Package 'creditmodel'

April 28, 2019

Version 1.0

Date 2019-04-05

Title Build Binary Classification Models in One Integrated Offering

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

Description Provides a toolkit for building predictive models in one integrated offering. Contains infrastructure functionalities such as data exploration and preparation, missing values treatment, outliers treatment, variable derivation, variable selection, dimensionality reduction, grid search for hyperparameters, data mining and visualization, model evaluation, strategy analysis etc. 'creditmodel' is designed to make the development of binary classification models (machine learning based models as well as credit scorecard) simpler and faster.

Depends R(>= 3.3.0)

Imports data.table,dplyr,randomForest,xgboost,glmnet,gbm,gridExtra,ggplot2(>= 1.0.1),car,foreach,doParallel,ggcorrplot,pmml,XML,rpart, sqldf,stringr

Suggests knitr,testthat

VignetteBuilder knitr

URL https://github.com/FanHansen/automodel

BugReports https://github.com/FanHansen/automodel/issues

Encoding UTF-8

ByteCompile yes

LazyData yes

LazyLoad yes

License AGPL-3

RoxygenNote 6.1.1

NeedsCompilation no

Author Dongping Fan [aut, cre]

Repository CRAN

Date/Publication 2019-04-28 07:30:03 UTC

R topics documented:

`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

`df`	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.

`add_variable_process` *add_variable_process*

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.

analysis_nas *Missing Analysis*

Description

#' analysis_nas is the function for missing value analysis

Usage

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

Arguments

dat	A data.frame with independent variables and target variable.
class_var	Logical, nas 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 *Outliers Analysis*

Description

#' analysis_outliers is the function for outliers analysis.

Usage

```
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)
```

`char_cor_vars` *Cramer's V matrix between categorical variables.*

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

```
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

```
## 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 transfering character variables which are actually numerical numbers containing strings to numeric.

Usage

```
char_to_num(dat, note = FALSE,
  ex_cols = "date$|id$|time$|DATA$|ID$|TIME$")
```

Arguments

dat	A data frame
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

```
dat_sub = lendingclub[c("mths_since_recent_revol_delinq", "mths_since_last_record")]
str(dat_sub)
#variables that are converted to numbers containing strings
dat_sub[is.na(dat_sub)] = "Unknown"
str(dat_sub)
dat_sub = char_to_num(dat_sub)
str(dat_sub)
```

---

checking_data *Checking Data*

Description

checking_data cheking dat before processing.

Usage

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

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

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

city_varieble *city_varieble*

Description

This function is used for city variables derivation.

Usage

```
city_varieble(df = df, city_cols = NULL, city_pattern = "city$",
  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.

city_varieble_process *Processing of Address Variables*

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.

cleaning_data *Data Cleaning*

Description

The cleaning_data 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

Usage

```
cleaning_data(dat, target = NULL, x_list = NULL, obs_id = NULL,
  occur_time = NULL, pos_flag = NULL, miss_values = NULL,
  ex_cols = NULL, outlier_proc = TRUE, missing_proc = TRUE,
  default_miss = TRUE, low_var = TRUE, parallel = FALSE,
  note = FALSE, save_data = FALSE, dir_path = tempdir(),
  file_name = NULL)
```

Arguments

dat	A data frame with x and target.
target	The name of target variable.
x_list	A list of x variables.
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".
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 "Unknown".
ex_cols	A list of excluded variables. Default is NULL.
outlier_proc	Logical, process outliers or not. Default is TRUE.
missing_proc	Logical, process nas or not. Default is TRUE.
default_miss	Logical. If TRUE, assigning the missing values to -1 or "Unknown", otherwise ,processing the missing values according to the results of missing analysis.
low_var	Logical, delete low variance variables or not. Default is TRUE.
parallel	Logical, parallel computing or not. Default is FALSE.
note	Logical. Outputs info. Default is TRUE.
save_data	Logical, save the result or not. 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 NULL.

Value

A preprocessed data.frame

See Also

remove_duplicated, null_blank_na, entry_rate_max, entry_rate_na, low_variance_filter, process_nas, process_outliers

Examples

```
#data cleaning
dat_cl <- cleaning_data(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,
                        default_miss = FALSE,
                        low_var = TRUE,
                        save_data = FALSE)
```

---

cor_plot *Correlation Plot*

Description

cor_plot is for ploting correlation matrix

Usage

```
cor_plot(dat, dir_path = tempdir(), 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 TRUE
plot_show	Logical, show graph in current graphic device.

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_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, margin = 1)
```

Arguments

x	A list
y	A list
margin	rows or cols

customer_segmentation *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 =
  0.000001, 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 number(default is 100) . • meth Method of clustering. Provides two mehods,"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

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

Examples

```
clust <- customer_segmentation(dat = lendingclub[1:10000,40:50],
                               x_list = NULL, ex_cols = "id$|loan_status",
                               cluster_control = list(meth = "FCM", kc = 2),
                           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 = list(-1, "Unknown"))
```

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, "Unknown")

See Also

get_breaks, get_breaks_all,get_tree_breaks

Examples

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

---

cv_split *Stratified Folds*

Description

this function creates stratified folds for cross validation.

Usage

```
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

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

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 *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 *derived_partial_acf*

Description

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

Usage

```
derived_partial_acf(dat_s)
```

Arguments

dat_s	A data.frame

derived_pct *derived_pct*

Description

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

Usage

```
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

```
derived_ts_vars(dat, grx, td = 12, der = c("cvs", "sums", "means",
  "maxs", "max_mins", "time_intervals", "cnt_intervals", "total_pcts",
  "cum_pcts", "partial_acfs"), parallel = TRUE)

derived_ts(dat = dat, grx_x = NULL, td = 12, 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.
der	Variables to derivate
parallel	Logical, parallel computing. Default is FALSE.
grx_x	Regular expression used to match a group of variable names.

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 "Unknown", 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 recorery 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 recoverying percent format..

Usage

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

Arguments

x	Character with percent formant.
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 caculating optimal digits number for numeric variables.

Usage

```
digits_num(num)
```

Arguments

num	A sample of numeric variable.

Value

A number of digits

Examples

```
## Not run:
digits_num(3.1415296)
# 7

## End(Not run)
```

entry_rate_max *Max Percent of Unique Values*

Description

entry_rate_max is the function to encode variables which unique value rate is more than 95

Usage

```
entry_rate_max(dat, rt = 0.95, note = FALSE)
```

Arguments

dat	A data frame with x and target.
rt	The maximum percent of unique value.
note	Logical.Outputs info.Default is TRUE.

