table` a data.frame generated by `get_bins_table`.
- `obs_id` The name of ID of observations or key variable of data. Default is NULL.
- `miss_values` Special values.
- `woe_name` Logical. Whether woe variable’s name contains ’woe’. Default is TRUE.
pred_xgb

Description

pred_xgb is for using xgboost model to predict new data.

Usage

pred_xgb(xgb_model = NULL, dat, x_list = NULL, miss_values = NULL, model_name = NULL, model_path = getwd())

Arguments

xgb_model XGboost Model generated by training_model.
dat Dataframe of new data.
x_list Into the model variables.
miss_values missing values.
model_name Name of model
model_path dir_path of model.

Value

new prob.

See Also

training_model, lr_params, xgb_params, rf_params

training_model, pred_score
process_nas

Description

process_nas_var is for missing value analysis and treatment using knn imputation, central imputation and random imputation. process_nas is a simpler wrapper for process_nas_var.

Usage

```r
code
process_nas(dat, x_list = NULL, class_var = FALSE,
            miss_values = NULL, parallel = FALSE, ex_cols = NULL,
            method = "median", note = FALSE, save_data = FALSE,
            file_name = NULL, dir_path = tempdir(), ...)
```

```r
code
process_nas_var(dat = dat, x, nas_rate = NULL, mat_nas_shadow = NULL,
                dt_nas_random = NULL, missing_type = NULL, method = "median",
                note = FALSE, save_data = FALSE, file_name = NULL,
                dir_path = tempdir(), ...)
```

Arguments

dat A data.frame with independent variables.
x_list Names of independent variables.
class_var Logical, nas analysis of the nominal variables. Default is TRUE.
miss_values Other extreme value might be used to represent missing values, e.g:-1, -9999, -9998. These miss_values will be encoded to NA.
parallel Logical, parallel computing. Default is FALSE.
ex_cols A list of excluded variables. Regular expressions can also be used to match variable names. Default is NULL.
method The methods of imputation by knn. If "median", then Nas imputation with k neighbors median. If "avg_dist", the distance weighted average method is applied to determine the NAs imputation with k neighbors. If "default", assigning the missing values to -1 or "missing", otherwise ,processing the missing values according to the results of missing analysis.
note Logical, outputs info. Default is TRUE.
save_data Logical. If TRUE, save missing analysis to dir_path
file_name The file name for periodically saved missing analysis file. Default is NULL.
dir_path The path for periodically saved missing analysis file. Default is "/variable".
... Other parameters.
x The name of variable to process.
nas_rate A list contains nas rate of each variable.
mat_nas_shadow A shadow matrix of variables which contain nas.
dt_nas_random A data.frame with random nas imputation.
missing_type Type of missing, genereted by codeanalysis_nas
Value

A data frame with no NAs.

Examples

\[
dat_na = process_nas(dat = UCICreditCard[1:1000,],
parallel = FALSE, ex_cols = "ID$", method = "median")
\]

Description

`outliers_kmeans_lof` is for outliers detection and treatment using Kmeans and Local Outlier Factor (lof). `process_outliers` is a simpler wrapper for `outliers_kmeans_lof`.

Usage

\[
\begin{align*}
\text{process_outliers}(dat, \text{target}, \text{ex_cols} = \text{NULL}, \text{kc} = 3, \text{kn} = 5, \\
x\_list = \text{NULL}, \text{parallel} = \text{FALSE}, \text{note} = \text{FALSE}, \text{process} = \text{TRUE}, \\
\text{save\_data} = \text{FALSE}, \text{file\_name} = \text{NULL}, \text{dir\_path} = \text{tempdir}() )
\end{align*}
\]

\[
\begin{align*}
\text{outliers\_kmeans\_lof}(dat, x, \text{target} = \text{NULL}, \text{kc} = 3, \text{kn} = 5, \\
\text{note} = \text{FALSE}, \text{process} = \text{TRUE}, \text{save\_data} = \text{FALSE}, \\
\text{file\_name} = \text{NULL}, \text{dir\_path} = \text{tempdir}() )
\end{align*}
\]

Arguments

- `dat`: Dataset with independent variables and target variable.
- `target`: The name of target variable.
- `ex_cols`: A list of excluded variables. Regular expressions can also be used to match variable names. Default is `NULL`.
- `kc`: Number of clustering centers for Kmeans.
- `kn`: Number of neighbors for LOF.
- `x_list`: Names of independent variables.
- `parallel`: Logical, parallel computing.
- `note`: Logical, outputs info. Default is `TRUE`.
- `process`: Logical, process outliers, not just analysis.
- `save_data`: Logical. If `TRUE`, save outliers analysis file to the specified folder at `dir_path`.
- `file_name`: The file name for periodically saved outliers analysis file. Default is `NULL`.
- `dir_path`: The path for periodically saved outliers analysis file. Default is `"./variable"`.
- `x`: The name of variable to process.
psi_iv_filter

Value
A data frame with outliers process to all the variables.

Examples

dat_out = process_outliers(UCICreditCard[1:10000,c(18:21,26)],
target = "default.payment.next.month",
ex_cols = "date$", kc = 3, kn = 10,
parallel = FALSE,note = TRUE)

---

psi_iv_filter

Variable reduction based on Information Value & Population Stability

Index filter

Description

psi_iv_filter is for selecting important and stable features using IV & PSI.

Usage

psi_iv_filter(dat, dat_test = NULL, target, x_list = NULL,
breaks_list = NULL, pos_flag = NULL, ex_cols = NULL,
occur_time = NULL, best = FALSE, equal_bins = TRUE, g = 10,
sp_values = NULL, tree_control = list(p = 0.05, cp = 1e-06, xval = 5,
maxdepth = 10), bins_control = list(bins_num = 10, bins_pct = 0.05,
b_chi = 0.05, b_odds = 0.1, b_psi = 0.05, b_or = 0.15, mono = 0.3,
odds_psi = 0.2, kc = 1), oot_pct = 0.7, psi_i = 0.1, iv_i = 0.01,
cos_i = 0.7, vars_name = FALSE, note = TRUE, parallel = FALSE,
save_data = FALSE, file_name = NULL, dir_path = tempdir(), ...)

Arguments

dat
A data frame with independent variables and target variable.
dat_test
A data frame of test data. Default is NULL.
target
The name of target variable.
x_list
Names of independent variables.
breaks_list
A table containing a list of splitting points for each independent variable. Default is NULL.
pos_flag
The value of positive class of target variable, default: "1".
ex_cols
A list of excluded variables. Regular expressions can also be used to match variable names. Default is NULL.
occur_time
The name of the variable that represents the time at which each observation takes place.
best
Logical, if TRUE, merge initial breaks to get optimal breaks for binning.
psi_iv_filter

equal_bins Logical, if TRUE, equal sample size initial breaks generates. If FALSE, tree breaks generates using decision tree.
g Integer, number of initial bins for equal_bins.
sp_values A list of missing values.
tree_control the list of tree parameters.
bins_control the list of parameters.
oot_pct Percentage of observations retained for overtime test (especially to calculate PSI). Default is 0.7
psi_i The maximum threshold of PSI. 0 <= psi_i <= 1; 0.05 to 0.2 usually work. Default: 0.1
iv_i The minimum threshold of IV. 0 < iv_i; 0.01 to 0.1 usually work. Default: 0.01
cos_i cos_similarity of positive rate of train and test. 0.7 to 0.9 usually work. Default: 0.5.
vars_name Logical, output a list of filtered variables or table with detailed IV and PSI value of each variable. Default is FALSE.
note Logical, outputs info. Default is TRUE.
parallel Logical, parallel computing. Default is FALSE.
save_data Logical, save results in locally specified folder. Default is FALSE.
file_name The name for periodically saved results files. Default is "Feature_importance_IV_PSI".
dir_path The path for periodically saved results files. Default is tempdir().
... Other parameters.

