Package 'PDtoolkit'

October 12, 2022

Title Collection of Tools for PD Rating Model Development and
Validation

Version 0.4.0

Maintainer Andrija Djurovic `<djandrija@gmail.com>`

Description The goal of this package is to cover the most common steps in probability of default (PD) rating model development and validation.
The main procedures available are those that refer to univariate, bivariate, multivariate analysis, calibration and validation.
Along with accompanied 'monobin' and 'monobinShiny' packages, 'PDtoolkit' provides functions which are suitable for different
data transformation and modeling tasks such as:
imputations, monotonic binning of numeric risk factors, binning of categorical risk factors, weights of evidence (WoE) and
information value (IV) calculations, WoE coding (replacement of risk factors modalities with WoE values), risk factor clustering,
area under curve (AUC) calculation and others. Additionally, package provides set of validation functions for testing homogeneity,
heterogeneity, discriminatory and predictive power of the model.

License GPL (>= 3)

URL <https://github.com/andrija-djurovic/PDtoolkit>

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

Depends monobin, R (>= 2.10)

Imports dplyr, rpart

NeedsCompilation no

Author Andrija Djurovic [aut, cre]

Repository CRAN

Date/Publication 2022-06-06 18:10:05 UTC

R **topics documented:**

---

auc.model   *Area under curve (AUC)*

Description

auc.model calculates area under curve (AUC) for a given predicted values and observed target variable.

Usage

```
auc.model(predictions, observed)
```

Arguments

predictions	Model predictions.
observed	Observed values of target variable.

Value

The command `auc.model` returns value of AUC.

See Also

[bivariate](#) for automatic bivariate analysis.

Examples

```
suppressMessages(library(PDtoolkit))
data(gcd)
#categorize numeric risk factor
gcd$maturity.bin <- ndr.bin(x = gcd$maturity, y = gcd$qual, y.type = "bina")[[2]]
#estimate simple logistic regression model
lr <- glm(qual ~ maturity.bin, family = "binomial", data = gcd)
#calculate auc
auc.model(predictions = predict(lr, type = "response", newdata = gcd),
    observed = gcd$qual)
```

---

bivariate	*Bivariate analysis*

Description

`bivariate` returns the bivariate statistics for risk factors supplied in data frame db.
Implemented procedure expects all risk factors to be categorical, thus numeric risk factors should
be first categorized. Additionally, maximum number of groups per risk factor is set to 10, so risk
factors with more than 10 categories will not be processed automatically, but manual inspection
can be still done using `woe.tbl` and `auc.model` functions in order to produce the same statistics.
Results of both checks (risk factor class and number of categories), if identified, will be reported in
second element of function output - `info` data frame.
Bivariate report (first element of function output - `results` data frame) includes:

- rf: Risk factor name.
- bin: Risk factor group (bin).
- no: Number of observations per bin.
- ng: Number of good cases (where target is equal to 0) per bin.
- nb: Number of bad cases (where target is equal to 1) per bin.
- pct.o: Percentage of observations per bin.
- pct.g: Percentage of good cases (where target is equal to 0) per bin.

- pct.b: Percentage of bad cases (where target is equal to 1) per bin.
- dr: Default rate per bin.
- so: Number of all observations.
- sg: Number of all good cases.
- sb: Number of all bad cases.
- dist.g: Distribution of good cases per bin.
- dist.b: Distribution of bad cases per bin.
- woe: WoE value.
- iv.b: Information value per bin.
- iv.s: Information value of risk factor (sum of individual bins' information values).
- auc: Area under curve of simple logistic regression model estimated as y ~ x, where y is selected target variable and x is categorical risk factor.

Additional info report (second element of function output - `info` data frame), if produced, includes:

- rf: Risk factor name.
- reason.code: Reason code takes value 1 if inappropriate class of risk factor is identified, while for check of maximum number of categories it takes value 2.
- comment: Reason description.

Usage

```
bivariate(db, target)
```

Arguments

db	Data frame of risk factors and target variable supplied for bivariate analysis.
target	Name of target variable within db argument.

Value

The command `bivariate` returns the list of two data frames. The first one contains bivariate metrics while the second data frame reports results of above explained validations (class of the risk factors and number of categories).

See Also

[woe.tbl](#) and [auc.model](#) for manual bivariate analysis.

Examples

```
suppressMessages(library(PDtoolkit))
data(gcd)
#categorize numeric risk factors
gcd$age.bin <- ndr.bin(x = gcd$age, y = gcd$qual)[[2]]
gcd$age.bin.1 <- cut(x = gcd$age, breaks = 20)
gcd$maturity.bin <- ndr.bin(x = gcd$maturity, y = gcd$qual, y.type = "bina")[[2]]
```

```
gcd$amount.bin <- ndr.bin(x = gcd$amount, y = gcd$qual)[[2]]
str(gcd)
#select target variable and categorized risk factors
gcd.bin <- gcd[, c("qual", "age.bin", "maturity.bin", "amount.bin")]
#run bivariate analysis on data frame with only categorical risk factors
bivariate(db = gcd.bin, target = "qual")
#run bivariate analysis on data frame with mixed risk factors (categorical and numeric).
#for this example info table is produced
bivariate(db = gcd, target = "qual")
#run woe table for risk factor with more than 10 modalities
woe.tbl(tbl = gcd, x = "age.bin.1", y = "qual")
#calculate auc for risk factor with more than 10 modalities
lr <- glm(qual ~ age.bin.1, family = "binomial", data = gcd)
auc.model(predictions = predict(lr, type = "response", newdata = gcd),
    observed = gcd$qual)
```

---

boots.vld	*Bootstrap model validation*

Description

boots.vld performs bootstrap model validation. The goal of this procedure is to generate main
model performance metrics such as absolute mean square error, root mean square error or area
under curve (AUC) based on resampling method.

Usage

```
boots.vld(model, B = 1000, seed = 1122)
```

Arguments

model	Model in use, an object of class inheriting from "glm".
B	Number of bootstrap samples. Default is set to 1000.
seed	Random seed needed for ensuring the result reproducibility. Default is 1122.

Value

The command boots.vld returns a list of two objects.
The first object (iter), returns iteration performance metrics.
The second object (summary), is the data frame of iterations averages of performance metrics.

Examples

```
suppressMessages(library(PDtoolkit))
data(loans)
#run stepMIV
res <- stepMIV(start.model = Creditability ~ 1,
   miv.threshold = 0.02,
   m.ch.p.val = 0.05,
```

```
    coding = "WoE",
    db = loans)
#check output elements
names(res)
#extract the final model
final.model <- res$model
#print coefficients
summary(final.model)$coefficients
#print head of coded development data
head(res$dev.db)
#calculate AUC
auc.model(predictions = predict(final.model, type = "response", newdata = res$dev.db),
    observed = res$dev.db$Creditability)
boots.vld (model = final.model, B = 10, seed = 1122)
```

---

cat.bin   *Categorical risk factor binning*

Description

`cat.bin` implements three-stage binning procedure for categorical risk factors. The first stage is possible correction for minimum percentage of observations. The second stage is possible correction for target rate (default rate), while the third one is possible correction for maximum number of bins. Last stage implements procedure known as adjacent pooling algorithm (APA) which aims to minimize information loss while iterative merging of the bins.