Value

A data.frame

Examples

```
datss = entry_rate_max(dat = UCICreditCard[1:1000, ] , rt  = 0.98)
```

---

entry_rate_na *Max Percent of NAs*

Description

entry_rate_na is the function to encode variables which NAs & miss value rate is more than 95

Usage

```
entry_rate_na(dat, nr = 0.95, 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

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

---

euclid_dist *euclid_dist*

Description

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

Usage

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

Arguments

x	A list
y	A list
margin	rows or cols

fast_high_cor_filter *high_cor_filter*

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 = 0.7, x_list = NULL, com_list = NULL,
  ex_cols = NULL, save_data = FALSE, cor_class = 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 = TRUE, 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 0.7.
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	Culculate catagery variables's correlation matrix. Default is FALSE.
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.

See Also

get_correlation_group, reduce_high_cor, char_cor_vars

Examples

```
# calculate iv for each variable.
iv_list = feature_select_wrapper(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)
```

---

feature_select_wrapper

Feature Selection Wrapper

Description

feature_select_wrapper 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_select_wrapper(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 = FALSE, 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.
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.
note	Logical.Outputs info.Default is TRUE.
seed	Random number seed. Default is 46.
save_data	Logical, save results in locally specified folder. Default is TRUE.
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"
...	Other parameters.

Value

A list of selected features

See Also

psi_iv_filter, xgb_filter, gbm_filter

Examples

```
feature_select_wrapper(dat_train = UCICreditCard[1:1000,c(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

```
fuzzy_cluster_means(dat, kc = 2, sf = 2, nstart = 1,
  max_iter = 100, epsm = 0.000001)

fuzzy_cluster(dat, kc = 2, init_centers, sf = 3, max_iter = 100,
  epsm = 0.000001)
```

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.

gbm_filter *Select Features using GBM*

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 = FALSE, 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.
pos_flag	The value of positive class of target variable, default: "1".
GBM.params	Parameters of GBM.The complete list of parameters is available at: 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 TRUE
file_name	The name for periodically saved results files. Default is "Featrue_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.

Value

Selected variables.

See Also

psi_iv_filter, xgb_filter, feature_select_wrapper

Examples

```
GBM.params = gbm_params(n.trees = 2, interaction.depth = 4, shrinkage = 0.1,
                        bag.fraction = 0.5, train.fraction = 0.7,
                        n.minobsinnode = 30,
                       cv.folds = NULL, best_iter = TRUE, method = "cv")

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)
```

gbm_params *GBM Parameters*

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, ...)
```

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.
interaction.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 randomnesses 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_best_lambda` *get_best_lambda* `plot_theme` *is for get Best lambda required in lasso_filter. This function required in* `lasso_filter`

Description

get_best_lambda `plot_theme` is for get Best lambda required in lasso_filter. This function required in `lasso_filter`

Usage

```
get_best_lambda(lasso_model, lasso_model_cv, type.measure = "auc",
  sim_sign = "negtive")
```

Arguments

`lasso_model`	A lasso model genereted by glmnet.
`lasso_model_cv`	A cv lasso model genereted by cv.glmnet
`type.measure`	Default is "auc".
`sim_sign`	Default is "negtive". This is related to pos_plag. If pos_flag equals 1 or bad, the value must be set to negetive. If pos_flag equals 0 or good, the value must be set to positive.

Details

lambda.min give a model with the best performance but the least number of independent variable. lambda.lse give the simplest model in the range of a standard deviation lambda.05se give the simplest model in the range of a half standard deviation lambda.sim_sign give the model with the same positive or negetive coefficients of all variables. lambda.sim_sign give the model with the same positive or negetive coefficients of all variables.

Value

Four lanmbda values with different thresholds.

get_bins_table_all *Table of Binning*

Description

get_bins_table is used to generates summary information of varaibles. 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, ex_cols = NULL, breaks_list = NULL,
  parallel = FALSE, note = FALSE, occur_time = NULL, oot_pct = 0.7,
  save_data = FALSE, file_name = NULL, dir_path = tempdir())

get_bins_table(dat, x, target = NULL, pos_flag = NULL, breaks = NULL,
  breaks_list = NULL, occur_time = NULL, oot_pct = 0.7,
  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".
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.
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). Defualt is 0.7
save_data	Logical, save results in locally specified folder. Default is TRUE
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", occur_time = "apply_date")
```

---

get_breaks_all *Generates 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

```
get_breaks_all(dat, target = NULL, x_list = NULL, ex_cols = NULL,
  pos_flag = NULL, occur_time = NULL, oot_pct = 0.7, best = FALSE,
  equal_bins = FALSE, g = 10, sp_values = NULL,
  tree_control = list(p = 0.05, cp = 0.000001, 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_gb = 0.15, mono = 0.3, gb_psi = 0.2, kc =
  1), parallel = FALSE, note = FALSE, save_data = FALSE,
  file_name = NULL, dir_path = tempdir(), ...)

get_breaks(dat, x, target = NULL, pos_flag = NULL, best = TRUE,
  equal_bins = FALSE, 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). Defualt 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 desison tree.

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 0.0000001 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_gb The maximum threshold of G/B index in any bins. 0 < b_gb < 1 ; 0.05 to 0.3 usually work. Default: 0.15 • gb_psi The maximum threshold of Training and Testing G/B index PSI in any bins. 0 < gb_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

```
#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,
                    b_psi = 0.05, b_gb = 15, mono = 0.2, gb_psi = 0.1, kc = 5)
# get categrory variable breaks
b <-  get_breaks(dat = UCICreditCard[1:1000,], x = "MARRIAGE",
                 target = "default.payment.next.month",
                 occur_time = "apply_date",
                 sp_values = list(-1, "Unknown"),
                 tree_control = tree_control, bins_control = bins_control)
# get numeric variable breaks
b2 <-  get_breaks(dat = UCICreditCard[1:1000,], x = "PAY_2",
                  target = "default.payment.next.month",
                  occur_time = "apply_date",
                  sp_values = list(-1, "Unknown"),
                  tree_control = tree_control, bins_control = bins_control)
# get breaks of all predictive variables
b3 <-  get_breaks_all(dat = UCICreditCard[1:1000,], target = "default.payment.next.month",
                      x_list = c("MARRIAGE","PAY_2"),
                      occur_time = "apply_date", ex_cols = "ID",
                      sp_values = list(-1, "Unknown"),
                     tree_control = tree_control, bins_control = bins_control,
                      save_data = FALSE)
```

---

get_correlation_group *get_correlation_group*

Description

`get_correlation_group` is funtion for obtaining highly correlated variable groups. `select_cor_group` is funtion for selecting highly correlated variable group. `select_cor_list` is funtion 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.

Examples

```
## Not run:
cor_mat = cor(UCICreditCard[8:20],
use = "complete.obs", method = "spearman")
get_correlation_group(cor_mat, p = 0.6 )

## End(Not run)
```

`get_ctree_rules` *Parse party ctree rules*

Description

`get_ctree_rules` This function is used to parse party ctree rules and percentage of bad 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 = 0.0001, 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 bad percent under each rule.

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)
```

get_iv_all *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.*

Description

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

```
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)

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.
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 detials: get_tree_breaks
bins_control	Parameters used to control binning. See detials: 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.

Details

IV Rules of Thumb for evaluating the strength a predictor Less than 0.02:unpredictive 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")
```

---

get_logistic_coef *get logistic coef*

Description

`get_logistic_coef` is for geting logistic coefficient.

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 TRUE.

Value

A data.frame with logistic coefficients.