Value

A list with the following elements:

- Feature Selected variables.
- IV IV of variables.
- PSI PSI of variables.
- COS cos_similarity of positive rate of train and test.

See Also

xgb_filter, gbm_filter, feature_selector

Examples

psi_iv_filter(dat = UCICreditCard[1:1000,c(2,4,8:9,26)],
               target = "default.payment.next.month",
               occur_time = "apply_date",
               parallel = FALSE)
**Description**

This function is not intended to be used by end user.

**Usage**

\[ p_{ij}(x) \]

\[ e_{ij}(x) \]

**Arguments**

\( x \)

A numeric vector.

**Value**

A numeric vector of entropy.

---

**Description**

\( p_{to\_score} \) is for transforming probability to score.

**Usage**

\[ p_{to\_score}(p, PDO = 20, base = 600, ratio = 1) \]

**Arguments**

\( p \)

Probability.

\( PDO \)

Point-to-Double Odds.

\( base \)

Base Point.

\( ratio \)

The corresponding odds when the score is base.

**Value**

new prob.

**See Also**

\[ training\_model, pred\_score \]
**quick_as_df**  

List as data.frame quickly

**Description**  

Quick_as_df is function for fast data frame transformation.

**Usage**  

quick_as_df(df_list)

**Arguments**  

df_list A list of data.

**Value**  

Packages installed and library.

**Examples**

UCICreditCard = quick_as_df(UCICreditCard)

---

**ranking_percent_proc**  

Ranking Percent Process

**Description**  

Ranking_percent_proc is for processing ranking percent variables. ranking_percent_dict is for generating ranking percent dictionary.

**Usage**  

ranking_percent_proc(dat, ex_cols = NULL, x_list = NULL, rank_dict = NULL, pct = 0.01, parallel = FALSE, note = FALSE, save_data = FALSE, file_name = NULL, dir_path = tempdir(), ...)

ranking_percent_proc_x(dat, x, rank_dict = NULL, pct = 0.01)

ranking_percent_dict(dat, x_list = NULL, ex_cols = NULL, pct = 0.01, parallel = FALSE, save_data = FALSE, file_name = NULL, dir_path = tempdir(), ...)

ranking_percent_dict_x(dat, x = NULL, pct = 0.01)
Arguments

dat          A data.frame.
ex_cols      Names of excluded variables. Regular expressions can also be used to match variable names. Default is NULL.
x_list       A list of x variables.
rank_dict    The dictionary of rank_percent generated by ranking_percent_dict.
pct          Percent of rank. Default is 0.01.
parallel     Logical, parallel computing. Default is FALSE.
note         Logical, outputs info. Default is TRUE.
save_data    Logical, save results in locally specified folder. Default is FALSE.
file_name    The name for periodically saved rank_percent data file. Default is "dat_rank_percent".
dir_path     The path for periodically saved rank_percent data file. Default is "tempdir()".
...          Additional parameters.
x            The name of an independent variable.

Value

Data.frame with new processed variables.

Examples

rank_dict = ranking_percent_dict(dat = UCI CreditCard[1:1000,],
x_list = c("LIMIT_BAL", "BILL_AMT2", "PAY_AMT3"), ex_cols = NULL )
UCICreditCard_new = ranking_percent_proc(dat = UCI CreditCard[1:1000,],
x_list = c("LIMIT_BAL", "BILL_AMT2", "PAY_AMT3"), rank_dict = rank_dict, parallel = FALSE)

read_data()  Read data

Description

read_data is for loading data, formats like csv, txt, data and so on.

Usage

read_data(path, pattern = NULL, encoding = "unknown", header = TRUE,
    sep = "auto", stringsAsFactors = FALSE, select = NULL,
    drop = NULL, nrows = Inf)

check_data_format(path)
reduce_high_cor_filter

Arguments

- **path**: Path to file or file name in working directory & path to file.
- **pattern**: An optional regular expression. Only file names which match the regular expression will be returned.
- **encoding**: Default is "unknown". Other possible options are "UTF-8" and "Latin-1".
- **header**: Does the first data line contain column names?
- **sep**: The separator between columns.
- **stringsAsFactors**: Logical. Convert all character columns to factors?
- **select**: A vector of column names or numbers to keep, drop the rest.
- **drop**: A vector of column names or numbers to drop, keep the rest.
- **nrows**: The maximum number of rows to read.

---

reduce_high_cor_filter

Filtering highly correlated variables with reduce method

Description

reduce_high_cor_filter is function for filtering highly correlated variables with reduce method.

Usage

reduce_high_cor_filter(dat, x_list = NULL, size = ncol(dat)/10, p = 0.95, com_list = NULL, ex_cols = NULL, cor_class = TRUE, parallel = FALSE)

Arguments

- **dat**: A data.frame with independent variables.
- **x_list**: Names of independent variables.
- **size**: Size of variable group.
- **p**: Threshold of correlation between features. Default is 0.7.
- **com_list**: A data.frame with important values of each variable. eg : IV_list
- **ex_cols**: A list of excluded variables. Regular expressions can also be used to match variable names. Default is NULL.
- **cor_class**: Culate catagory variables’s correlation matrix. Default is FALSE.
- **parallel**: Logical, parallel computing. Default is FALSE.
**remove_duplicated**  
*Remove Duplicated Observations*

**Description**

`remove_duplicated` is the function to remove duplicated observations.

**Usage**

```r
remove_duplicated(dat = dat, obs_id = NULL, occur_time = NULL, target = NULL, note = FALSE)
```

**Arguments**

- `dat`  
  A data frame with x and target.
- `obs_id`  
  The name of ID of observations. Default is NULL.
- `occur_time`  
  The name of occur time of observations. Default is NULL.
- `target`  
  The name of target variable.
- `note`  
  Logical. Outputs info. Default is TRUE.

**Value**

A data.frame

**Examples**

```r
datss = remove_duplicated(dat = UCICreditCard,  
target = "default.payment.next.month",  
obs_id = "ID", occur_time = "apply_date")
```

---

**require_packages**  
*Packages required and intallment*

**Description**

`require_packages` is function for librarying required packages and installing missing packages if needed.

**Usage**

```r
require_packages(..., pkg = as.character(substitute(list(...))))
```

**Arguments**

- `...`  
  Packages need loaded
- `pkg`  
  A list or vector of names of required packages.
**Value**

packages installed and library.

**Examples**

```r
## Not run:
require_packages(data.table, ggplot2, dplyr)
## End(Not run)
```

---

**re_code**

`re_code` search for matches to argument pattern within each element of a character vector:

**Description**

`re_code` search for matches to argument pattern within each element of a character vector:

**Usage**

```r
re_code(x, codes)
```

**Arguments**

- `x`: Variable to recode.
- `codes`: A data.frame of original value & recode value

**Examples**

```r
SEX = sample(c("F","M"),1000,replace = TRUE)
codes= data.frame(ori_value = c("F","M"), code = c(0,1) )
SEX_re = re_code(SEX,codes)
```

---

**re_name**

`re_name` for renaming variables.

**Description**

`re_name` is for renaming variables.

**Usage**

```r
re_name(dat, oldname = c(), newname = c())
```
Arguments

dat A data frame with variables to rename.
oldname Old names of variables.
newname New names of variables.

Value
data with new variable names.

Examples

dt = re_name(dat = UCICreditCard, "default.payment.next.month", "target")
names(dt['target'])

rf_params

Random Forest Parameters

Description

rf_params is the list of parameters to train a Random Forest using in training_model.

Usage

rf_params(ntree = 100, nodesize = 30, samp_rate = 0.5, tune_rf = FALSE, ...)

Arguments

ntree Number of trees to grow. This should not be set to too small a number, to ensure that every input row gets predicted at least a few times.
nodesize Minimum size of terminal nodes. Setting this number larger causes smaller trees to be grown (and thus take less time). Note that the default values are different for classification (1) and regression (5).
samp_rate Percentage of sample to draw. Default is 0.2.
tune_rf A logical. If TRUE, then tune Random Forest model. Default is FALSE.
... Other parameters

Details

See details at: https://www.stat.berkeley.edu/~breiman/Using_random_forests_V3.1.pdf

Value

A list of parameters.
See Also

`training_model, lr_params, gbm_params, xgb_params`

---

**Description**

Functions for vector operation.