Usage

```
cat.bin(
  x,
  y,
  sc = NA,
  sc.merge = "none",
  min.pct.obs = 0.05,
  min.avg.rate = 0.01,
  max.groups = NA,
  force.trend = "modalities"
)
```

Arguments

x	Categorical risk factor.
y	Numeric target vector (binary).
sc	Special case elements. Default value is NA.
sc.merge	Define how special cases will be treated. Available options are: "none", "first", "last", "closest". If "none" is selected, then the special cases will be kept in separate bin. If "first" or "last" is selected, then the

special cases will be merged with first or last bin. Depending on sorting option `force.trend`, first or last bin will be determined based on alphabetic order (if `force.trend` is selected as `"modalities"`) or on minimum or maximum default rate (if `force.trend` is selected as `"dr"`). If `"closest"` is selected, then the special case will be merged with the bin that is closest based on default rate. Merging of the special cases with other bins is performed at the beginning i.e. before running any of three-stage procedures. Default value is `"none"`.

min.pct.obs   Minimum percentage of observations per bin. Default is 0.05 or minimum 30 observations.

min.avg.rate   Minimum default rate. Default is 0.01 or minimum 1 bad case for y 0/1.

max.groups   Maximum number of bins (groups) allowed for analyzed risk factor. If in the first two stages number of bins is less or equal to selected `max.groups` or if `max.groups` is default value (NA), no adjustment is performed. Otherwise, APA algorithm is applied which minimize information loss in further iterative process of bin merging.

force.trend   Defines how initial summary table will be ordered. Possible options are: `"modalities"` and `"dr"`. If `"modalities"` is selected, then merging will be performed forward based on alphabetic order of risk factor modalities. On the other hand, if `"dr"` is selected, then bins merging will be performed forward based on increasing order of default rate per modality. This direction of merging is applied in the all three stages.

Value

The command `cat.bin` generates a list of two objects. The first object, data frame `summary.tbl` presents a summary table of final binning, while `x.trans` is a vector of new grouping values.

References

Anderson, R. (2007). The credit scoring toolkit: theory and practice for retail credit risk management and decision automation, Oxford University Press

Examples

```
suppressMessages(library(PDtoolkit))
data(loans)
#prepare risk factor Purpose for the analysis
loans$Purpose <- ifelse(nchar(loans$Purpose) == 2, loans$Purpose, paste0("0", loans$Purpose))
#artificially add missing values in order to show functions' features
loans$Purpose[1:6] <- NA
#run binning procedure
res <- cat.bin(x = loans$Purpose,
   y = loans$Creditability,
   sc = NA,
   sc.merge = "none",
   min.pct.obs = 0.05,
   min.avg.rate = 0.05,
   max.groups = NA,
   force.trend = "modalities")
```

```
res[[1]]
#check new risk factor against the original
table(loans$Purpose, res[[2]], useNA = "always")
#repeat the same process with setting max.groups to 4 and force.trend to dr
res <- cat.bin(x = loans$Purpose,
   y = loans$Creditability,
   sc = NA,
   sc.merge = "none",
   min.pct.obs = 0.05,
   min.avg.rate = 0.05,
   max.groups = 4,
   force.trend = "dr")
res[[1]]
#check new risk factor against the original
table(loans$Purpose, res[[2]], useNA = "always")
#example of shrinking number of groups for numeric risk factor
#copy exisitng numeric risk factor to new called maturity
loans$maturity <- loans$"Duration of Credit (month)"
#artificially add missing values in order to show functions' features
loans$maturity[1:10] <- NA
#categorize maturity with MAPA algorithim from monobin package
loans$maturity.bin <- cum.bin(x = loans$maturity,
y = loans$Creditability, g = 50)[[2]]
table(loans$maturity.bin)
#run binning procedure to decrease number of bins from the previous step
res <- cat.bin(x = loans$maturity.bin,
   y = loans$Creditability,
   sc = "SC",
   sc.merge = "closest",
   min.pct.obs = 0.05,
   min.avg.rate = 0.01,
   max.groups = 5,
   force.trend = "modalities")
res[[1]]
#check new risk factor against the original
table(loans$maturity.bin, res[[2]], useNA = "always")
```

---

create.partitions   *Create partitions (aka nested dummy variables)*

Description

create.partitions performs creation of partitions (aka nested dummy variables). Using directly into logistic regression, partitions provide insight into difference of log-odds of adjacent risk factor bins (groups). Adjacent bins are selected based on alphabetic order of analyzed risk factor modalities, therefore it is important to ensure that modality labels are defined in line with expected monotonicity or any other criterion that is considered while engineering the risk factors.

Usage

```
create.partitions(db)
```

Arguments

db   Data set of risk factors to be converted into partitions.

Value

The command `create.partitions` returns a list of two objects (data frames).
The first object (`partitions`), returns the data set with newly created nested dummy variables.
The second object (`info`), is the data frame that returns info on partition process. Set of quality checks are performed and reported if any of them observed. Two of them are of terminal nature i.e. if observed, risk factor is not processed further (less then two non-missing groups and more than 10 modalities) while the one provides only info (warning) as usually deviates from the main principles of risk factor processing (less than 5% of observations per bin).

References

Scallan, G. (2011). Class(ic) Scorecards: Selecting Characteristics and Attributes in Logistic Regression, Edinburgh Credit Scoring Conference, downloaded from here.

Examples

```
suppressMessages(library(PDtoolkit))
data(loans)
#identify numeric risk factors
num.rf <- sapply(loans, is.numeric)
num.rf <- names(num.rf)[!names(num.rf)%in%"Creditability" & num.rf]
#discretized numeric risk factors using ndr.bin from monobin package
loans[, num.rf] <- sapply(num.rf, function(x)
cum.bin(x = loans[, x], y = loans[, "Creditability"])[[2]])
str(loans)
loans.p <- create.partitions(db = loans[, num.rf])
head(loans.p[["partitions"]])
loans.p[["info"]]
#bring target to partitions
db.p <- cbind.data.frame(Creditability = loans$Creditability, loans.p[[1]])
#prepare risk factors for stepMIV
db.p[, -1] <- sapply(db.p[, -1], as.character)
#run stepMIV
res <- stepMIV(start.model = Creditability ~ 1,
   miv.threshold = 0.02,
   m.ch.p.val = 0.05,
   coding = "dummy",
   db = db.p)
#check output elements
names(res)
#extract the final model
final.model <- res$model
#print coefficients
summary(final.model)$coefficients
```

`dp.testing`   *Testing the discriminatory power of PD rating model*

Description

`dp.testing` performs testing of discriminatory power of the model in use applied to application portfolio in comparison to the discriminatory power from the moment of development. Testing is performed based on area under curve (AUC) from the application portfolio and development sample under assumption that latter is a deterministic (as given) and that test statistics follow the normal distribution. Standard error of AUC for application portfolio is calculated as proposed by Hanley and McNeil (see References).

Usage

```
dp.testing(app.port, def.ind, pdc, auc.test, alternative, alpha = 0.05)
```

Arguments

`app.port`	Application portfolio (data frame) which contains default indicator (0/1) and calibrated probabilities of default (PD) in use.
`def.ind`	Name of the column that represents observed default indicator (0/1).
`pdc`	Name of the column that represent calibrated PD in use.
`auc.test`	Value of tested AUC (usually AUC from development sample).
`alternative`	Alternative hypothesis. Available options are: `"less"`, `"greater"`, `"two.sided"`.
`alpha`	Significance level of p-value for hypothesis testing. Default is 0.05.

Details

Due to the fact that test of discriminatory power is usually implemented on the application portfolio, certain prerequisites are needed to be fulfilled. In the first place model should be developed and rating scale should be formed. In order to reflect appropriate role and right moment of tests application, presented simplified example covers all steps before test implementation.