Examples

```
# dataset spliting
sub = cv_split(UCICreditCard, k = 30)[[1]]
dat = UCICreditCard[sub,]
#rename the target variable
dat = re_name(dat, "default.payment.next.month", "target")
dat = cleaning_data(dat, target = "target", obs_id = "ID",
occur_time = "apply_date", miss_values =  list("", -1))
#train_ test pliting
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",
                                occur_time = "apply_date",
                                 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_median *get central value.*

Description

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

Usage

```
get_median(x, weight_avg = NULL)
```

Arguments

x	A vector or list.
weight_avg	avarage weigth to calculate means.

`get_names` *Get Variable Names*

Description

`get_names` is for getting names of particular classes of variables

Usage

```
get_names(dat, types = c("numeric", "integer", "double"),
  ex_cols = NULL, get_ex = FALSE)
```

Arguments

dat	A data.frame with independent variables and target variable.
types	The class or types of variables which names to get. Default: c('numeric', 'integer', 'double')
ex_cols	A list of excluded variables. Regular expressions can also be used to match variable names. Default is NULL.
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

```
x_list = get_names(dat = UCICreditCard, types = c('factor', 'character'),
ex_cols = c("default.payment.next.month","ID$|_date$"), get_ex = FALSE)
x_list = get_names(dat = UCICreditCard,
ex_cols = c("default.payment.next.month","ID$|_date$"), get_ex = FALSE)
```

get_nas_random *get_nas_random*

Description

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

Usage

```
get_nas_random(dat)
```

Arguments

dat	A data.frame contained only predict variables.

get_plots *Plot Independent Variables*

Description

plot_vars is used for independent variables with target variable ploting. get_plots can loop through plots for all specified independent variables.

Usage

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

plot_vars(dat_train, x, dat_test = NULL, target = "target",
  g_width = 8, breaks_list = NULL, pos_flag = list("1", 1, "bad",
  "positive"), occur_time = NULL, equal_bins = FALSE, best = TRUE,
  g = 20, 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.

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.
pos_flag	Value of positive class, Default is "1".
occur_time	The name of the variable that represents the time at which each observation takes place.
equal_bins	Logical, generates initial breaks for equal frequency binning.
best	Logical, merge initial breaks to get optimal breaks for binning.
g	Number of initial breakpoints for equal frequency binning.
tree_control	Parameters of using Decision Tree to segment initial breaks. See detials: get_tree_breaks
bins_control	Parameters used to control binning. See detials: select_best_class, select_best_breaks
plot_show	Logical, show model performance in current graphic device. Default is FALSE.
save_data	Logical, save results in locally specified folder. Default is TRUE
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.

Examples

```
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")
```

---

get_psi_all

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.*

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.

Usage

```
get_psi_all(dat, x_list = 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, 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.
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 spliting data sets, e.g. : spliting Actual and Expected data sets.
oot_pct	Percentage of observations retained for overtime test (especially to calculate PSI). Defualt 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

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,
  tree_control = NULL, bins_control = NULL, bins_total = TRUE,
  best = TRUE, g = 10, as_table = TRUE, note = FALSE,
  parallel = FALSE, bins_no = FALSE)

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, tree_control = NULL,
  bins_control = NULL, bins_total = TRUE, best = TRUE, g = 10,
  as_table = TRUE, note = FALSE, bins_no = FALSE)
```

Arguments

`dat`	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.
`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). Defualt is 0.7
equal_bins	Logical, generates initial breaks for equal frequency binning.
tree_control	Parameters of using Decision Tree to segment initial breaks. See detials: get_tree_breaks
bins_control	Parameters used to control binning. See detials: 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 =FALSE)
```

---

get_score_card *Score Card*

Description

get_score_card is for generating a stardard scorecard

Usage

```
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 TRUE

Value

scorecard

Examples

```
# dataset spliting
sub = cv_split(UCICreditCard, k = 30)[[1]]
dat = UCICreditCard[sub,]
#rename the target variable
dat = re_name(dat, "default.payment.next.month", "target")
dat = cleaning_data(dat, target = "target", obs_id = "ID",
occur_time = "apply_date", miss_values =  list("", -1))
#train_ test pliting
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",
                                occur_time = "apply_date",
```

```
                                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` *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_tree_breaks` *Getting the breaks for terminal nodes from decision tree*

Description

`get_tree_breaks` is for generating initial braks 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 = 0.000001, xval = 5, maxdepth = 10), sp_values = NULL)
```

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

```
#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(x_list = NULL, dat_train = NULL, dat_test = NULL,
  ex_cols = NULL)
```

Arguments

x_list	Names of independent variables.
dat_train	A data.frame with independent variables.
dat_test	Another data.frame.
ex_cols	A list of excluded variables. Regular expressions can also be used to match variable names. Default is NULL.

Value

A list contains names of variables

See Also

get_names

Examples

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

---

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 by k neighbors median.

ks_table *ks_table & plot*

Description

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

Usage

```
ks_table(train_pred, test_pred, target = NULL, score = "score",
  g = 10, breaks = NULL, pos_flag = NULL)

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 TRUE
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

```
sub = cv_split(UCICreditCard, k = 30)[[1]]
dat = UCICreditCard[sub,]
dat = re_name(dat, "default.payment.next.month", "target")
dat = cleaning_data(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))

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 *ks_value*

Description

ks_value is for get plots for a variable.

Usage

```
ks_value(dat_pred = NULL, target = NULL, score = NULL, g = 20,
  breaks = NULL)
```

Arguments

dat_pred	A data frame 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.

Examples

```
sub = cv_split(UCICreditCard, k = 30)[[1]]
dat = UCICreditCard[sub,]
dat = re_name(dat, "default.payment.next.month", "target")
dat = cleaning_data(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))

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)
ks_value(dat_pred = dat_train,
target = "target", score ="pred_LR", g = 20)
```

lasso_filter *Selected Variables by LASSO*

Description

lasso_filter filter variables by lasso.

Usage

```
lasso_filter(dat_train, dat_test = NULL, target = NULL,
  x_list = NULL, pos_flag = list(1, "1", "bad"), ex_cols = NULL,
  best_lambda = "lambda.min", sim_sign = "negtive",
  save_data = FALSE, parallel = FALSE, plot.it = TRUE, seed = 46,
  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.
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.
best_lambda	Best lanmbda stardards. one of ("lambda.min", "lambda.1se", "lambda.05se" , "lambda.sim_sign"). Default is "lambda.min".
sim_sign	The coefficients of all variables should be all negetive or positive, after turning to woe. Default is "negetive" for pos_flag is "1".
save_data	Logical, save results in locally specified folder. Default is TRUE
parallel	Logical, parallel computing. 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".

Value

A list of filtered x variables by lasso.