**Usage**

```r
rowAny(x)
rowAllnas(x)
colAllnas(x)
colAllzeros(x)
rowAll(x)
rowCVs(x, na.rm = FALSE)
rowSds(x, na.rm = FALSE)
colSds(x, na.rm = TRUE)
rowMaxs(x, na.rm = FALSE)
rowMins(x, na.rm = FALSE)
rowMaxMins(x, na.rm = FALSE)
colMaxMins(x, na.rm = FALSE)
cnt_x(x)
sum_x(x)
max_x(x)
min_x(x)
avg_x(x)
```
Arguments

x A data.frame or Matrix.
na.rm Logical, remove NAs.

Value

A data.frame or Matrix.

Examples

#any row has missing values
row_amy = rowAny(UCICreditCard[8:10])
#rows which is all missing values
row_na = rowAllnas(UCICreditCard[8:10])
#cols which is all missing values
col_na = colAllnas(UCICreditCard[8:10])
#cols which is all zeros
row_zero = colAllzeros(UCICreditCard[8:10])
#sum all numbers of a row
row_all = rowAll(UCICreditCard[8:10])
#caculate cv of a row
row_cv = rowCVs(UCICreditCard[8:10])
#caculate sd of a row
row_sd = rowSds(UCICreditCard[8:10])
#caculate sd of a column
col_sd = colSds(UCICreditCard[8:10])

Description

rules_filter This function is used to filter or select samples by rules.

Usage

rules_filter(rules_list, dat, filtering = TRUE)

Arguments

rules_list A list of rules.
dat A data.frame.
filtering Logical, if TRUE, filtering samples, if FALSE, selecting samples.

Value

A data frame with tree rules and percent under each rule.
See Also

get_ctree_rules, check_rules

Examples

```r
train_test <- train_test_split(UCICreditCard, split_type = "Random", prop = 0.8, save_data = FALSE)
dat_train = train_test$train
dat_test = train_test$test
dat_train$default.payment.next.month = as.numeric(dat_train$default.payment.next.month)
rules_list = get_ctree_rules(tree_fit = NULL, train_dat = dat_train[, 8:26],
target = "default.payment.next.month", test_dat = dat_test)[1:3, 2]
new_dat = rules_filter(rules_list = rules_list[3], dat = dat_test)
```

Description

save_data is for saving a data.frame or a list fast.

Usage

```r
save_data(..., files = list(...),
file_name = as.character(substitute(list(...))), dir_path = getwd(),
note = FALSE, as_list = FALSE, row_names = FALSE, append = FALSE)
```

Arguments

... datasets
files A dataset or a list of datasets.
file_name The file name of data.
dir_path A string. The dir path to save breaks_list.
ote Logical. Outputs info. Default is TRUE.
as_list Logical. List format or data.frame format to save. Default is FALSE.
row_names Logical, retain rownames.
append Logical, append newdata to old.

Examples

```r
save_data(UCICreditCard,"UCICreditCard", tempdir())
```
**score_transfer**  

**Score Transformation**

**Description**

score_transfer is for transfer woe to score.

**Usage**

```r
score_transfer(model, tbl_woe, a = 600, b = 50, file_name = NULL,  
dir_path = tempdir(), save_data = FALSE)
```

**Arguments**

- `model` A data frame with x and target.
- `tbl_woe` A data frame with woe variables.
- `a` Base line of score.
- `b` Numeric. Increased scores from doubling Odds.
- `file_name` The name for periodically saved score file. Default is "dat_score".
- `dir_path` The path for periodically saved score file. Default is "/data".
- `save_data` Logical, save results in locally specified folder. Default is FALSE.

**Value**

A data frame with variables which values transferred to score.

**Examples**

```r
# dataset splitting  
sub = cv_split(UCICreditCard, k = 30)[[1]]  
dat = UCICreditCard[sub,]  
# rename the target variable  
dat = re_name(dat, "default.payment.next.month", "target")  
dat = data_cleansing(dat, target = "target", obs_id = "ID",  
occur_time = "apply_date", miss_values = list("", -1))  
# train_test splitting  
train_test <- train_test_split(dat, split_type = "OOT", prop = 0.7,  
occur_time = "apply_date")  
dat_train = train_test$train  
dat_test = train_test$test  
# get breaks of all predictive variables  
x_list = c("PAY_0", "LIMIT_BAL", "PAY_AMT5", "EDUCATION", "PAY_3", "PAY_2")  
breaks_list <- get_breaks_all(dat = dat_train, target = "target",  
x_list = x_list, occur_time = "apply_date", ex_cols = "ID",  
save_data = FALSE, note = FALSE)  
# woe transforming  
train_woe = woe_trans_all(dat = dat_train,  
```

```r
model, tbl_woe, a = 600, b = 50, file_name = NULL,  
dir_path = tempdir(), save_data = FALSE)
```
target = "target",
breaks_list = breaks_list,
woe_name = FALSE)
test_woe = woe_trans_all(dat = dat_test,
target = "target",
breaks_list = breaks_list,
note = FALSE)
Formula = as.formula(paste("target", paste(x_list, collapse = ' + '), sep = ' - '))
set.seed(46)
lr_model = glm(Formula, data = train_woe[, c("target", x_list)], family = binomial(logit))
# get LR coefficient
dt_imp_LR = get_logistic_coef(lg_model = lr_model, save_data = FALSE)
bins_table = get_bins_table_all(dat = dat_train, target = "target",
x_list = x_list, dat_test = dat_test,
breaks_list = breaks_list, note = FALSE)
# score card
LR_score_card <- get_score_card(lg_model = lr_model, bins_table, target = "target")
# scoring
train_pred = dat_train[, c("ID", "apply_date", "target")]
test_pred = dat_test[, c("ID", "apply_date", "target")]
train_pred$pred_LR = score_transfer(model = lr_model,
tbl_woe = train_woe,
save_data = FALSE)[, "score"]
test_pred$pred_LR = score_transfer(model = lr_model,
tbl_woe = test_woe, save_data = FALSE)[, "score"]

select_best_class

Generates Best Binning Breaks

Description

select_best_class & select_best_breaks are for merging initial breaks of variables using chi-square, odds-ratio, PSI, G/B index and so on. The get_breaks is a simpler wrapper for select_best_class & select_best_class.

Usage

select_best_class(dat, x, target, breaks = NULL, occur_time = NULL,
oot_pct = 0.7, pos_flag = NULL, bins_control = NULL,
sp_values = NULL, ...)

select_best_breaks(dat, x, target, breaks = NULL, pos_flag = NULL,
sp_values = NULL, occur_time = NULL, oot_pct = 0.7,
bins_control = NULL, ...)

Arguments

dat A data frame with x and target.
The name of variable to process.

The name of target variable.

Splitting points for an independent variable. Default is NULL.

The name of the variable that represents the time at which each observation takes place.

The percentage of Actual and Expected set for PSI calculating.

The value of positive class of target variable, default: "1".

the list of parameters.

- `bins_num` The maximum number of bins. 5 to 10 usually work. Default: 10
- `bins_pct` The minimum percent of observations in any bins. 0 < bins_pct < 1, 0.01 to 0.1 usually work. Default: 0.02.
- `b_chi` The minimum threshold of chi-square merge. 0 < b_ch< 1; 0.01 to 0.1 usually work. Default: 0.02.
- `b_odds` The minimum threshold of odds merge. 0 < b_odds < 1; 0.05 to 0.2 usually work. Default: 0.02.
- `b_psi` The maximum threshold of PSI in any bins. 0 < b_psi < 1; 0 to 0.1 usually work. Default: 0.05.
- `b_or` The maximum threshold of G/B index in any bins. 0 < b_or < 1; 0.05 to 0.3 usually work. Default: 0.15.
- `odds_psi` The maximum threshold of Training and Testing G/B index PSI in any bins. 0 < odds_psi < 1; 0.01 to 0.3 usually work. Default: 0.1.
- `mono` Monotonicity of all bins, the larger, the more nonmonotonic the bins will be. 0 < mono < 0.5; 0.2 to 0.4 usually work. Default: 0.2.
- `kc` number of cross-validations. 1 to 5 usually work. Default: 1.