Value

The command `dp.testing` returns a data frame with input parameters along with hypothesis testing metrics such as estimated difference of observed (application portfolio) and testing AUC, standard error of observed AUC, p-value of testing procedure and accepted hypothesis.

References

Hanley J. and McNeil B. (1982). The meaning and use of the area under a receiver operating characteristic (ROC) curve. Radiology (1982) 43 (1) pp. 29-36.

Examples

```
suppressMessages(library(PDtoolkit))
data(loans)
#estimate some dummy model
mod.frm <- `Creditability` ~ `Account Balance` + `Duration of Credit (month)` +
`Age (years)`
lr.mod <- glm(mod.frm, family = "binomial", data = loans)
summary(lr.mod)$coefficients
#model predictions
loans$pred <-  unname(predict(lr.mod, type = "response", newdata = loans))
#scale probabilities
loans$score <- scaled.score(probs = loans$pred, score = 600, odd = 50/1, pdo = 20)
#group scores into rating
loans$rating <- sts.bin(x = round(loans$score), y = loans$Creditability, y.type = "bina")[[2]]
#create rating scale
rs <- loans %>%
group_by(rating) %>%
summarise(no = n(),
    nb = sum(Creditability),
    ng = sum(1 - Creditability)) %>%
mutate(dr = nb / no)
rs
#calcualte portfolio default rate
sum(rs$dr * rs$no / sum(rs$no))
#calibrate rating scale to central tendency of 27% with minimum PD of 5%
ct <- 0.27
min.pd <- 0.05
rs$pd <- rs.calibration(rs = rs,
dr = "dr",
w = "no",
ct = ct,
min.pd = min.pd,
method = "log.odds.ab")
#check
rs
sum(rs$pd * rs$no / sum(rs$no))
#bring calibrated PDs to the development sample
loans <- merge(loans, rs, by = "rating", all.x = TRUE)
#calculate development AUC
auc.dev <- auc.model(predictions = loans$pd, observed = loans$Creditability)
auc.dev
#simulate some dummy application portfolio
set.seed(321)
app.port <- loans[sample(1:nrow(loans), 400), ]
#calculate application portfolio AUC
auc.app <- auc.model(predictions = app.port$pd, observed = app.port$Creditability)
auc.app
#test deterioration of descriminatory power measured by AUC
dp.testing(app.port = app.port,
    def.ind = "Creditability",
    pdc = "pd", auc.test = 0.7557,
    alternative = "less",
```

```
      alpha = 0.05)
```

---

embedded.blocks   *Embedded blocks regression*

Description

embedded.blocks performs blockwise regression where the predictions of each blocks' model is used as an risk factor for the model of the following block.

Usage

```
embedded.blocks(
  method,
  target,
  db,
  coding = "WoE",
  blocks,
  p.value = 0.05,
  miv.threshold = 0.02,
  m.ch.p.val = 0.05
)
```

Arguments

method	Regression method applied on each block. Available methods: "stepMIV", "stepFWD" or "stepRPC".
target	Name of target variable within db argument.
db	Modeling data with risk factors and target variable.
coding	Type of risk factor coding within the model. Available options are: "WoE" and "dummy". If "WoE" is selected, then modalities of the risk factors are replaced by WoE values, while for "dummy" option dummies (0/1) will be created for n-1 modalities where n is total number of modalities of analyzed risk factor.
blocks	Data frame with defined risk factor groups. It has to contain the following columns: rf and block.
p.value	Significance level of p-value for the estimated coefficient. For WoE coding this value is is directly compared to p-value of the estimated coefficient, while for dummy coding multiple Wald test is employed and its p-value is used for comparison with selected threshold (p.value). This argument is applicable only for "stepFWD" and "stepRPC" selected methods.
miv.threshold	MIV (marginal information value) entrance threshold applicable only for code"stepMIV" method. Only the risk factors with MIV higher than the threshold are candidate for the new model. Additional criteria is that MIV value should significantly separate good from bad cases measured by marginal chi-square test.
m.ch.p.val	Significance level of p-value for marginal chi-square test applicable only for code"stepMIV" method. This test additionally supports MIV value of candidate risk factor for final decision.

Value

The command embedded.blocks returns a list of three objects.
The first object (model) is the list of the models of each block (an object of class inheriting from
"glm").
The second object (steps), is the data frame with risk factors selected from the each block.
The third object (dev.db), returns the list of block's model development databases.

References

Anderson, R.A. (2021). Credit Intelligence & Modelling, Many Paths through the Forest of Credit
Rating and Scoring, OUP Oxford

See Also

staged.blocks, ensemble.blocks, stepMIV, stepFWD and stepRPC.

Examples

```
suppressMessages(library(PDtoolkit))
data(loans)
#identify numeric risk factors
num.rf <- sapply(loans, is.numeric)
num.rf <- names(num.rf)[!names(num.rf)%in%"Creditability" & num.rf]
#discretized numeric risk factors using ndr.bin from monobin package
loans[, num.rf] <- sapply(num.rf, function(x)
ndr.bin(x = loans[, x], y = loans[, "Creditability"])[[2]])
str(loans)
#create risk factor priority groups
rf.all <- names(loans)[-1]
set.seed(22)
blocks <- data.frame(rf = rf.all, block = sample(1:3, length(rf.all), rep = TRUE))
blocks <- blocks[order(blocks$block), ]
blocks
#method: stepFWD
res <- embedded.blocks(method = "stepFWD",
    target = "Creditability",
    db = loans,
    coding = "WoE",
    blocks = blocks,
    p.value = 0.05)
names(res)
nb <- length(res[["models"]])
res$models[[nb]]

auc.model(predictions = predict(res$models[[nb]], type = "response",
    newdata = res$dev.db[[nb]]),
      observed = res$dev.db[[nb]]$Creditability)
```

---

ensemble.blocks   *Ensemble blocks regression*

Description

`ensemble.blocks` performs blockwise regression where the predictions of each blocks' model are
integrated into a final model. The final model is estimated in the form of logistic regression with-
out any check of the estimated coefficients (e.g. statistical significance or sign of the estimated
coefficients).

Usage

```
ensemble.blocks(
  method,
  target,
  db,
  coding = "WoE",
  blocks,
  p.value = 0.05,
  miv.threshold = 0.02,
  m.ch.p.val = 0.05
)
```

Arguments

method	Regression method applied on each block. Available methods: `"stepMIV"`, `"stepFWD"` or `"stepRPC"`.
target	Name of target variable within db argument.
db	Modeling data with risk factors and target variable.
coding	Type of risk factor coding within the model. Available options are: `"WoE"` and `"dummy"`. If `"WoE"` is selected, then modalities of the risk factors are replaced by WoE values, while for `"dummy"` option dummies (0/1) will be created for `n-1` modalities where `n` is total number of modalities of analyzed risk factor.
blocks	Data frame with defined risk factor groups. It has to contain the following columns: `rf` and `block`.
p.value	Significance level of p-value for the estimated coefficient. For `WoE` coding this value is is directly compared to p-value of the estimated coefficient, while for dummy coding multiple Wald test is employed and its p-value is used for comparison with selected threshold (`p.value`). This argument is applicable only for `"stepFWD"` and `"stepRPC"` selected methods.
miv.threshold	MIV (marginal information value) entrance threshold applicable only for code`"stepMIV"` method. Only the risk factors with MIV higher than the threshold are candidate for the new model. Additional criteria is that MIV value should significantly separate good from bad cases measured by marginal chi-square test.

m.ch.p.val   Significance level of p-value for marginal chi-square test applicable only for
                  code"stepMIV" method. This test additionally supports MIV value of candidate
                  risk factor for final decision.