Examples

```
lasso_filter(dat_train = UCICreditCard[1:1000,c(4,12:15,26)],
             target = "default.payment.next.month",
             best_lambda = "lambda.min", 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

- acc_now_delinq: The number of accounts on which the borrower is now delinquent.
- acc_open_past_24mths: Number of trades opened in past 24 months.
- addr_state: The state provided by the borrower in the loan application
- all_util: Balance to credit limit on all trades
- annual_inc: The self:reported annual income provided by the borrower during registration.
- annual_inc_joint: The combined self:reported annual income provided by the co:borrowers during registration
- application_type: Indicates whether the loan is an individual application or a joint application with two co:borrowers
- 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.
- chargeoff_within_12_mths: Number of charge:offs within 12 months
- collection_recovery_fee: post charge off collection fee
- collections_12_mths_ex_med: Number of collections in 12 months excluding medical collections
- delinq_2yrs: The number of 30+ days past:due incidences of delinquency in the borrower's credit file for the past 2 years
- delinq_amnt: The past:due amount owed for the accounts on which the borrower is now delinquent.
- desc: Loan description provided by the borrower
- 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
- earliest_cr_line: The month the borrower's earliest reported credit line was opened
- 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.
- fico_range_high: The upper boundary range the borrower's FICO at loan origination belongs to.
- fico_range_low: The lower boundary range the borrower's FICO at loan origination belongs to.
- funded_amnt: The total amount committed to that loan at that point in time.
- funded_amnt_inv: The total amount committed by investors for that loan at that point in time.
- grade: LC assigned loan grade
- home_ownership: The home ownership status provided by the borrower during registration or obtained from the credit report.Our values are: RENT, OWN, MORTGAGE, OTHER
- id: A unique LC assigned ID for the loan listing.
- il_util: Ratio of total current balance to high credit/credit limit on all install acct
- initial_list_status: The initial listing status of the loan. Possible values are:W, F
- inq_fi: Number of personal finance inquiries
- inq_last_12m: Number of credit inquiries in past 12 months
- inq_last_6mths: The number of inquiries in past 6 months (excluding auto and mortgage inquiries)
- installment: The monthly payment owed by the borrower if the loan originates.
- int_rate: Interest Rate on the loan
- issue_d: The month which the loan was funded
- last_credit_pull_d: The most recent month LC pulled credit for this loan
- last_fico_range_high: The upper boundary range the borrower's last FICO pulled belongs to.
- last_fico_range_low: The lower boundary range the borrower's last FICO pulled belongs to.
- last_pymnt_amnt: Last total payment amount received
- last_pymnt_d: Last month payment was received
- loan_amnt: The listed amount of the loan applied for by the borrower. If at some point in time, the credit department reduces the loan amount, then it will be reflected in this value.
- loan_status: Current status of the loan
- max_bal_bc: Maximum current balance owed on all revolving accounts
- member_id: A unique LC assigned Id for the borrower member.
- 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.
- mths_since_last_delinq: The number of months since the borrower's last delinquency.
- mths_since_last_major_derog: Months since most recent 90:day or worse rating
- mths_since_last_record: The number of months since the last public record.
- mths_since_rcnt_il: Months since most recent installment accounts opened
- mths_since_recent_bc: Months since most recent bankcard account opened.
- mths_since_recent_bc_dlq: Months since most recent bankcard delinquency
- mths_since_recent_inq: Months since most recent inquiry.
- mths_since_recent_revol_delinq: Months since most recent revolving delinquency.
- next_pymnt_d: Next scheduled payment date
- num_accts_ever_120_pd: Number of accounts ever 120 or more days past due
- num_actv_bc_tl: Number of currently active bankcard accounts
- num_actv_rev_tl: Number of currently active revolving trades
- num_bc_sats: Number of satisfactory bankcard accounts
- num_bc_tl: Number of bankcard accounts
- num_il_tl: Number of installment accounts
- num_op_rev_tl: Number of open revolving accounts
- num_rev_accts: Number of revolving accounts
- num_rev_tl_bal_gt_0: Number of revolving trades with balance >0
- num_sats: Number of satisfactory accounts
- num_tl_120dpd_2m: Number of accounts currently 120 days past due (updated in past 2 months)
- num_tl_30dpd: Number of accounts currently 30 days past due (updated in past 2 months)
- num_tl_90g_dpd_24m: Number of accounts 90 or more days past due in last 24 months
- num_tl_op_past_12m: Number of accounts opened in past 12 months
- open_acc: The number of open credit lines in the borrower's credit file.
- open_acc_6m: Number of open trades in last 6 months
- open_il_12m: Number of installment accounts opened in past 12 months
- open_il_24m: Number of installment accounts opened in past 24 months
- open_act_il: Number of currently active installment trades
- open_rv_12m: Number of revolving trades opened in past 12 months
- open_rv_24m: Number of revolving trades opened in past 24 months
- out_prncp: Remaining outstanding principal for total amount funded
- out_prncp_inv: Remaining outstanding principal for portion of total amount funded by investors
- pct_tl_nvr_dlq: Percent of trades never delinquent
- percent_bc_gt_75: Percentage of all bankcard accounts > 75

- policy_code: publicly available policy_code=1 new products not publicly available policy_code=2
- pub_rec: Number of derogatory public records
- pub_rec_bankruptcies: Number of public record bankruptcies
- purpose: A category provided by the borrower for the loan request.
- pymnt_plan: Indicates if a payment plan has been put in place for the loan
- recoveries: post charge off gross recovery
- revol_bal: Total credit revolving balance
- revol_util: Revolving line utilization rate, or the amount of credit the borrower is using relative to all available revolving credit.
- sub_grade: LC assigned loan subgrade
- tax_liens: Number of tax liens
- term: The number of payments on the loan. Values are in months and can be either 36 or 60.
- title: The loan title provided by the borrower
- tot_coll_amt: Total collection amounts ever owed
- tot_cur_bal: Total current balance of all accounts
- tot_hi_cred_lim: 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_bal_il: Total current balance of all installment accounts
- 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
- total_pymnt: Payments received to date for total amount funded
- total_pymnt_inv: Payments received to date for portion of total amount funded by investors
- total_rec_int: Interest received to date
- total_rec_late_fee: Late fees received to date
- total_rec_prncp: Principal received to date
- total_rev_hi_lim: Total revolving high credit/credit limit
- url: URL for the LC page with listing data.
- verification_status: Indicates if income was verified by LC, not verified, or if the income source was verified
- verified_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.
- revol_bal_joint : Sum of revolving credit balance of the co:borrowers, net of duplicate balances
- sec_app_fico_range_low : FICO range (high) for the secondary applicant
- sec_app_fico_range_high : FICO range (low) for the secondary applicant

- sec_app_earliest_cr_line : Earliest credit line at time of application for the secondary applicant
- sec_app_inq_last_6mths : Credit inquiries in the last 6 months at time of application for the secondary applicant
- sec_app_mort_acc : Number of mortgage accounts at time of application for the secondary applicant
- sec_app_open_acc : Number of open trades at time of application for the secondary applicant
- sec_app_revol_util : Ratio of total current balance to high credit/credit limit for all revolving accounts
- sec_app_open_act_il: Number of currently active installment trades at time of application for the secondary applicant
- sec_app_num_rev_accts : Number of revolving accounts at time of application for the secondary applicant
- sec_app_chargeoff_within_12_mths : Number of charge:offs within last 12 months at time of application for the secondary applicant
- sec_app_collections_12_mths_ex_med : Number of collections within last 12 months excluding medical collections at time of application for the secondary applicant
- sec_app_mths_since_last_major_derog : Months since most recent 90:day or worse rating at time of application for the secondary applicant
- hardship_flag: Flags whether or not the borrower is on a hardship plan
- hardship_type: Describes the hardship plan offering
- hardship_reason: Describes the reason the hardship plan was offered
- hardship_status: Describes if the hardship plan is active, pending, canceled, completed, or broken
- deferral_term: Amount of months that the borrower is expected to pay less than the contractual monthly payment amount due to a hardship plan
- hardship_amount: The interest payment that the borrower has committed to make each month while they are on a hardship plan
- hardship_start_date: The start date of the hardship plan period
- hardship_end_date: The end date of the hardship plan period
- payment_plan_start_date: The day the first hardship plan payment is due. For example, if a borrower has a hardship plan period of 3 months, the start date is the start of the three:month period in which the borrower is allowed to make interest:only payments.
- hardship_length: The number of months the borrower will make smaller payments than normally obligated due to a hardship plan
- hardship_dpd: Account days past due as of the hardship plan start date
- hardship_loan_status: Loan Status as of the hardship plan start date
- orig_projected_additional_accrued_interest: The original projected additional interest amount that will accrue for the given hardship payment plan as of the Hardship Start Date. This field will be null if the borrower has broken their hardship payment plan.
- hardship_payoff_balance_amount: The payoff balance amount as of the hardship plan start date

- hardship_last_payment_amount: The last payment amount as of the hardship plan start date
- disbursement_method: The method by which the borrower receives their loan. Possible values are: CASH, DIRECT_PAY
- debt_settlement_flag: Flags whether or not the borrower, who has charged:off, is working with a debt:settlement company.