A list of special value.

... Other parameters.

The following is the list of Reference Principles

- 1. The increasing or decreasing trend of variables is consistent with the actual business experience. (The percent of Non-monotonic intervals of which are not head or tail is less than 0.35)
- 2. Maximum 10 intervals for a single variable.
- 3. Each interval should cover more than 2
- 4. Each interval needs at least 30 or 1
- 5. Combining the values of blank, missing or other special value into the same interval called missing.
- 6. The difference of Chi effect size between intervals should be at least 0.02 or more.
- 7. The difference of absolute odds ratio between intervals should be at least 0.1 or more.
8. The difference of positive rate between intervals should be at least 1/10 of the total positive rate.

9. The difference of G/B index between intervals should be at least 15 or more.

10. The PSI of each interval should be less than 0.1.

Value

A list of breaks for x.

See Also

`get_tree_breaks, cut_equal, get_breaks`

Examples

```r
#equal sample size breaks
equ_breaks = cut_equal(dat = UCICreditCard[, "PAY_AMT2"], g = 10)

# select best bins
bins_control = list(bins_num = 10, bins_pct = 0.02, b_chi = 0.02,
b_odds = 0.1, b_psi = 0.05, b_or = 0.15, mono = 0.3, odds_psi = 0.1, kc = 1)
select_best_breaks(dat = UCICreditCard, x = "PAY_AMT2", breaks = equ_breaks,
target = "default.payment.next.month", occur_time = "apply_date",
sp_values = NULL, bins_control = bins_control)
```

Description

This function is not intended to be used by end user.

Usage

```r
sim_str(a, b, sep = "_[.]_[A-Z]")
```

Arguments

- **a**: A string
- **b**: A string
- **sep**: Seprater of strings. Default is "_[.]_[A-Z]".
**split_bins**

Description

`split_bins` is for binning using breaks.

Usage

```r
split_bins(dat, x, breaks = NULL, bins_no = TRUE)
```

Arguments

- `dat`: A data.frame with independent variables.
- `x`: The name of an independent variable.
- `breaks`: Breaks for binning.
- `bins_no`: Number the generated bins. Default is TRUE.

Value

A data.frame with Bined x.

Examples

```r
bins = split_bins(dat = UCICreditCard, x = "PAY_AMT1", breaks = NULL, bins_no = TRUE)
```

**split_bins_all**

Description

`split_bins_all` is for transforming data to bins. The `split_bins_all` function is a simpler wrapper for `split_bins`.

Usage

```r
split_bins_all(dat, x_list = NULL, ex_cols = NULL, breaks_list = NULL, bins_no = TRUE, note = FALSE, save_data = FALSE, file_name = NULL, dir_path = tempdir(), ...)
```

...
split_bins_all

Arguments

- `dat` A data.frame with independent variables.
- `x_list` A list of x variables.
- `ex_cols` Names of excluded variables. Regular expressions can also be used to match variable names. Default is NULL.
- `breaks_list` A list contains breaks of variables. it is generated by `get_breaks_all`.
- `bins_no` Number the generated bins. Default is TRUE.
- `note` Logical, outputs info. Default is TRUE.
- `save_data` Logical, save results in locally specified folder. Default is TRUE.
- `file_name` The name for periodically saved woe file. Default is "dat_woe".
- `dir_path` The path for periodically saved woe file. Default is "/data"
- `...` Additional parameters.

Value

A data.frame with splitted bins.

See Also

- `get_tree_breaks`
- `cut_equal`
- `select_best_class`
- `select_best_breaks`

Examples

```r
sub = cv_split(UCICreditCard, k = 30)[[1]]
dat = UCICreditCard[sub,]
dat = re_name(dat, "default.payment.next.month", "target")
dat = data_cleansing(dat, target = "target", obs_id = "ID", occur_time = "apply_date",
miss_values = list("", -1))

train_test <- train_test_split(dat, split_type = "OOT", prop = 0.7,
occur_time = "apply_date")
dat_train = train_test$train
dat_test = train_test$test
# get breaks of all predictive variables
x_list = c("PAY_0", "LIMIT_BAL", "PAY_AMT5", "EDUCATION", "PAY_3", "PAY_2")
breaks_list <- get_breaks_all(dat = dat_train, target = "target",
  x_list = x_list, occur_time = "apply_date", ex_cols = "ID",
save_data = FALSE, note = FALSE)

# woe transform
train_bins = split_bins_all(dat = dat_train,
  breaks_list = breaks_list,
  woe_name = FALSE)

test_bins = split_bins_all(dat = dat_test,
  breaks_list = breaks_list,
  note = FALSE)
```
### start_parallel_computing

Parallel computing and export variables to global Env.

**Description**

This function is not intended to be used by end user.

**Usage**

```r
start_parallel_computing(parallel = TRUE)
```

**Arguments**

- `parallel` A logical, default is TRUE.

**Value**

parallel works.

### stop_parallel_computing

Stop parallel computing

**Description**

This function is not intended to be used by end user.

**Usage**

```r
stop_parallel_computing(cluster)
```

**Arguments**

- `cluster` Parallel works.

**Value**

stop clusters.
**str_match**

`string match #`' str_match search for matches to argument pattern within each element of a character vector:

**Description**

string match `#` str_match search for matches to argument pattern within each element of a character vector:

**Usage**

```r
str_match(pattern, str_r)
```

**Arguments**

- `pattern`: character string containing a regular expression (or character string for fixed = TRUE) to be matched in the given character vector. Coerced by as.character to a character string if possible. If a character vector of length 2 or more is supplied, the first element is used with a warning. missing values are allowed except for regexpr and gregexpr.
- `str_r`: a character vector where matches are sought, or an object which can be coerced by as.character to a character vector. Long vectors are supported.

**Examples**

```r
original_nam = c("12mdd","11mdd","10mdd")
str_match(str_r = original_nam,pattern= "\d+"
```

---

**swap_analysis**

*Swap Out/Swap In Analysis*

**Description**

swap_analysis is for swap out/swap in analysis.

**Usage**

```r
swap_analysis(dat, new_rules, old_rules, target = NULL,
cross_type = "total_pct", value = NULL)
```
term_tfidf

Arguments

dat A data.frame with independent variables.
new_rules A list of new rules.
old_rules A list of old rules.
target The name of target variable.
cross_type Output form of the result of crosstable. Provide these four forms: "total_sum","total_pct","bad_sum","bad_pct".
value The name of the variable to sum. When this parameter is NULL, the default statistics is to sum frequency.

Value

A cross table.

Examples

swap_analysis(dat = UCICreditCard, new_rules = list("SEX == 'male' & AGE < 25"),
old_rules = list("SEX == 'male' & AGE < 30"),
target = "default.payment.next.month", cross_type = "bad_pct", value = "LIMIT_BAL")

term_tfidf

TF-IDF

Description

The term_filter is for filtering stop_words and low frequency words. The term_idf is for computing idf(inverse documents frequency) of terms. The term_tfidf is for computing tf-idf of documents.