Value

The command `embeded.blocks` returns a list of three objects.
The first object (`model`) is the list of the models of each block (an object of class inheriting from
`"glm"`).
The second object (`steps`), is the data frame with risk factors selected from the each block.
The third object (`dev.db`), returns the list of block's model development databases.

References

Anderson, R.A. (2021). Credit Intelligence & Modelling, Many Paths through the Forest of Credit
Rating and Scoring, OUP Oxford

See Also

staged.blocks, embedded.blocks, stepMIV, stepFWD and stepRPC.

Examples

```
suppressMessages(library(PDtoolkit))
data(loans)
#identify numeric risk factors
num.rf <- sapply(loans, is.numeric)
num.rf <- names(num.rf)[!names(num.rf)%in%"Creditability" & num.rf]
#discretized numeric risk factors using ndr.bin from monobin package
loans[, num.rf] <- sapply(num.rf, function(x)
ndr.bin(x = loans[, x], y = loans[, "Creditability"])[[2]])
str(loans)
#create risk factor priority groups
rf.all <- names(loans)[-1]
set.seed(22)
blocks <- data.frame(rf = rf.all, block = sample(1:3, length(rf.all), rep = TRUE))
blocks <- blocks[order(blocks$block), ]
blocks
#method: stepRPC
res <- ensemble.blocks(method = "stepRPC",
     target = "Creditability",
     db = loans,
     coding = "dummy",
     blocks = blocks,
     p.value = 0.05)
names(res)
nb <- length(res[["models"]])
res$models[[nb]]
auc.model(predictions = predict(res$models[[nb]], type = "response",
    newdata = res$dev.db[[nb]]),
     observed = res$dev.db[[nb]]$Creditability)
```

---

evrs   *Modelling the Economic Value of Credit Rating System*

Description

evrs calculates the economic benefits of improved PD model based on increase of portfolio return. Implemented algorithm replicates the framework presented in the Reference under assumption that bank adopts continuous PD rating scale. Despite this assumption, results are almost identical for scenarios of base case portfolio from the Reference.

Usage

```
evrs(
  db,
  pd,
  benchmark,
  lgd,
  target,
  sigma = NA,
  r,
  elasticity,
  prob.to.leave.threshold,
  sim.num = 500,
  seed = 991
)
```

Arguments

db	Data frame with at least the following columns: default indicator (target), PDs of model in use, PDs of benchmark model and LGD values.
pd	Name of PD of model in use within db argument.
benchmark	Name of PD of benchmark model within db argument.
lgd	Name of LGD values within db argument.
target	Name of target (default indicator 0/1) within db argument.
sigma	Measurement error of model in use. If default value (NA) is passed, then measurement error is calculated as standard deviation of PD difference of model in use and benchmark model.
r	Risk-free rate.
elasticity	Elasticity parameter used to define customer churn in case of loan overpricing.
prob.to.leave.threshold	
	Threshold for customers' probability to leave in case of loan overpricing.
sim.num	Number of simulations. Default is 500.
seed	Random seed to ensure reproducibility. Default is 991.

Value

The command `evrs` returns a list of two elements. The first element is data frame `summary.tbl` and it provides simulation summary: number of simulations, number of successful simulations, population size (number of observations of supplied db data frame), measurement error, average churn value (number of customers that left the portfolio due to the overpricing), average return of simulated portfolios, return of benchmark portfolio and return difference (main result of the simulation). The second element is numeric vector of return averages of simulated portfolios.

References

Jankowitsch at al. (2007). Modelling the economic value of credit rating systems. Journal of Banking & Finance, Volume 31, Issue 1, doi: 10.1016/j.jbankfin.2006.01.003.

Examples

```
suppressMessages(library(PDtoolkit))
data(loans)
#simulate model in use
#discretized numeric risk factors using ndr.bin from monobin package
num.rf <- sapply(loans, is.numeric)
num.rf <- names(num.rf)[!names(num.rf)%in%"Creditability" & num.rf]
loans[, num.rf] <- sapply(num.rf, function(x)
  ndr.bin(x = loans[, x], y = loans[, "Creditability"])[[2]])
#bin all risk factors on max number of groups, pct of observations and defaults
cat.rf <- names(loans)[!names(loans)%in%"Creditability"]
loans[, cat.rf] <- sapply(cat.rf, function(x)
                    cat.bin(x = loans[, x],
                        y = loans[, "Creditability"],
                          sc = c("SC", NA),
                         sc.merge = "closest",
                          min.pct.obs = 0.05,
                          min.avg.rate = 0.01,
                          max.groups = 5,
                          force.trend = "dr")[[2]])
loans.woe <- replace.woe(db = loans, target = "Creditability")[[1]]
#estimate model
miu.formula <- Creditability ~ `Account Balance` +`Age (years)` + `Duration of Credit (month)` +
      `Value Savings/Stocks` + `Purpose`
miu <- glm(miu.formula, family = "binomial", data = loans.woe)
miu
miu.pd <- unname(predict(miu, type = "response", newdata = loans.woe))

#simulate benchmark model with interaction.transformer support
data(loans)
#define risk factors that will be used for interactions
rf.mat <- matrix(c(1, 2, 14, 2, 6, 7, 4, 5, 19, 1, 3, 14, 7, 8, 12,
 8, 13, 16, 4, 13, 14, 8, 11, 17, 6, 9, 11,
 4, 10, 11), nrow = 3, byrow = FALSE)
rf.l <- ncol(rf.mat)
it <- data.frame(matrix(NA, ncol = rf.l, nrow = nrow(loans)))
it <- cbind.data.frame(Creditability = loans$Creditability, it)
```

```
names(it)[-1] <- paste0("rf.inter", apply(rf.mat, 2, paste, collapse = "."))
for (i in 1:rf.l) {
it.l <- interaction.transformer(db = loans,
  rf = names(loans)[rf.mat[, i] + 1],
    target = "Creditability",
  min.pct.obs = 0.05,
    min.avg.rate = 0.01,
    max.depth = 3,
    monotonicity = TRUE,
    create.interaction.rf = TRUE)[[2]]
it[, i + 1] <- as.character(unname(c(it.l, recursive = TRUE)))
}
it.woe <- replace.woe(db = it, target = "Creditability")[[1]]
bnm <- glm(Creditability ~ ., family = "binomial", data = it.woe)
bnm
bnm.pd <- unname(predict(bnm, type = "response", newdata = it.woe))

#prepare data for evrs function
db <- data.frame("Creditability" = loans$Creditability,
    pd = miu.pd,
    pd.benchmark = bnm.pd,
    lgd = 0.75)
#calculate the difference in portfolio return between model in use the benchmark model
res <- evrs(db = db,
pd = "pd",
benchmark = "pd.benchmark",
lgd = "lgd",
target = "Creditability",
sigma = NA,
r = 0.03,
elasticity = 100,
prob.to.leave.threshold = 0.5,
sim.num = 500,
seed = 991)
names(res)
#print simulation summary table
res[["summary.tbl"]]
#portfolio return increase in case of using benchmark model
res[["summary.tbl"]][, "return.difference", drop = FALSE]
res[["summary.tbl"]]$return.difference
#summary of simulated returns
summary(res[["return.sim"]])
```

---

heterogeneity   *Testing heterogeneity of the PD rating model*

Description

heterogeneity performs heterogeneity testing of PD model based on the rating grades. This test
is usually applied on application portfolio, but it can be applied also on model development sample.