- debt_settlement_flag_date: The most recent date that the Debt_Settlement_Flag has been set.
- settlement_status: The status of the borrower's settlement plan. Possible values are: COMPLETE, ACTIVE, BROKEN, CANCELLED, DENIED, DRAFT
- settlement_date: The date that the borrower agrees to the settlement plan
- settlement_amount: The loan amount that the borrower has agreed to settle for
- settlement_percentage: The settlement amount as a percentage of the payoff balance amount on the loan
- settlement_term: The number of months that the borrower will be on the settlement plan
- acceptD: The date which the borrower accepted the offer
- accNowDelinq: The number of accounts on which the borrower is now delinquent.
- accOpenPast24Mths: Number of trades opened in past 24 months.
- addrState: The state provided by the borrower in the loan application
- all_util: Balance to credit limit on all trades
- annual_inc_joint: The combined self:reported annual income provided by the co:borrowers during registration
- annualInc: The self:reported annual income provided by the borrower during registration.
- application_type: Indicates whether the loan is an individual application or a joint application with two co:borrowers
- avg_cur_bal: Average current balance of all accounts
- bcOpenToBuy: Total open to buy on revolving bankcards.
- bcUtil: Ratio of total current balance to high credit/credit limit for all bankcard accounts.
- chargeoff_within_12_mths: Number of charge:offs within 12 months
- collections_12_mths_ex_med: Number of collections in 12 months excluding medical collections
- creditPullD: The date LC pulled credit for this loan
- delinq2Yrs: The number of 30+ days past:due incidences of delinquency in the borrower's credit file for the past 2 years
- delinqAmnt: The past:due amount owed for the accounts on which the borrower is now delinquent.
- desc: Loan description provided by the borrower
- 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

- earliestCrLine: The date the borrower's earliest reported credit line was opened
- effective_int_rate: The effective interest rate is equal to theinterestrate on a Note reduced by Lending Club's estimate of the impact of uncollectedinterestprior to charge off.
- emp_title: The job title supplied by the Borrower when applying for the loan.
- empLength: 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.
- expD: The date the listing will expire
- expDefaultRate: The expected default rate of the loan.
- ficoRangeHigh: The upper boundary range the borrower's FICO at loan origination belongs to.
- ficoRangeLow: The lower boundary range the borrower's FICO at loan origination belongs to.
- fundedAmnt: The total amount committed to that loan at that point in time.
- grade: LC assigned loan grade
- homeOwnership: The home ownership status provided by the borrower during registration. Our values are: RENT, OWN, MORTGAGE, OTHER.
- id: A unique LC assigned ID for the loan listing.
- il_util: Ratio of total current balance to high credit/credit limit on all install acct
- ils_exp_d: wholeloan platform expiration date
- initialListStatus: The initial listing status of the loan. Possible values are:W, F
- inq_fi: Number of personal finance inquiries
- inq_last_12m: Number of credit inquiries in past 12 months
- inqLast6Mths: The number of inquiries in past 6 months (excluding auto and mortgage inquiries)
- installment: The monthly payment owed by the borrower if the loan originates.
- intRate: Interest Rateon the loan
- isIncV: Indicates if income was verified by LC, not verified, or if the income source was verified
- listD: The date which the borrower's application was listed on the platform.
- loanAmnt: The listed amount of the loan applied for by the borrower. If at some point in time, the credit department reduces the loan amount, then it will be reflected in this value.
- max_bal_bc: Maximum current balance owed on all revolving accounts
- memberId: A unique LC assigned Id for the borrower member.
- 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
- mortAcc: Number of mortgage accounts.
- msa: Metropolitan Statistical Area of the borrower.
- mths_since_last_major_derog: Months since most recent 90:day or worse rating

- mths_since_oldest_il_open: Months since oldest bank installment account opened
- mths_since_rcnt_il: Months since most recent installment accounts opened
- mthsSinceLastDelinq: The number of months since the borrower's last delinquency.
- mthsSinceLastRecord: The number of months since the last public record.
- mthsSinceMostRecentInq: Months since most recent inquiry.
- mthsSinceRecentBc: Months since most recent bankcard account opened.
- mthsSinceRecentLoanDelinq: Months since most recent personal finance delinquency.
- mthsSinceRecentRevolDelinq: Months since most recent revolving delinquency.
- num_accts_ever_120_pd: Number of accounts ever 120 or more days past due
- num_actv_bc_tl: Number of currently active bankcard accounts
- num_actv_rev_tl: Number of currently active revolving trades
- num_bc_sats: Number of satisfactory bankcard accounts
- num_bc_tl: Number of bankcard accounts
- num_il_tl: Number of installment accounts
- num_op_rev_tl: Number of open revolving accounts
- num_rev_accts: Number of revolving accounts
- num_rev_tl_bal_gt_0: Number of revolving trades with balance >0
- num_sats: Number of satisfactory accounts
- num_tl_120dpd_2m: Number of accounts currently 120 days past due (updated in past 2 months)
- num_tl_30dpd: Number of accounts currently 30 days past due (updated in past 2 months)
- num_tl_90g_dpd_24m: Number of accounts 90 or more days past due in last 24 months
- num_tl_op_past_12m: Number of accounts opened in past 12 months
- open_acc_6m: Number of open trades in last 6 months
- open_il_12m: Number of installment accounts opened in past 12 months
- open_il_24m: Number of installment accounts opened in past 24 months
- open_act_il: Number of currently active installment trades
- open_rv_12m: Number of revolving trades opened in past 12 months
- open_rv_24m: Number of revolving trades opened in past 24 months
- openAcc: The number of open credit lines in the borrower's credit file.
- pct_tl_nvr_dlq: Percent of trades never delinquent
- percentBcGt75: Percentage of all bankcard accounts > 75
- pub_rec_bankruptcies: Number of public record bankruptcies
- pubRec: Number of derogatory public records
- purpose: A category provided by the borrower for the loan request.
- reviewStatus: The status of the loan during the listing period. Values: APPROVED, NOT_APPROVED.
- reviewStatusD: The date the loan application was reviewed by LC

- revolBal: Total credit revolving balance
- revolUtil: Revolving line utilization rate, or the amount of credit the borrower is using relative to all available revolving credit.
- serviceFeeRate: Service fee rate paid by the investor for this loan.
- subGrade: LC assigned loan subgrade
- tax_liens: Number of tax liens
- term: The number of payments on the loan. Values are in months and can be either 36 or 60.
- title: The loan title provided by the borrower
- tot_coll_amt: Total collection amounts ever owed
- tot_cur_bal: Total current balance of all accounts
- tot_hi_cred_lim: Total high credit/credit limit
- total_bal_il: Total current balance of all installment accounts
- total_cu_tl: Number of finance trades
- total_il_high_credit_limit: Total installment high credit/credit limit
- total_rev_hi_lim: Total revolving high credit/credit limit
- totalAcc: The total number of credit lines currently in the borrower's credit file
- totalBalExMort: Total credit balance excluding mortgage
- totalBcLimit: Total bankcard high credit/credit limit
- url: URL for the LC page with listing data.
- verified_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.
- revol_bal_joint : Sum of revolving credit balance of the co:borrowers, net of duplicate balances
- sec_app_fico_range_low : FICO range (high) for the secondary applicant
- sec_app_fico_range_high : FICO range (low) for the secondary applicant
- sec_app_earliest_cr_line : Earliest credit line at time of application for the secondary applicant
- sec_app_inq_last_6mths : Credit inquiries in the last 6 months at time of application for the secondary applicant
- sec_app_mort_acc : Number of mortgage accounts at time of application for the secondary applicant
- sec_app_open_acc : Number of open trades at time of application for the secondary applicant
- sec_app_revol_util : Ratio of total current balance to high credit/credit limit for all revolving accounts
- sec_app_open_act_il: Number of currently active installment trades at time of application for the secondary applicant
- sec_app_num_rev_accts : Number of revolving accounts at time of application for the secondary applicant
- sec_app_chargeoff_within_12_mths : Number of charge:offs within last 12 months at time of application for the secondary applicant

- sec_app_collections_12_mths_ex_med : Number of collections within last 12 months excluding medical collections at time of application for the secondary applicant
- sec_app_mths_since_last_major_derog : Months since most recent 90:day or worse rating at time of application for the secondary applicant
- disbursement_method: The method by which the borrower receives their loan. Possible values are: CASH, DIRECT_PAY
- Amount Requested: The total amount requested by the borrower
- Application Date: The date which the borrower applied
- Loan Title: The loan title provided by the borrower
- Risk_Score: For applications prior to November 5, 2013 the risk score is the borrower's FICO score. For applications after November 5, 2013 the risk score is the borrower's Vantage score.
- Debt_To_Income Ratio: 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.
- Zip Code: The first 3 numbers of the zip code provided by the borrower in the loan application.
- State: The state provided by the borrower in the loan application
- Employment 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.
- Policy Code: publicly available policy_code=1 new products not publicly available policy_code=2

Source

https://www.lendingclub.com/info/download:data.action

See Also

UCICreditCard

`local_outlier_factor` *local_outlier_factor* `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.*

Description

local_outlier_factor `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

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

Arguments

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

loop_function *Loop Function. #' loop_function is an iterator to loop through*

Description

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

Usage