Usage

term_tfidf(term_df, idf = NULL)

term_idf(term_df, n_total = NULL)

term_filter(term_df, low_freq = 0.01, stop_words = NULL)

Arguments

term_df A data.frame with id and term.
idf A data.frame with idf.
n_total Number of documents.
low_freq Use rate of terms or use numbers of terms.
stop_words Stop words.
time_series_proc 111

Value
A data.frame

Examples

term_df = data.frame(id = c(1,1,2,2,3,3,4,4,4,4,4,4,5,5,6,7,7,
                        8,8,8,9,9,10,10,11,11,11,11,11,11,11,11,11),
terms = c('a','b','c','a','c','d','d','a','b','c','a','c','d','a','c',
          'd','a','e','f','b','c','f','b','c','h','h','i','c','d','g','k','k'))
term_df = term_filter(term_df = term_df, low_freq = 1)
idf = term_idf(term_df)
tf_idf = term_tfidf(term_df,idf)

Description
This function is used for time series data processing.

Usage
time_series_proc(dat, ID = NULL, group = NULL, time = NULL)

Arguments
dat A data.frame contained only predict variables.
ID The name of ID of observations or key variable of data. Default is NULL.
group The group of behavioral or status variables.
time The name of variable which is time when behavior was happened.

Details
The key to creating a good model is not the power of a specific modelling technique, but the breadth and depth of derived variables that represent a higher level of knowledge about the phenomena under examination.

Examples
dat = data.frame(id = c(1,1,2,2,3,3,4,4,4,4,4,4,5,5,6,7,7,
                      8,8,8,9,9,10,10,11,11,11,11,11,11,11,11,11),
terms = c('a','b','c','a','c','d','d','a','b','c','a','c','d','a','c',
          'd','a','e','f','b','c','f','b','c','h','h','i','c','d','g','k','k'),
time = c(8,3,1,9,6,1,4,9,1,3,4,8,2,7,1,
          3,4,1,8,7,2,5,7,8,8,2,1,5,7,2,7,3))
time_series_proc(dat = dat, ID = 'id', group = 'terms', time = 'time')
**time_transfer**

**Time Format Transfering**

**Description**

`time_transfer` is for transferring time variables to time format.

**Usage**

\[
\text{time}\_\text{transfer}(\text{dat}, \text{date}\_\text{cols} = \text{NULL}, \text{ex}\_\text{cols} = \text{NULL}, \text{note} = \text{FALSE})
\]

**Arguments**

- `dat`: A data frame
- `date_cols`: Names of time variable or regular expressions for finding time variables. Default is "DATE$|time$|date$|timestamp$|stamp$".
- `ex_cols`: Names of excluded variables. Regular expressions can also be used to match variable names. Default is `NULL`.
- `note`: Logical, outputs info. Default is `TRUE`.

**Value**

A data.frame with transferred time variables.

**Examples**

# transfer a variable.
\[
\text{dat} = \text{time}\_\text{transfer}(\text{dat} = \text{lendingclub}, \text{date}\_\text{cols} = \text{"issue}\_\text{d"})
\]
\[
\text{class}(\text{dat}[,"issue}\_\text{d"})
\]

# transfer a group of variables with similar name.
\[
\text{dat} = \text{time}\_\text{transfer}(\text{dat} = \text{lendingclub}, \text{date}\_\text{cols} = \text{"_d"})
\]
\[
\text{class}(\text{dat}[,"issue}\_\text{d"})
\]

# transfer all time variables.
\[
\text{dat} = \text{time}\_\text{transfer}(\text{dat} = \text{lendingclub}, \text{date}\_\text{cols} = \text{NULL})
\]
\[
\text{class}(\text{dat}[,"issue}\_\text{d"})
\]

**time_variable**

This function is not intended to be used by end user.

**Usage**

\[
\text{time}\_\text{variable}(\text{dat}, \text{date}\_\text{cols} = \text{NULL}, \text{enddate} = \text{NULL})
\]
**time_vars_process**

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dat</td>
<td>A data.frame.</td>
</tr>
<tr>
<td>date_cols</td>
<td>Time variables.</td>
</tr>
<tr>
<td>enddate</td>
<td>End time.</td>
</tr>
</tbody>
</table>

**Description**

This function is not intended to be used by end user.

**Usage**

```r
time_vars_process(df_tm = df_tm, x, enddate = NULL)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>df_tm</td>
<td>A data.frame</td>
</tr>
<tr>
<td>x</td>
<td>Time variable.</td>
</tr>
<tr>
<td>enddate</td>
<td>End time.</td>
</tr>
</tbody>
</table>

**tnr_value**

**tnr_value**

**Description**

`tnr_value` is for get true negative rate for a prob or score.

**Usage**

```r
tnr_value(prob, target)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>prob</td>
<td>A list of predict probability or score.</td>
</tr>
<tr>
<td>target</td>
<td>Vector of target.</td>
</tr>
</tbody>
</table>

**Value**

True Positive Rate
training_model

Training model

Description

training_model Model builder

Usage

training_model(model_name = "mymodel", dat, dat_test = NULL,
target = NULL, occur_time = NULL, obs_id = NULL, x_list = NULL,
ex_cols = NULL, pos_flag = NULL, prop = 0.7, preproc = TRUE,
low_var = 0.99, missing_rate = 0.98, merge_cat = 30,
outlier_proc = TRUE, missing_proc = "median", miss_values = NULL,
one_hot = FALSE, trans_log = FALSE, feature_filter = list(filter =
c("IV", "PSI", "COR", "XGB"), iv_cp = 0.02, psi_cp = 0.1, xgb_cp = 0,
cv_folds = 1, hopper = FALSE), algorithm = list("LR", "XGB", "GBM",
"RF"), LR.params = lr_params(), XGB.params = xgb_params(),
GBM.params = gbm_params(), RF.params = rf_params(),
breaks_list = NULL, parallel = FALSE, cores_num = NULL,
save_pmml = FALSE, plot_show = FALSE, vars_plot = TRUE,
model_path = tempdir(), seed = 46, ...)

Arguments

model_name A string, name of the project. Default is "mymodel"
dat A data.frame with independent variables and target variable.
dat_test A data.frame of test data. Default is NULL.
target The name of target variable.
occur_time The name of the variable that represents the time at which each observation takes place. Default is NULL.
obs_id The name of ID of observations or key variable of data. Default is NULL.
x_list Names of independent variables. Default is NULL.
ex_cols Names of excluded variables. Regular expressions can also be used to match variable names. Default is NULL.
pos_flag The value of positive class of target variable, default: "1".
prop Percentage of train-data after the partition. Default: 0.7.
preproc Logical. Preprocess data. Default is TRUE.
low_var Logical, delete low variance variables or not. Default is TRUE.
missing_rate The maximum percent of missing values for recoding values to missing and non_missing.
merge_cat merge categories of character variables that is more than m.
outlier_proc Logical, process outliers or not. Default is TRUE.
missing_proc  If logical, process missing values or not. If "median", then Nas imputation with k neighbors median. If "avg_dist", the distance weighted average method is applied to determine the NAs imputation with k neighbors. If "default", assigning the missing values to -1 or "missing", otherwise ,processing the missing values according to the results of missing analysis.

miss_values  Other extreme value might be used to represent missing values, e.g: -9999, -9998. These miss_values will be encoded to -1 or "missing".

one_hot  Logical. If TRUE, one-hot_encoding of category variables. Default is FALSE.

trans_log  Logical, Logarithmic transformation. Default is FALSE.

feature_filter  Parameters for selecting important and stable features. See details at: feature_selector

algorithm  Algorithms for training a model. list("LR", "XGB", "GBDT", "RF") are available.

LR.params  Parameters of logistic regression & scorecard. See details at : lr_params.

XGB.params  Parameters of xgboost. See details at : xgb_params.

GBM.params  Parameters of GBM. See details at : gbm_params.

RF.params  Parameters of Random Forest. See details at : rf_params.

breaks_list  A table containing a list of splitting points for each independent variable. Default is NULL.

parallel  Default is FALSE.

cores_num  The number of CPU cores to use.

save_pmml  Logical, save model in PMML format. Default is TRUE.

plot_show  Logical, show model performance in current graphic device. Default is FALSE.

vars_plot  Logical, if TRUE, plot distribution ,correlation or partial dependence of model input variables . Default is TRUE.

model_path  The path for periodically saved data file. Default is tempdir().

seed  Random number seed. Default is 46.