Usage

```
heterogeneity(app.port, def.ind, rating, alpha = 0.05)
```

Arguments

app.port	Application portfolio (data frame) which contains default indicator (0/1) and ratings in use.
def.ind	Name of the column that represents observed default indicator (0/1).
rating	Name of the column that represent rating grades in use.
alpha	Significance level of p-value for two proportion test. Default is 0.05.

Details

Testing procedure starts with summarizing the number of observations and defaults per rating grade. After that, two proportion test is applied on adjacent rating grades. Testing hypothesis is that default rate of grade i is less or greater than default rate of grade i − 1, where i takes the values from 2 to the number of unique grades. Direction of alternative hypothesis (less or greater) is determined automatically based on correlation direction of observed default on rating grades. Incomplete cases, identified based on default indicator (def.ind) and rating grade (rating) columns are excluded from the summary table and testing procedure. If identified, warning will be returned.

Value

The command heterogeneity returns a data frame with the following columns:

- rating: Unique values of rating grades from application portfolio.
- no: Number of complete observations.
- nb: Number of defaults (bad cases) in complete observations.
- p.val: Test p-value (two proportion test of adjacent rating grades).
- alpha: Selected significance level.
- res: Accepted hypothesis.

Examples

```
suppressMessages(library(PDtoolkit))
data(loans)
#estimate some dummy model
mod.frm <- `Creditability` ~ `Account Balance` + `Duration of Credit (month)` +
`Age (years)` + `Value Savings/Stocks`
lr.mod <- glm(mod.frm, family = "binomial", data = loans)
summary(lr.mod)$coefficients
#model predictions
loans$pred <-  unname(predict(lr.mod, type = "response", newdata = loans))
#scale probabilities
loans$score <- scaled.score(probs = loans$pred, score = 600, odd = 50/1, pdo = 20)
#group scores into ratings
loans$rating.1 <- sts.bin(x = round(loans$score), y = loans$Creditability, y.type = "bina")[[2]]
#group probabilities into ratings
```

```
loans$rating.2 <- sts.bin(x = round(loans$pred, 4), y = loans$Creditability, y.type = "bina")[[2]]
#simulate dummy application portfolio
set.seed(1984)
app.port <- loans[sample(1:nrow(loans), 400, rep = TRUE), ]
#run heterogeneity test on ratings based on the scaled score
#higher score lower default rate
heterogeneity(app.port = app.port,
   def.ind = "Creditability",
   rating = "rating.1",
   alpha = 0.05)
#run test on predicted default rate - direction of the test is changed
heterogeneity(app.port = app.port,
   def.ind = "Creditability",
   rating = "rating.2",
   alpha = 0.05)
```

---

homogeneity   *Testing homogeneity of the PD rating model*

Description

`homogeneity` performs homogeneity testing of PD model based on the rating grades and selected segment. This test is usually applied on application portfolio, but it can be applied also on model development sample. Additionally, this method requires higher number of observations per segment modalities within each rating in order to produce available results. For segments with less than 30 observations, test is not performed. If as a segment user selects numeric variable from the application portfolio, variable will be grouped in selected number of groups (argument `segment.num`).

Usage

```
homogeneity(app.port, def.ind, rating, segment, segment.num, alpha = 0.05)
```

Arguments

`app.port`	Application portfolio (data frame) which contains default indicator (0/1), ratings in use and variable used as a segment.
`def.ind`	Name of the column that represents observed default indicator (0/1).
`rating`	Name of the column that represent rating grades in use.
`segment`	Name of the column that represent testing segments. If it is of numeric type, than it is first grouped into `segment.num` of groups otherwise is it used as supplied.
`segment.num`	Number of groups used for numeric variables supplied as a segment. Only applicable if `segment` is of numeric type.
`alpha`	Significance level of p-value for two proportion test. Default is 0.05.

Details

Testing procedure is implemented for each rating separately comparing default rate from one segment modality to the default rate from the rest of segment modalities.

Value

The command `homogeneity` returns a data frame with the following columns:

- segment.var: Variable used as a segment.

- rating: Unique values of rating grades from application portfolio..

- segment.mod: Tested segment modality. Default rate from this segment is compared with default rate from the rest of the modalities within the each rating.

- no: Number of observations of the analyzed rating.

- nb: Number of defaults (bad cases) of the analyzed rating.

- no.segment: Number of observations of the analyzed segment modality.

- no.rest: Number of observations of the rest of the segment modalities.

- nb.segment: Number of defaults of the analyzed segment modality.

- nb.rest: Number of defaults of the rest of the segment modalities.

- p.val: Two proportion test (two sided) p-value.

- alpha: Selected significance level.

- res: Accepted hypothesis.

Examples

```
suppressMessages(library(PDtoolkit))
data(loans)
#estimate some dummy model
mod.frm <- `Creditability` ~ `Account Balance` + `Duration of Credit (month)` +
`Age (years)` + `Value Savings/Stocks` +
`Duration in Current address`
lr.mod <- glm(mod.frm, family = "binomial", data = loans)
summary(lr.mod)$coefficients
#model predictions
loans$pred <-  unname(predict(lr.mod, type = "response", newdata = loans))
#scale probabilities
loans$score <- scaled.score(probs = loans$pred, score = 600, odd = 50/1, pdo = 20)
#group scores into ratings
loans$rating <- ndr.bin(x = round(loans$score), y = loans$Creditability, y.type = "bina")[[2]]
#simulate dummy application portfolio (oversample loans data)
set.seed(2211)
app.port <- loans[sample(1:nrow(loans), 2500, rep = TRUE), ]
#run homogeneity test on ratings based on the Credit Amount segments
homogeneity(app.port = app.port,
def.ind = "Creditability",
rating = "rating",
segment = "Credit Amount",
segment.num = 4,
alpha = 0.05)
```

---

imp.outliers   *Imputation methods for outliers*

Description

imp.outliers replaces predefined quantum of the smallest and largest values by the less extreme values. This procedure is applicable only to the numeric risk factors.

Usage

```
imp.outliers(
  db,
  sc = c(NA, NaN, Inf),
  method = "iqr",
  range = 1.5,
  upper.pct = 0.95,
  lower.pct = 0.05
)
```

Arguments

db	Data frame of risk factors supplied for imputation.
sc	Vector of all special case elements. Default values are c(NA, NaN, Inf). Those values will be excluded from calculation of imputed value and replacements.
method	Imputation method. Available options are: "iqr" and "percentile". Method iqr performs identification of outliers by the method applied in boxplot 5-figures, while for percentile method user defines lower and upper limits for replacement. Default value is "iqr".
range	Determines how far the plot whiskers extend out from the box. If range is positive, the whiskers extend to the most extreme data point which is no more than range times the interquartile range from the box. A value of zero causes the whiskers to extend to the data extremes. Default range is set to is 1.5.
upper.pct	Upper limit for percentile method. All values above this limit will be replaced by the value identified at this percentile. Default value is set to 95th percentile (0.95). This parameter is used only if selected method is percentile.
lower.pct	Lower limit for percentile method. All values below this limit will be replaced by the value identified at this percentile. Default value is set to 5th percentile (0.05). This parameter is used only if selected method is percentile.

Value

This function returns list of two data frames. The first data frame contains analyzed risk factors with imputed values for outliers, while the second data frame presents the imputation report. Using the imputation report, for each risk factor, user can inspect imputed info (info), imputation method (imputation.method), imputed value (imputation.val.upper and imputation.val.lower), number of imputed observations (imputation.num.upper and imputation.num.lower).