```
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 argauments required by function.
x_list	Names of objects to loop through.
bind	Complie 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

```
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)
```

love_color *love_color*

Description

love_color is for get plots for a variable.

Usage

```
love_color(color = "dark_cyan")
```

Arguments

color	The name of colors.

Examples

```
love_color("dark_cyan")
```

---

low_variance_filter *Filtering Low Variance Variables*

Description

low_variance_filter is the function for filtering low variance variables.

Usage

```
low_variance_filter(dat, lvp = 0.97, note = FALSE)
```

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.

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.

Usage

```
lr_params(tree_control = list(p = 0.02, cp = 0.00000001, 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_gb = 0.15, mono = 0.2, gb_psi
  = 0.15, kc = 1), best_lambda = "lambda.sim_sign", sp_values = NULL,
  forced_in = NULL, obsweight = c(1, 1), lasso = TRUE,
  vars_plot = TRUE, step_wise = TRUE, score_card = TRUE,
  cor_p = 0.8, iv_i = 0.02, psi_i = 0.1, ...)
```

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
best_lambda	Metheds of best lanmbda stardards using to filter variables by LASSO.There are four methods: ("lambda.min", "lambda.1se", "lambda.05se" , "lambda.sim_sign") . Default is "lambda.sim_sign". See details at: get_best_lambda
sp_values	Vaules will be in separate bins.e.g. list(-1, "Unknown") means that -1 & Unknown 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 diffrent 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 good obs and 500 bad obs before oversampling or under-sampling, 5,000 good obs and 3,000 bad obs after oversampling. Then this parameter should be set to c(10000/5000, 500/3000). Default is NULL..
lasso	Logical, if TRUE, variables filtering by LASSO. Default is TRUE.
vars_plot	Logical, if TRUE, plot distribution and correlation of input variables . 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.
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.
...	Other parameters

Value

A list of parameters.

See Also

training_model, xgb_params, gbm_params, rf_params

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

```
merge_category(dat, ex_cols = "date$|id$|time$|DATA$|ID$|TIME$",
  p = 0.01, m = 10, note = FALSE)
```

Arguments

dat	A data frame with x and target.
ex_cols	A list of excluded variables. Default is NULL.
p	The minimum percent of samples in a category to merge.
m	The minimum number of categories.
note	Logical, outputs info. Default is TRUE.

Value

A data.frame with merged category variables.

Examples

```
#merge_catagory
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])
```

min_max_norm *Min Max Normalization*

Description

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

Usage

```
min_max_norm(x)
```

Arguments

x	Vector

Value

Normalized vector

Examples

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

---

null_blank_na *Encode NAs*

Description

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

Usage

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

Arguments

dat	A data frame with x and target.
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 "Unknown".
note	Logical.Outputs info.Default is TRUE.

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

dat	A dat frame.
cat_vars	The name or Column index list to be one_hot encoded.
ex_cols	Variables to be excluded, use regular expression matching
merge_cat	Logical. If TRUE, to merge categories greater than 8, default is TRUE.
na_act	Logical,If true, the missing value is processed, if FALSE missing value is omitted .
note	Logical.Outputs info.Default is TRUE.