...  Other parameters.

Value
A list containing Model Objects.

See Also

train_test_split, data_cleansing, feature_selector, lr_params, xgb_params, gbm_params, rf_params, fast_high_cor_filter, get_breaks_all, lasso_filter, woe_trans_all, get_logistic_coef, score_transfer, get_score_card, model_key_index, ks_psi_plot, get_plots, ks_table_plot

Examples

sub = cv_split(UCICreditCard, k = 30)[[1]]
dat = UCICreditCard[sub,]
x_list = c("LIMIT_BAL")
B_model = training_model(dat = dat,
model_name = "UCICreditCard",
target = "default.payment.next.month",
x_list = x_list,
occurrence_time = NULL,
obs_id = NULL,
dat_test = NULL,
preproc = FALSE,
outlier_proc = FALSE,
missing_proc = FALSE,
feature_filter = NULL,
algorithm = list("LR"),
LR.params = lr_params(lasso = FALSE,
                  step_wise = FALSE,
                  score_card = FALSE),
breaks_list = NULL,
parallel = FALSE,
cores_num = NULL,
save_pmm = FALSE,
plot_show = FALSE,
vars_plot = FALSE,
model_path = tempdir(),
seed = 46)

Description

train_lr is for training the logistic regression model using in \texttt{training_model}.

Usage

\begin{verbatim}
train_lr(dat_train, dat_test = NULL, target, x_list = NULL, occurrence_time = NULL, prop = 0.7, tree_control = list(p = 0.02, cp = 1e-08, xval = 5, maxdepth = 10), bins_control = list(bins_num = 10, bins_pct = 0.05, b_chi = 0.02, b_odds = 0.1, b_psi = 0.03, b_or = 0.15, mono = 0.2, odds_psi = 0.15, kc = 1), thresholds = list(cor_p = 0.8, iv_i = 0.02, psi_i = 0.1, cos_i = 0.6), lasso = TRUE,
step_wise = TRUE, best_lambda = "lambda.auc", seed = 1234, ...)
\end{verbatim}

Arguments

- \texttt{dat_train} data.frame of train data. Default is NULL.
- \texttt{dat_test} data.frame of test data. Default is NULL.
- \texttt{target} name of target variable.
- \texttt{x_list} names of independent variables. Default is NULL.
occur_time: The name of the variable that represents the time at which each observation takes place. Default is NULL.

prop: Percentage of train-data after the partition. Default: 0.7.

tree_control: the list of parameters to control cutting initial breaks by decision tree. See details at: get_tree_breaks

bins_control: the list of parameters to control merging initial breaks. See details at: select_best_breaks, select_best_class

thresholds: Thresholds for selecting variables.
   - cor_p: The maximum threshold of correlation. Default: 0.8.
   - iv_i: The minimum threshold of IV. 0.01 to 0.1 usually work. Default: 0.02.
   - psi_i: The maximum threshold of PSI. 0.1 to 0.3 usually work. Default: 0.1.
   - cos_i: Cos_similarity of positive rate of train and test. 0.7 to 0.9 usually work. Default: 0.5.

lasso: Logical, if TRUE, variables filtering by LASSO. Default is TRUE.

step_wise: Logical, stepwise method. Default is TRUE.

best_lambda: Methods of best lambda standards using to filter variables by LASSO. There are 3 methods: "lambda.auc", "lambda.ks", "lambda.sim_sign". Default is "lambda.auc".

seed: Random number seed. Default is 1234.

...: Other parameters

---

train_test_split: Train-Test-Split

Description

train_test_split Functions for partition of data.

Usage

train_test_split(dat, prop = 0.7, split_type = "Random",
                 occur_time = NULL, cut_date = NULL, start_date = NULL,
                 save_data = FALSE, dir_path = tempdir(), file_name = NULL,
                 note = FALSE, seed = 43)

Arguments

dat: A data.frame with independent variables and target variable.

prop: The percentage of train data samples after the partition.

split_type: Methods for partition.
   - "Random" is to split train & test set randomly.
   - "OOT" is to split by time for observation over time test.
• "byRow" is to split by rownumbers.

occur_time The name of the variable that represents the time at which each observation takes place. It is used for "OOT" split.
cut_date Time points for splitting data sets, e.g.: splitting Actual and Expected data sets.
start_date The earliest occurrence time of observations.
save_data Logical, save results in locally specified folder. Default is FALSE.
dir_path The path for periodically saved data file. Default is "/data".
file_name The name for periodically saved data file. Default is "dat".
note Logical. Outputs info. Default is TRUE.
seed Random number seed. Default is 46.

Value
A list of indices (train-test)

Examples

train_test <- train_test_split(lendingclub,
split_type = "OOT", prop = 0.7,
occur_time = "issue_d", seed = 12, save_data = FALSE)
dat_train = train_test$train
dat_test = train_test$test

train_xgb Training XGBoost

Description

train_xgb is for training a xgb model using in training_model.

Usage

train_xgb(seed_number = 1234, dtrain, nthread = 2, nfold = 1,
watchlist = NULL, nrounds = 100, f_eval = "ks",
early_stopping_rounds = 10, verbose = 0, params = NULL, ...)

Arguments

seed_number Random number seed. Default is 1234.
dtrain train-data of xgb.DMatrix datasets.
nthread Number of threads
nfold Number of the cross validation of xgboost
watchlist named list of xgb.DMatrix datasets to use for evaluating model performance generating by xgb_data
Description

This research aimed at the case of customers’s default payments in Taiwan and compares the predictive accuracy of probability of default among six data mining methods. This research employed a binary variable, default payment (Yes = 1, No = 0), as the response variable. This study reviewed the literature and used the following 24 variables as explanatory variables.

Format

A data frame with 30000 rows and 26 variables.

Details

- ID: Customer id
- apply_date: This is a fake occur time.
- LIMIT_BAL: Amount of the given credit (NT dollar): it includes both the individual consumer credit and his/her family (supplementary) credit.
- SEX: Gender (male; female).
- EDUCATION: Education (1 = graduate school; 2 = university; 3 = high school; 4 = others).
- MARRIAGE: Marital status (1 = married; 2 = single; 3 = others).
- AGE: Age (year) History of past payment. We tracked the past monthly payment records (from April to September, 2005) as follows:
  - PAY_0: the repayment status in September
  - PAY_2: the repayment status in August
  - PAY_3: ...
  - PAY_4: ...
  - PAY_5: ...
• PAY_6: the repayment status in April. The measurement scale for the repayment status is:
  -1 = pay duly; 1 = payment delay for one month; 2 = payment delay for two months; ... 8 =
  payment delay for eight months; 9 = payment delay for nine months and above. Amount of
  bill statement (NT dollar)
• BILL_AMT1: amount of bill statement in September
• BILL_AMT2: amount of bill statement in August
• BILL_AMT3: ...
• BILL_AMT4: ...
• BILL_AMT5: ...
• BILL_AMT6: amount of bill statement in April. Amount of previous payment (NT dollar)
• PAY_AMT1: amount paid in September
• PAY_AMT2: amount paid in August
• PAY_AMT3: ...
• PAY_AMT4: ...
• PAY_AMT5: ...
• PAY_AMT6: amount paid in April
• default.payment.next.month: default payment (Yes = 1, No = 0), as the response variable

Source

http://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients

See Also

lendingclub

Description

This function is not intended to be used by end user.

Usage

variable_process(add)

Arguments

add A data.frame
**Description**

This function is used for grouped numeric data processing.

**Usage**

```r
var_group_proc(dat, ID = NULL, group = NULL, num_var = NULL)
```

**Arguments**

- `dat`: A data.frame contained only predict variables.
- `ID`: The name of ID of observations or key variable of data. Default is NULL.
- `group`: The group of behavioral or status variables.
- `num_var`: The name of numeric variable to process.