Examples

```
suppressMessages(library(PDtoolkit))
data(gcd)
gcd$age[1:20] <- NA
gcd$age.bin <- ndr.bin(x = gcd$age, y = gcd$qual, sc.method = "separately", y.type = "bina")[[2]]
gcd$dummy1 <- NA
imput.res.1 <- imp.outliers(db = gcd[, -1],
      method = "iqr",
      range = 1.5)
#analyzed risk factors with imputed values
head(imput.res.1[[1]])
#imputation report
imput.res.1[[2]]
#percentile method
imput.res.2 <- imp.outliers(db = gcd[, -1],
      method = "percentile",
      upper.pct = 0.95,
      lower.pct = 0.05)
#analyzed risk factors with imputed values
head(imput.res.2[[1]])
#imputation report
imput.res.2[[2]]
```

---

imp.sc   *Imputation methods for special cases*

Description

`imp.sc` imputes value for special cases.

Usage

```
imp.sc(
  db,
  sc.all = c(NA, NaN, Inf),
  sc.replace = c(NA, NaN, Inf),
  method.num = "automatic",
  p.val = 0.05
)
```

Arguments

db	Data frame of risk factors supplied for imputation.
sc.all	Vector of all special case elements. Default values are c(NA, NaN, Inf).
sc.replace	Vector of special case element to be replaced. Default values are c(NA, NaN, Inf).
method.num	Imputation method for numeric risk factors. Available options are: "automatic", "mean", "median", "zero".

| p.val | Significance level of p-value for Pearson normality test. Applicable only if `method.num` is `automatic`. |

Value

This function returns list of two data frames. The first data frame contains analyzed risk factors with imputed values for special cases, while the second data frame presents the imputation report. Using the imputation report, for each risk factor, user can inspect imputed info (`info`), imputation method (`imputation.method`), imputed value (`imputed.value`), number of imputed observations (`imputation.num`) and imputed mode (`imputed.mode` - applicable only for categorical risk factors) for each risk factor.

Examples

```
suppressMessages(library(PDtoolkit))
data(gcd)
gcd$age[1:20] <- NA
gcd$age.bin <- ndr.bin(x = gcd$age, y = gcd$qual, sc.method = "separately", y.type = "bina")[[2]]
gcd$dummy1 <- NA
#select risk factors for which we want to impute missing values (NA)
db.imp <- gcd[, c("age", "age.bin", "dummy1")]
colSums(is.na(db.imp))
imput.res <- imp.sc(db = db.imp,
      method.num = "automatic",
      p.val = 0.05)
#analyzed risk factors with imputed values
head(imput.res[[1]])
#imputation report
imput.res[[2]]
```

---

interaction.transformer

Extract risk factors interaction from decision tree

Description

`interaction.transformer` extracts the interaction between supplied risk factors from decision tree. It implements customized decision tree algorithm that takes into account different conditions such as minimum percentage of observations and defaults in each node, maximum tree depth and monotonicity condition at each splitting node. Gini index is used as metric for node splitting .

Usage

```
interaction.transformer(
  db,
  rf,
  target,
  min.pct.obs,
```

```
    min.avg.rate,
    max.depth,
    monotonicity,
    create.interaction.rf
)
```

Arguments

db	Data frame of risk factors and target variable supplied for interaction extraction.
rf	Character vector of risk factor names on which decision tree is run.
target	Name of target variable (default indicator 0/1) within db argument.
min.pct.obs	Minimum percentage of observation in each leaf.
min.avg.rate	Minimum percentage of defaults in each leaf.
max.depth	Maximum number of splits.
monotonicity	Logical indicator. If TRUE, observed trend between risk factor and target will be preserved in splitting node.
create.interaction.rf	
	Logical indicator. If TRUE, second element of the output will be data frame with interaction modalities.

Value

The command `interaction.transformer` returns a list of two data frames. The first data frame provides the tree summary. The second data frame is a new risk factor extracted from decision tree.

Examples

```
suppressMessages(library(PDtoolkit))
data(loans)
#modify risk factors in order to show how the function works with missing values
loans$"Account Balance"[1:10] <- NA
loans$"Duration of Credit (month)"[c(13, 15)] <- NA
it <- interaction.transformer(db = loans,
 rf = c("Account Balance", "Duration of Credit (month)"),
 target = "Creditability",
 min.pct.obs = 0.05,
 min.avg.rate = 0.01,
 max.depth = 2,
   monotonicity = TRUE,
 create.interaction.rf = TRUE)
names(it)
it[["tree.info"]]
tail(it[["interaction"]])
table(it[["interaction"]][, "rf.inter"], useNA = "always")
```

| kfold.vld | *K-fold model cross-validation* |

Description

kfold.vld performs k-fold model cross-validation. The main goal of this procedure is to generate main model performance metrics such as absolute mean square error, root mean square error or area under curve (AUC) based on resampling method.

Usage

```
kfold.vld(model, k = 10, seed = 1984)
```

Arguments

model	Model in use, an object of class inheriting from "glm"
k	Number of folds. If k is equal or greater than the number of observations of modeling data frame, then validation procedure is equivalent to leave one out cross-validation (LOOCV) method. For LOOCV, AUC is not calculated. Default is set to 10.
seed	Random seed needed for ensuring the result reproducibility. Default is 1984.

Value

The command kfold.vld returns a list of two objects.
The first object (iter), returns iteration performance metrics.
The second object (summary), is the data frame of iterations averages of performance metrics.

Examples

```
suppressMessages(library(PDtoolkit))
data(loans)
#identify numeric risk factors
num.rf <- sapply(loans, is.numeric)
num.rf <- names(num.rf)[!names(num.rf)%in%"Creditability" & num.rf]
#discretized numeric risk factors using ndr.bin from monobin package
loans[, num.rf] <- sapply(num.rf, function(x)
ndr.bin(x = loans[, x], y = loans[, "Creditability"])[[2]])
str(loans)
#run stepMIV
res <- stepMIV(start.model = Creditability ~ 1,
   miv.threshold = 0.02,
   m.ch.p.val = 0.05,
   coding = "WoE",
   db = loans)
#check output elements
names(res)
#extract the final model
```

```
final.model <- res$model
#print coefficients
summary(final.model)$coefficients
#print head of coded development data
head(res$dev.db)
#calculate AUC
auc.model(predictions = predict(final.model, type = "response", newdata = res$dev.db),
    observed = res$dev.db$Creditability)
kfold.vld(model = final.model, k = 10, seed = 1984)
```

---

loans   *German Credit Data*

Description

The German Credit Data contains data on 20 variables and the classification whether an applicant is considered a Good or a Bad credit risk for 1000 loan applicants. Name of the columns are used as give in the source file. Note that subset of those data is available also in 'monobin' package (gcd) and used for some examples in 'PDtoolkit' package.

Usage

```
loans
```

Format

An object of class data.frame with 1000 rows and 21 columns.

Source

<https://online.stat.psu.edu/stat857/node/215/>

---

power   *Power of statistical tests for predictive ability testing*

Description

power performs Monte Carlo simulation of power of statistical test used for testing the predictive ability of the PD rating model. It covers fours tests: binomial, Jeffreys, z-score and Hosmer-Lemeshow test. This procedure is applied under assumption that observed default rate is the true one and it make sense to use it in order to check if calibrated PDs are underestimated. Therefore, for the cases where observed default rate is lower than calibrated PD, power calculation is not performed and will report the comment.