Value

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

See Also

de_one_hot_encoding

Examples

```
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)
```

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

Description

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

Usage

```
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.

PCA_reduce *PCA Dimension Reduction*

Description

PCA_reduce is used for PCA reduction of high demension 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

```
## 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_theme *plot_theme*

Description

plot_theme is a simper wrapper of theme for ggplot2.

Usage

```
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")
```

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

process_nas *Missing Values Treatment*

Description

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

Usage

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

process_nas_var(dat = dat, x, default_miss = TRUE, 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.
default_miss	Logical. If TRUE, assigning the missing values to -1 or "Unknown", otherwise ,processing the missing values according to the results of missing analysis.
class_var	Logical, nas analysis of the nominal variables. Default is TRUE.
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."median" is knn imputation by k neighbors median.
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 dat frame with no NAs.

Examples

```
dat_na = process_nas(dat = UCICreditCard[1:1000,], default_miss = FALSE,
target = "default.payment.next.month",
parallel = FALSE,ex_cols = "ID$" ,method = "median")
```

---

process_outliers *Outliers Treatment*

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

```
process_outliers(dat, target, ex_cols = NULL, kc = 3, kn = 5,
  x_list = NULL, parallel = FALSE, note = FALSE, process = TRUE,
  save_data = FALSE, file_name = NULL, dir_path = tempdir())

outliers_kmeans_lof(dat, x, target = NULL, kc = 3, kn = 5,
  note = FALSE, process = TRUE, save_data = FALSE,
  file_name = NULL, dir_path = tempdir())
```

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.

Value

A data frame with outliers process to all the variables.

Examples

```
dat_out = process_outliers(UCICreditCard,
                           target = "default.payment.next.month",
                           ex_cols = "date$", kc = 3, kn = 10, parallel = FALSE)
```

`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, oot_pct = 0.7, psi_i = 0.1, iv_i = 0.01,
  vars_name = FALSE, note = FALSE, 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.
`oot_pct`	Percentage of observations retained for overtime test (especially to calculate PSI). Defualt 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

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 "Featrue_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.

See Also

xgb_filter, gbm_filter, feature_select_wrapper

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)
```

---

quick_as_df *List as data.frame quickly*

Description

quick_as_df is function for fast dat frame transfromation.

Usage

```
quick_as_df(df_list)
```

Arguments

df_list	A list of data.

Value

packages installed and library,

Examples

```
UCICreditCard = quick_as_df(UCICreditCard)
```

`reduce_high_cor` *Compare the two highly correlated variables*

Description

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

Usage

```
reduce_high_cor(cor_mat, p = 0.9, 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.

`remove_duplicated` *Remove Duplicated Observations*

Description

`remove_duplicated` is the function to remove duplicated observations

Usage

```
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

```
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

```
require_packages(pkg)
```

Arguments

pkg	A list or vector of names of required packages.

Value

packages installed and library.

Examples

```
## Not run:
require_packages(c("dplyr","ggplot2"))

## End(Not run)
```

re_name *Rename*

Description

re_name is for renaming variables.

Usage

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

Arguments

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

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

---

rowAny *Functions for vector operation.*

Description

Functions for vector operation.

Usage

```
rowAny(x)

rowAllnas(x)

colAllnas(x)

colAllzeros(x)

rowAll(x)

rowCVs(x, na.rm = FALSE)

rowSds(x, na.rm = FALSE)
```

```
colSds(x, na.rm = FALSE)

rowMaxs(x, na.rm = FALSE)

rowMins(x, na.rm = FALSE)

rowMaxMins(x, na.rm = FALSE)
```

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])
```

---

save_dt *Save data*

Description

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

Usage

```
save_dt(dt, file_name = NULL, dir_path = tempdir(), note = FALSE,
  as_list = FALSE, row_names = FALSE, append = FALSE)
```

Arguments

dt	A data frame or list.
file_name	A string. The file name to save breaks_list.
dir_path	A string. The dir path to save breaks_list.
note	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

```
save_dt(UCICreditCard,"UCICreditCard", tempdir())
```

---

score_transfer *Scoring*

Description

score_transfer is for transfer woe to score.

Usage

```
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 TRUE

Value

A data.frame with variables which values transfered to score.

Examples

```
# dataset spliting
sub = cv_split(UCICreditCard, k = 30)[[1]]
dat = UCICreditCard[sub,]
#rename the target variable
dat = re_name(dat, "default.payment.next.month", "target")
dat = cleaning_data(dat, target = "target", obs_id = "ID",
occur_time = "apply_date", miss_values =  list("", -1))
#train_ test pliting
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",
                                occur_time = "apply_date",
                                 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"]
```

`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.
x	The name of variable to process.
target	The name of target variable.
breaks	Splitting points for an independent variable. Default is NULL.
occur_time	The name of the variable that represents the time at which each observation takes place.
oot_pct	The percentage of Actual and Expected set for PSI calculating.
pos_flag	The value of positive class of target variable, default: "1".
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_gb The maximum threshold of G/B index in any bins. 0 < b_gb < 1 ; 0.05 to 0.3 usually work. Default: 0.15.
- gb_psi The maximum threshold of Training and Testing G/B index PSI in any bins. 0 < gb_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.

sp_values	A list of special value.
...	Other parameters.

Details

The folloiwing 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 0.03 or more.
- 7.The difference of absolute odds ratio between intervals should be at least 0.1 or more.
- 8.The difference of bad rate between intervals should be at least 1/10 of the total bad 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 0.05.

Value

A list of breaks for x.

See Also

get_tree_breaks, cut_equal, get_breaks

Examples

```
#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_gb = 0.15, mono = 0.3, gb_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)
```

---

sim_str *sim_str*

Description

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

Usage

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

Arguments

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

---

split_bins *split_bins*

Description

split_bins is for binning using breaks.

Usage

```
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

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

`start_parallel_computing` *Parallel computing and export variables to global Env.*

Description

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

Usage

```
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

```
stop_parallel_computing(cluster)
```

Arguments

`cluster`	Parallel works.

Value

stop clusters.

time_transfer *Time Format Transfering*

Description

time_transfer is for transfering time variables to time format.