**Examples**

```r
dat = data.frame(id = c(1,1,1,2,2,3,3,3,4,4,4,4,4,5,5,6,7,7,8,8,8,9,9,9,10,10,11,11,11,11,11,11,11),
terms = c('Vara','Varb','Varc','Vara','Varc','Vara','Varc','Varb','Varb','Varc','Varb','Varc','Varb',
         'Varc','Varh','Varh','Vari','Varc','Varg','Varh','Vark','Vark'),
time = c(8,3,1,9,6,1,4,9,1,3,4,8,2,7,1,3,4,1,8,7,2,5,7,8,8,2,1,5,7,2,7,3))
time_series_proc(dat = dat, ID = 'id', group = 'terms', time = 'time')
```

---

**woe_trans_all**

**WOE Transformation**

**Description**

`woe_trans_all` is for transforming data to woe. The `woe_trans_all` function is a simpler wrapper for `woe_trans`.

**Usage**

```r
woe_trans_all(dat, x_list = NULL, ex_cols = NULL, bins_table = NULL, target = NULL,
              breaks_list = NULL, note = FALSE, save_data = FALSE, parallel = FALSE,
              woe_name = FALSE, file_name = NULL, dir_path = tempdir(), ...)
```

```r
woe_trans(dat, x, bins_table = NULL, target = NULL, breaks_list = NULL, woe_name = FALSE)
```
woe_trans_all

Arguments

dat (data.frame) A data.frame with independent variables.

x_list (list) A list of x variables.

ex_cols (character or list) Names of excluded variables. Regular expressions can also be used to match variable names. Default is NULL.

bins_table (data.frame) A table contains woe of each bin of variables, it is generated by codeget_bins_table_all, codeget_bins_table

target (character) The name of target variable. Default is NULL.

breaks_list (list) A list contains breaks of variables. it is generated by codeget_breaks_all, codeget_breaks

note (logical) Logical, outputs info. Default is TRUE.

save_data (logical) Logical, save results in locally specified folder. Default is TRUE

parallel (logical) Logical, parallel computing. Default is FALSE.

woe_name (logical) Logical. Add "_woe" at the end of the variable name.

file_name (character) The name for periodically saved woe file. Default is "dat_woe".

dir_path (character) The path for periodically saved woe file. Default is "./data".

Additional parameters.

x (character) The name of an independent variable.

Value

A list of breaks for each variables.

See Also

get_tree_breaks, cut_equal, select_best_class, select_best_breaks

Examples

sub = cv_split(UCICreditCard, k = 30)[[1]]
dat = UCICreditCard[sub,]
dat = re_name(dat, "default.payment.next.month", "target")
dat = data_cleansing(dat, target = "target", obs_id = "ID", occur_time = "apply_date", miss_values = list("", -1))

train_test <- train_test_split(dat, split_type = "OOT", prop = 0.7, occur_time = "apply_date")
dat_train = train_test$train
dat_test = train_test$test

# get breaks of all predictive variables
x_list = c("PAY_0", "LIMIT_BAL", "PAY_AMT5", "EDUCATION", "PAY_3", "PAY_2")
breaks_list <- get_breaks_all(dat = dat_train, target = "target", x_list = x_list, occur_time = "apply_date", ex_cols = "ID",
save_data = FALSE, note = FALSE)

# woe transform
train_woe = woe_trans_all(dat = dat_train, target = "target", breaks_list = breaks_list,
xgb_data

Usage

xgb_data(dat_train, target, dat_test = NULL, x_list = NULL, prop = 0.7, occur_time = NULL)

Arguments

dat_train  data.frame of train data. Default is NULL.
target     name of target variable.
dat_test   data.frame of test data. Default is NULL.
x_list     names of independent variables of raw data. Default is NULL.
prop        Percentage of train-data after the partition. Default: 0.7.
occur_time  The name of the variable that represents the time at which each observation takes place. Default is NULL.

xgb_filter

Usage

xgb_filter(dat_train, dat_test = NULL, target = NULL, pos_flag = NULL, x_list = NULL, occur_time = NULL, ex_cols = NULL, xgb_params = list(nrounds = 100, max_depth = 6, eta = 0.1, min_child_weight = 1, subsample = 1, colsample_bytree = 1, gamma = 0, scale_pos_weight = 1, early_stopping_rounds = 10, objective = "binary:logistic"), f_eval = "auc", cv_folds = 1, cp = NULL, seed = 46, vars_name = TRUE, note = TRUE, save_data = FALSE, file_name = NULL, dir_path = tempdir(), ...)

Description

xgb_data is for prepare data using in training_model.

Description

xgb_filter is for selecting important features using xgboost.
Arguments

dat_train  A data.frame with independent variables and target variable.
dat_test  A data.frame of test data. Default is NULL.
target  The name of target variable.
pos_flag  The value of positive class of target variable, default: "1".
x_list  Names of independent variables.
occur_time  The name of the variable that represents the time at which each observation takes place.
ex_cols  A list of excluded variables. Regular expressions can also be used to match variable names. Default is NULL.
f_eval  Customized evaluation function, "ks" & "auc" are available.
cv_folds  Number of cross-validations. Default: 5.
cp  Threshold of XGB feature’s Gain. Default is 1/number of independent variables.
seed  Random number seed. Default is 46.
vars_name  Logical, output a list of filtered variables or table with detailed IV and PSI value of each variable. Default is FALSE.
note  Logical, outputs info. Default is TRUE.
save_data  Logical, save results results in locally specified folder. Default is FALSE.
file_name  The name for periodically saved results files. Default is "Feature_importance_XGB".
dir_path  The path for periodically saved results files. Default is "./variable".
...  Other parameters to pass to xgb_params.

Value

Selected variables.

See Also

psi_iv_filter, gbm_filter, feature_selector

Examples

dat = UCICreditCard[1:1000, c(2, 4, 8:9, 26)]
xgb_params = list(nrounds = 100, max_depth = 6, eta = 0.1,
  min_child_weight = 1, subsample = 1,
  colsample_bytree = 1, gamma = 0, scale_pos_weight = 1,
  early_stopping_rounds = 10,
  objective = "binary:logistic")

## Not run:
xgb_features <- xgb_filter(dat_train = dat, dat_test = NULL,
target = "default.payment.next.month", occur_time = "apply_date", f_eval = 'ks',
xgb_params = xgb_params,
cv_folds = 1, ex_cols = "ID$|date$|default.payment.next.month$", vars_name = FALSE)

## End(Not run)

---

**xgb_params**  
*XGBoost Parameters*

### Description

`xgb_params` is the list of parameters to train a XGB model using in `training_model`. `xgb_params_search` is for searching the optimal parameters of xgboost, if any parameters of params in `xgb_params` is more than one.

### Usage

```r
xgb_params(nrounds = 1000, params = list(max_depth = 6, eta = 0.01,  
gamma = 0, min_child_weight = 1, subsample = 1, colsample_bytree = 1,  
scale_pos_weight = 1), early_stopping_rounds = 100,  
method = "random_search", iters = 10, f_eval = "auc", nfold = 1,  
nthread = 2, ...)
```

```r
xgb_params_search(dat_train, target, dat_test = NULL, x_list = NULL,  
prop = 0.7, occur_time = NULL, method = "random_search",  
iters = 10, nrounds = 100, early_stopping_rounds = 10,  
params = list(max_depth = 6, eta = 0.01, gamma = 0, min_child_weight = 1,  
subsample = 1, colsample_bytree = 1, scale_pos_weight = 1),  
f_eval = "auc", nfold = 1, nthread = 2, ...)
```

### Arguments

- `nrounds`: Max number of boosting iterations.
- `early_stopping_rounds`: If NULL, the early stopping function is not triggered. If set to an integer `k`, training with a validation set will stop if the performance doesn’t improve for `k` rounds.
- `method`: Method of searching optimal parameters. "random_search", "grid_search", "local_search" are available.
- `iters`: Number of iterations of "random_search" optimal parameters.
- `f_eval`: Customized evaluation function, "ks" & "auc" are available.
- `nfold`: Number of the cross validation of xgboost
- `nthread`: Number of threads
- `...`: Other parameters
- `dat_train`: A data.frame of train data. Default is NULL.
Fuzzy String matching

Description

Fuzzy String matching

Usage

x %alike% y

Arguments

x A string.
y A string.