Usage

```
power(rating.label, pdc, no, nb, alpha = 0.05, sim.num = 1000, seed = 2211)
```

Arguments

`rating.label`	Vector of rating labels.
`pdc`	Vector of calibrated probabilities of default (PD).
`no`	Number of observations per rating grade.
`nb`	Number of defaults (bad cases) per rating grade.
`alpha`	Significance level of p-value for implemented tests. Default is 0.05.
`sim.num`	Number of Monte Carlo simulations. Default is 1000.
`seed`	Random seed needed for ensuring the result reproducibility. Default is 2211.

Details

Due to the fact that test of predictive power is usually implemented on the application portfolio, certain prerequisites are needed to be fulfilled. In the first place model should be developed and rating scale should be formed. In order to reflect appropriate role and right moment of tests application, presented simplified example covers all steps before test implementation.

Value

The command power returns a list with two objects. Both are the data frames and while the first one presents power calculation of the tests applied usually on the rating level (binomial, Jeffreys and z-score test), the second one presents results of the Hosmer-Lemeshow test which is applied on the complete rating scale. For both level of the implementation (rating or complete scale) if the observed default rate is less than calibrated PD, function will return the comment and power simulation will not be performed.

Examples

```
suppressMessages(library(PDtoolkit))
data(loans)
#estimate some dummy model
mod.frm <- `Creditability` ~ `Account Balance` + `Duration of Credit (month)` +
`Age (years)`
lr.mod <- glm(mod.frm, family = "binomial", data = loans)
summary(lr.mod)$coefficients
#model predictions
loans$pred <-  unname(predict(lr.mod, type = "response", newdata = loans))
#scale probabilities
loans$score <- scaled.score(probs = loans$pred, score = 600, odd = 50/1, pdo = 20)
#group scores into rating
loans$rating <- sts.bin(x = round(loans$score), y = loans$Creditability, y.type = "bina")[[2]]
#create rating scale
rs <- loans %>%
group_by(rating) %>%
summarise(no = n(),
    nb = sum(Creditability),
    ng = sum(1 - Creditability)) %>%
mutate(dr = nb / no)
rs
#calcualte portfolio default rate
```

```
sum(rs$dr * rs$no / sum(rs$no))
#calibrate rating scale to central tendency of 27% with minimum PD of 5%
ct <- 0.27
min.pd <- 0.05
rs$pd <- rs.calibration(rs = rs,
dr = "dr",
w = "no",
ct = ct,
min.pd = min.pd,
method = "log.odds.ab")
#check
rs
sum(rs$pd * rs$no / sum(rs$no))
#simulate some dummy application portfolio
set.seed(22)
app.port <- loans[sample(1:nrow(loans), 400), ]
#summarise application portfolio on rating level
ap.summary <- app.port %>%
  group_by(rating) %>%
  summarise(no = n(),
nb = sum(Creditability),
ng = sum(1 - Creditability)) %>%
  mutate(dr = nb / no)
#bring calibrated pd as a based for predictive power testing
ap.summary <- merge(rs[, c("rating", "pd")], ap.summary, by = "rating", all.x = TRUE)
ap.summary
#perform predictive power testing
pp.res <- pp.testing(rating.label = ap.summary$rating,
     pdc = ap.summary$pd,
     no = ap.summary$no,
     nb = ap.summary$nb,
     alpha = 0.05)
pp.res
power(rating.label = ap.summary$rating,
  pdc = ap.summary$pd,
  no = ap.summary$no,
  nb = ap.summary$nb,
  alpha = 0.05,
  sim.num = 1000,
  seed = 2211)
```

---

pp.testing   *Testing the predictive power of PD rating model*

Description

pp.testing performs testing of predictive power of the PD rating model. This procedure should
be applied on the level of the rating scale. Four tests are implemented: binomial, Jeffreys, z-score
and Hosmer-Lemeshow test. Only Hosmer-Lemeshow test refers to complete rating scale, while
the remaining three are implemented on the rating grade level. The null hypothesis for all tests is
that observed default rate $\frac{n_b}{n_o}$ is less or equal to the calibrated PD (pdc).

Usage

```
pp.testing(rating.label, pdc, no, nb, alpha = 0.05)
```

Arguments

rating.label	Vector of rating labels.
pdc	Vector of calibrated probabilities of default (PD).
no	Number of observations per rating grade.
nb	Number of defaults (bad cases) per rating grade.
alpha	Significance level of p-value for implemented tests. Default is 0.05.

Details

Due to the fact that test of predictive power is usually implemented on the application portfolio, certain prerequisites are needed to be fulfilled. In the first place model should be developed and rating scale should be formed. In order to reflect appropriate role and right moment of tests application, presented simplified example covers all steps before test implementation.

Value

The command pp.testing returns a data frame with input parameters along with p-value for each implemented test and the accepted hypothesis. Due to the fact that Hosmer-Lemeshow test is applied to complete rating scale, returned p-values are all equal between the rating grades as well as the test results.

References

Tasche, D. (2008). Validation of internal rating systems and PD estimates, The Analytics of Risk Model Validation, Quantitative Finance, Elsevier B.V., doi: 10.1016/B9780750681582.500147. Oesterreichische Nationalbank (2004). Rating Models and Validation, Oesterreichische Nationalbank (OeNB).

Examples

```
suppressMessages(library(PDtoolkit))
data(loans)
#estimate some dummy model
mod.frm <- `Creditability` ~ `Account Balance` + `Duration of Credit (month)` +
`Age (years)`
lr.mod <- glm(mod.frm, family = "binomial", data = loans)
summary(lr.mod)$coefficients
#model predictions
loans$pred <-  unname(predict(lr.mod, type = "response", newdata = loans))
#scale probabilities
loans$score <- scaled.score(probs = loans$pred, score = 600, odd = 50/1, pdo = 20)
#group scores into rating
loans$rating <- sts.bin(x = round(loans$score), y = loans$Creditability, y.type = "bina")[[2]]
#create rating scale
rs <- loans %>%
```

```
group_by(rating) %>%
summarise(no = n(),
    nb = sum(Creditability),
    ng = sum(1 - Creditability)) %>%
mutate(dr = nb / no)
rs
#calcualte portfolio default rate
sum(rs$dr * rs$no / sum(rs$no))
#calibrate rating scale to central tendency of 27% with minimum PD of 5%
ct <- 0.33
min.pd <- 0.05
rs$pd <- rs.calibration(rs = rs,
dr = "dr",
w = "no",
ct = ct,
min.pd = min.pd,
method = "log.odds.ab")
#checks
rs
sum(rs$pd * rs$no / sum(rs$no))
#simulate some dummy application portfolio
set.seed(11)
app.port <- loans[sample(1:nrow(loans), 400), ]
#summarise application portfolio on rating level
ap.summary <- app.port %>%
  group_by(rating) %>%
  summarise(no = n(),
nb = sum(Creditability),
ng = sum(1 - Creditability)) %>%
  mutate(dr = nb / no)
#bring calibrated pd as a based for predictive power testing
ap.summary <- merge(rs[, c("rating", "pd")], ap.summary, by = "rating", all.x = TRUE)
ap.summary
#perform predictive power testing
pp.res <- pp.testing(rating.label = ap.summary$rating,
    pdc = ap.summary$pd,
    no = ap.summary$no,
    nb = ap.summary$nb,
    alpha = 0.05)
pp.res
```

---

psi   *Population Stability Index (PSI)*

Description

psi calculates Population Stability Index (PSI) for a given base and target vectors. Function can be used for testing the stability of final model score, but also for testing a risk factor stability (aka Characteristic Stability Index). Function also provides so-called critical values of z-score (based on normal distribution assumption) and chi-square (based on Chi-square distribution) that can be used as alternatives for fixed "rule of thumb" thresholds (10% and 25%). For details see the Reference.