Usage

```
time_transfer(dat, date_cols = "DATE$|time$|date$|timestamp$|stamp$",
  ex_cols = NULL, note = 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 transfermed time variables.

Examples

```
#transfer a variable.
dat = time_transfer(dat = lendingclub,date_cols = "issue_d")
class(dat[,"issue_d"])
#transfer a group of variables with similar name.
dat = time_transfer(dat = lendingclub,date_cols = "_d$")
class(dat[,"issue_d"])
#transfer all time variables.
dat = time_transfer(dat = lendingclub,date_cols = NULL)
class(dat[,"issue_d"])
```

time_varieble *time_varieble*

Description

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

Usage

```
time_varieble(dat, date_cols = NULL, enddate = NULL)
```

Arguments

dat	A data.frame.
date_cols	Time variables.
enddate	End time.

---

time_vars_process *Processing of Time or Date Variables*

Description

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

Usage

```
time_vars_process(df_tm = df_tm, x, enddate = "occur_time")
```

Arguments

df_tm	A data.frame
x	Time variable.
enddate	End time.

---

training_model *Training model*

Description

`training_model` Model builder

Usage

```
training_model(model_name = "mymodel", dat_train, 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,
  miss_values = NULL, outlier_proc = TRUE, missing_proc = TRUE,
  default_miss = 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"),
  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, model_path = tempdir(),
  seed = 46, ...)
```

Arguments

model_name	A string, name of the project. Default is "mymodel"
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.
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
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 "Unknown".
outlier_proc	Logical. If TRUE, Outliers processing using Kmeans and Local Outlier Factor. Default is TRUE
missing_proc	Logical. If TRUE, missing value analysis and process missing value by knn imputation or central impulation or random imputation. Default is TRUE
default_miss	Logical. If TRUE, assigning the missing values to -1 or "Unknown", otherwise, processing the missing values according to the results of missing analysis. See details at: process_nas
feature_filter	Parameters for selecting important and stable features.See details at: feature_select_wrapper
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.

- 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
- best_lambda Metheds of best lanmbda stardards using to filter variables by LASSO.There are four methods: ("lambda.min", "lambda.1se", "lambda.05se" , "lambda.sim_sign") . Default is "lambda.sim_sign". See details at: get_best_lambda
- 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 diffrent 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 good obs and 500 bad obs before oversampling or under-sampling, 5,000 good obs and 3,000 bad obs after oversampling. Then this parameter should be set to c(10000/5000, 500/3000). Default is NULL..

- forced_inNames of forced input variables. Default is NULL.
- sp_values Vaules will be in separate bins.e.g. list(-1, "Unknown") means that -1 & Unknown as special values.Default is NULL.
- 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.
- cor_p The maximum threshold of correlation.0 <= cor_p <=1; 0.5 to 0.8 usually work. Default: 0.7.
- iv_i The minimum threshold of IV. 0 < iv_i ; 0.01 to 0.1 usually work. Default: 0.01
- psi_i The maximum threshold of PSI. 0 <= psi_i <=1; 0.05 to 0.2 usually work. Default: 0.1

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.
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,cleaning_data, feature_select_wrapper, 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 = 40)[[1]]
dat = UCICreditCard[sub,]
dat = re_name(dat, "default.payment.next.month", "target")
dat = cleaning_data(dat, target = "target", obs_id = "ID",
occur_time = "apply_date", miss_values = list("", -1, -2))
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")
B_model = training_model(dat_train = dat_train,
model_name = "UCICreditCard", target = "target", x_list = x_list,
occur_time = "apply_date", obs_id = "ID", dat_test = dat_test,
                           preproc = FALSE,
                           feature_filter = NULL,
                           algorithm = list("LR"),
                           LR.params = lr_params(lasso = FALSE,
                           step_wise = FALSE, vars_plot = FALSE),
                           XGB.params = xgb_params(),
                           breaks_list = NULL,
                           parallel = FALSE, cores_num = NULL,
                           save_pmml = FALSE, plot_show = FALSE,
                           model_path = tempdir(),
                           seed = 46)
```

`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 = c("Random", "OOT",
  "byRow"), 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 spliting data sets, e.g. : spliting 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
```

UCICreditCard *UCI Credit Card 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: mount 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

`variable_process` *variable_process*

Description

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

Usage

```
variable_process(add)
```

Arguments

add	A data.frame

vintage_function *vintage_function* vintage_function *is for vintage analysis.*

Description

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

Usage

```
vintage_function(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

dat	A data.frame contained id, occur_time, mob, status ...
obs_id	The name of ID of observations or key variable of data. Default is NULL.
occur_time	The name of the variable that represents the time at which each observation takes place.
MOB	Mobility of book
period	Period of event to analysiss. Default is "monthly"
status	Status of observations
amount	The name of variable representing amount. Default is NULL.
by_out	Output: amount (amt) or count (cnt)
start_date	The earliest occurrence time of observations.
end_date	The latest occurrence time of observations.
dead_status	Status of dead observations.

woe_trans_all *Converting data to WOE*

Description

woe_trans is for converting dat to woe. The woe_trans_all function is a simpler wrapper for woe_trans.

Usage

```
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(), ...)

woe_trans(dat, x, bins_table = NULL, target = NULL,
  breaks_list = NULL, woe_name = TRUE)
```

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.
bins_table	A table contians woe of each bin of variables, it is generated by codeget_bins_table_all,codeget_bins_table
target	The name of target variable. Default is NULL.
breaks_list	A list contains breaks of variables. it is generated by codeget_breaks_all,codeget_breaks
note	Logical, outputs info. Default is TRUE.
save_data	Logical, save results in locally specified folder. Default is TRUE
parallel	Logical, parallel computing. Default is FALSE.
woe_name	Logical. Add "_woe" at the end of the variable name.
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.
x	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 = cleaning_data(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,
```

```
                          woe_name = FALSE)
test_woe = woe_trans_all(dat = dat_test,
                       target = "target",
                         breaks_list = breaks_list,
                         note = FALSE)
```

---

xgb_filter *Select Features using XGB*

Description

xgb_filter is for selecting important features using xgboost.

Usage

```
xgb_filter(dat_train, dat_test = NULL, target = "flag",
  pos_flag = NULL, x_list = NULL, occur_time = NULL,
  ex_cols = NULL, xgb_params = list(nrounds = 1000, max.depth = 6, eta
  = 0.1, min_child_weight = 1, subsample = 1, colsample_bytree = 1, gamma =
  0, max_delta_step = 0, early_stopping_rounds = 100, eval_metric = "auc",
  objective = "binary:logistic"), cv_folds = 3, cp = NULL, seed = 46,
  vars_name = TRUE, note = FALSE, 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.
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.
xgb_params	Parameters of xgboost.The complete list of parameters is available at: http://xgboost.readthedocs.io/en/latest/parameter.html.
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 TRUE
file_name	The name for periodically saved results files. Default is "Featrue_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_select_wrapper

Examples

```
xgb_filter(dat_train = UCICreditCard[1:1000,c(2,4,8:9,26)], dat_test = NULL,
target = "default.payment.next.month", occur_time = "apply_date",cv_folds = 1,
ex_cols = "ID$|date$|default.payment.next.month$", vars_name = FALSE)
```

---

xgb_params *Logistic Regression & Scorecard Parameters*

Description

xgb_params is the list of parameters to train a LR model or Scorecard using in training_model.

Usage

```
xgb_params(nrounds = 1000, params = list(max.depth = 6, eta = 0.1,
  min_child_weight = 1, subsample = 1, colsample_bytree = 1, gamma = 0,
  max_delta_step = 0, eval_metric = "auc", objective = "binary:logistic"),
  early_stopping_rounds = 100, ...)
```

Arguments

nrounds	Max number of boosting iterations.
params	A list contains parameters of xgboost.The complete list of parameters is available at: http://xgboost.readthedocs.io/en/latest/parameter.html
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.
...	Other parameters

Value

A list of parameters.

See Also

training_model, lr_params,gbm_params, rf_params

`%alike%` *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