Value

Logical.

Examples

"xyz" %alike% "xy"
%islike%

Fuzzy String matching

Description
Fuzzy String matching

Usage
x %islike% y

Arguments
x A string.
y A string.

Value
Logical.

Examples
"xyz" %islike% "yz$"
Index

*Topic datasets
   ewm_data, 29
   lendingclub, 62
   UCICreditCard, 119
%alike%, 126
%islike%, 127

add_variable_process, 6
address_varieble, 5
analysis_nas, 6, 88
analysis_outliers, 7
as_percent, 8
auc_value, 8
avg_x (rowAny), 99
char_cor (char_cor_vars), 9
char_cor_vars, 9, 31
char_to_num, 10
check_data_format (read_data), 94
check_rules, 11, 42, 101
checking_data, 10
city_varieble, 12
city_varieble_process, 12
cnt_x (rowAny), 99
cohort_analysis, 13
cohort_plot (cohort_table_plot), 13
cohort_table (cohort_analysis), 13
cohort_table_plot, 13
colAllnas (rowAny), 99
colAllzeros (rowAny), 99
colMaxMins (rowAny), 99
colSds (rowAny), 99
cor_heat_plot, 14
cor_plot, 15
cos_sim, 15
creditmodel (creditmodel-package), 5
creditmodel-package, 5
cross_table, 16
customer_segmentation, 17
cut_equal, 18, 40, 105, 107, 122
cv_split, 19
data_cleansing, 19, 115
data_exploration, 21
date_cut, 22
de_one_hot_encoding, 25, 76
de_percent, 26
derived_interval, 22
derived_partial_acf, 23
derived_pct, 23
derived_ts (derived_ts_vars), 24
derived_ts_vars, 24
digits_num, 26
e_ij (p_ij), 92
entropy_weight, 27
entry_rate_na, 20, 28
euclid_dist, 28
eval_auc, 29
eval_ks (eval_auc), 29
eval_lift (eval_auc), 29
eval_tnr (eval_auc), 29
ewm_data, 29
fast_high_cor_filter, 30, 115
feature_selector, 31, 35, 91, 115, 124
fuzzy_cluster (fuzzy_cluster_means), 33
fuzzy_cluster_means, 33
gather_data, 33
gbm_filter, 32, 34, 91, 124
gbm_params, 35, 69, 99, 115, 126
get_auc ks lambda, 37
get_bins_table, 53, 86, 122
get_bins_table (get_bins_table_all), 37
get_bins_table_all, 37, 122
get_breaks, 18, 56, 105, 107, 122
get_breaks (get_breaks_all), 39
get_breaks_all, 18, 39, 56, 107, 115, 122
get_correlation_group, 31, 41
INDEX

get_ctree_rules, 11, 42, 101
get_iv, 38, 44, 50, 51
get_iv(get_iv_all), 43
get_iv_all, 38, 43, 44, 50, 51
get_logistic_coef, 44, 115
get_median, 45
get_names, 46, 57
get_nas_random, 47
get_partial_dependence_plots
  (partial_dependence_plot), 77
get_plots, 47, 115
get_psi, 38, 44, 50, 51
get_psi (get_psi_all), 48
get_psi_all, 38, 44, 48, 50, 51
get_psi_iv (get_psi_iv_all), 50
get_psi_iv_all, 50
get_psi_plots, 52
get_score_card, 53, 115
get_shadow_nas, 54
get_sim_sign_lambda, 37, 55
get_tree_breaks, 18, 40, 43, 48, 51, 55, 68, 105, 107, 117, 122
get_x_list, 46, 56
high_cor_filter (fast_high_cor_filter), 30
high_cor_selector, 31, 57
is_date, 58
knn_nas_imp, 58
ks_plot (model_result_plot), 72
ks_psi_plot, 115
ks_psi_plot (ks_table), 59
ks_table, 59, 59
ks_table_plot, 115
ks_table_plot (ks_table), 59
ks_value, 60
lasso_filter, 37, 61, 115
lendingclub, 62, 120
lift_plot (model_result_plot), 72
lift_value, 64
local_outlier_factor, 64
log_trans, 65
log_vars (log_trans), 65
love_color, 66
low_variance_filter, 20, 67

lr_params, 36, 67, 67, 87, 99, 115, 126
lr_params_search (lr_params), 67
lr_vif, 69
max_min_norm, 70
max_x (rowAny), 99
merge_category, 70
min_max_norm, 71
min_x (rowAny), 99
model_key_index, 115
model_key_index (ks_table), 59
model_result_plot, 72
multi_grid, 74
multi_left_join, 75
null_blank_na, 20, 75
one_hot_encoding, 25, 76
outliers_detection, 77
outliers_kmeans_lof (process_outliers), 89
p_ij, 92
p_to_score, 92
partial_dependence_plot, 77
PCA_reduce, 79
perf_table (model_result_plot), 72
plot_bar, 79
plot_box, 80
plot_density, 80
plot_distribution, 81
plot_distribution_x
  (plot_distribution), 81
plot_oot_perf, 82
plot_relative_freq_histogram, 83
plot_table, 84
plot_theme, 85
plot_vars (get_plots), 47
pred_score, 86, 87, 92
pred_xgb, 87
process_nas, 20, 88
process_nas_var (process_nas), 88
process_outliers, 20, 89
psi_iv_filter, 32, 35, 90, 124
psi_plot (get_psi_plots), 52
quick_as_df, 93
ranking_percent_dict
  (ranking_percent_proc), 93
INDEX

ranking_percent_dict_x
(ranking_percent_proc), 93
ranking_percent_proc, 93
ranking_percent_proc_x
(ranking_percent_proc), 93
re_code, 97
re_name, 97
read_data, 94
reduce_high_cor_filter, 95
remove_duplicated, 20, 96
require_packages, 96
rf_params, 36, 69, 87, 98, 115, 126
roc_plot (model_result_plot), 72
rowAll (rowAny), 99
rowAllnas (rowAny), 99
rowAny, 99
rowCVs (rowAny), 99
rowMaxMins (rowAny), 99
rowMaxs (rowAny), 99
rowMins (rowAny), 99
rowSds (rowAny), 99
rules_filter, 11, 42, 100
save_data, 101
score_distribution_plot
(model_result_plot), 72
score_transfer, 102, 115
select_best_breaks, 40, 43, 48, 51, 68, 107, 117, 122
select_best_breaks (select_best_class), 103
select_best_class, 40, 43, 48, 51, 68, 103, 107, 117, 122
select_cor_group
(get_correlation_group), 41
select_cor_list
(get_correlation_group), 41
sim_str, 105
split_bins, 106
split_bins_all, 106
start_parallel_computing, 108
stop_parallel_computing, 108
str_match, 109
sum_x (rowAny), 99
swap_analysis, 109
term_filter (term_tfidf), 110
term_idf (term_tfidf), 110
term_tfidf, 110
time_series_proc, 111
time_transfer, 112
time_variable, 112
time_vars_process, 113
tnr_value, 113
train_lr, 116
train_test_split, 115, 117
train_xgb, 118
training_model, 35, 36, 67, 69, 86, 87, 92, 98, 99, 114, 116, 118, 123, 125, 126
UCICreditCard, 63, 119
var_group_proc, 121
variable_process, 120
woe_trans (woe_trans_all), 121
woe_trans_all, 115, 121
xgb_data, 118, 123
xgb_filter, 32, 35, 91, 123
xgb_params, 36, 69, 87, 99, 115, 125, 125
xgb_params_search (xgb_params), 125