Usage

```
psi(base, target, bin = 10, alpha = 0.05)
```

Arguments

base   Vector of value from base sample. Usually this is training (model development)
                sample.

target   Vector of value from target sample. Usually this is testing or portfolio applica-
                tion sample.

bin   Number of bins. Applied only for numeric base and target and used for dis-
                cretization of its values. Default is 10.

alpha   Significance level used for calculation of statistical critical values (`cv.zscore`
                and `cv.chisq`). Default is 0.05, which refers to 0.95 confidence interval.

Value

The command `psi` returns a list of two data frames. The first data frame contains values of PSI
along with statistical critical values for confidence level of `1 - alpha`, while second data frame
presents summary table used for the calculation of overall PSI. For numeric base and target vectors,
summary table is presented on the bin (bucket level), while for the categorical modalities of base
and target vectors are tabulated.

References

Yurdakul, B. (2018). Statistical Properties of Population Stability Index . Dissertations. 3208.
downloaded from here

Examples

```
suppressMessages(library(PDtoolkit))
data(loans)
#split on training and testing data set
set.seed(1122)
tt.indx <- sample(1:nrow(loans), 700, replace = FALSE)
training <- loans[tt.indx, ]
testing <- loans[-tt.indx, ]
#calculate psi for numeric risk factor
psi(base = training[, "Age (years)"], target = testing[, "Age (years)"],
   bin = 10, alpha = 0.05)
#calculate psi for categorical risk factor
psi(base = training[, "Account Balance"], target = testing[, "Account Balance"],
   bin = 10, alpha = 0.05)
```

---

replace.woe	*Replace modalities of risk factor with weights of evidence (WoE) value*

Description

`replace.woe` replaces modalities of risk factor with calculated WoE value. This function process only categorical risk factors, thus it is assumed that numerical risk factors are previously categorized. Additional info report (second element of function output - `info` data frame), if produced, includes:

- rf: Risk factor name.
- reason.code: Reason code takes value 1 if inappropriate class of risk factor is identified. It takes value 2 if maximum number of categories exceeds 10, while 3 if there are any problem with weights of evidence (WoE) calculations (usually if any bin contains only good or bad cases). If validation 1 and 3 are observed, risk factor is not process for WoE replacement.
- comment: Reason description.

Usage

```
replace.woe(db, target)
```

Arguments

db	Data frame of categorical risk factors and target variable supplied for WoE coding.
target	Name of target variable within db argument..

Value

The command `replace.woe` returns the list of two data frames. The first one contains WoE replacement of analyzed risk factors' modalities, while the second data frame reports results of above mentioned validations regarding class of the risk factors, number of modalities and WoE calculation.

Examples

```
suppressMessages(library(PDtoolkit))
data(gcd)
#categorize numeric risk factor
gcd$maturity.bin <- ndr.bin(x = gcd$maturity, y = gcd$qual, y.type = "bina")[[2]]
gcd$amount.bin <- ndr.bin(x = gcd$amount, y = gcd$qual, y.type = "bina")[[2]]
gcd$age.bin <- ndr.bin(x = gcd$age, y = gcd$qual, y.type = "bina")[[2]]
head(gcd)
#replace modalities with WoE values
woe.rep <- replace.woe(db = gcd, target = "qual")
#results overview
head(woe.rep[[1]])
woe.rep[[2]]
```

---

rf.clustering   *Risk factor clustering*

Description

rf.clustering implements correlation based clustering of risk factors. Clustering procedure is base on hclust from stats package.

Usage

```
rf.clustering(db, metric, k = NA)
```

Arguments

db   Data frame of risk factors supplied for clustering analysis.

metric   Correlation metric used for distance calculation. Available options are:

- "raw pearson" - calculated distance as.dist(1 – cor(db, method = "pearson"));
- "raw spearman" - calculated distance as.dist(1 – cor(db, method = "spearman"));
- "common pearson" - calculated distance as.dist((1 – cor(db, method = "pearson")) / 2);
- "common spearman" - calculated distance as.dist((1 – cor(db, method = "spearman")) / 2);
- "absolute pearson" - calculated distance as.dist(1 – abs(cor(db, method = "pearson")));
- "absolute spearman" - calculated distance as.dist(1 – abs(cor(db, method = "spearman")));
- "sqrt pearson" - calculated distance as.dist(sqrt(1 – cor(db, method = "pearson")));
- "sqrt spearman" - calculated distance as.dist(sqrt(1 – cor(db, method = "spearman")));
- "x2y" - calculated distance as.dist(1 – dx2y(d = db)[[2]])).

x2y metric is proposed by Professor Rama Ramakrishnan and details can be found on this link. This metric is especially handy if analyst wants to perform clustering before any binning procedures and to decrease number of risk factors. Additionally, x2y algorithm process numerical and categorical risk factors at once and it is able to identify non-linear relationship between the pairs. Metric x2y is not symmetric with respect to inputs - x, y, therefore arithmetic average of values between xy and yx is used to produce the final value for each pair.

k   Number of clusters. If default value (NA) is passed, then automatic elbow method will be used to determine the optimal number of clusters, otherwise selected number of clusters will be used.

Value

The function `rf.clustering` returns a data frame with: risk factors, clusters assigned and distance to centroid (ordered from smallest to largest). The last column (distance to centroid) can be used for selection of one or more risk factors per cluster.

Examples

```
suppressMessages(library(PDtoolkit))
library(rpart)
data(loans)
#risk factors clustering using x2y metric
cr <- rf.clustering(db = loans[, -which(names(loans)%in%"Creditability")],
  metric = "x2y",
  k = 15)
cr
cr %>% group_by(clusters) %>%
 slice(which.min(dist.to.centroid))
#clustering using common spearman metric
#first we need to categorize numeric risk factors
num.rf <- sapply(loans, is.numeric)
num.rf <- names(num.rf)[!names(num.rf)%in%"Creditability" & num.rf]
loans[, num.rf] <- sapply(num.rf, function(x)
  sts.bin(x = loans[, x], y = loans[, "Creditability"])[[2]])
#replace woe in order to convert to all numeric factors
loans.woe <- replace.woe(db = loans, target = "Creditability")[[1]]
cr <- rf.clustering(db = loans.woe[, -which(names(loans.woe)%in%"Creditability")],
  metric = "common spearman",
  k = NA)
cr
#select one risk factor per cluster with min distance to centorid
cr %>% group_by(clusters) %>%
 slice(which.min(dist.to.centroid))
```

---

rs.calibration *Calibration of the rating scale*

Description

`rs.calibration` performs calibration of the observed default rates for a given rating scale.

Usage

```
rs.calibration(rs, dr, w, ct, min.pd, method)
```

Arguments

rs	Rating scale that contain observed default rate and weights used for optimization.
dr	Observed default rate per rating.

w	Weights, usually number of observations (clients/accounts) per rating.
ct	Value of central tendency to which calibration is performed.
min.pd	Minimum probability of default (PD) per rating, as constrain for calibration process.
method	Calibration method. Available options are "scaling", "log.odds.a", "log.odds.ab".

Details

Method "scaling" relies on linear rescaling of observed default rate. Rescaling factor is calculated as a ratio between ct and observed portfolio default rate. Method "log.odds.a" optimize intercept of logit transformation in a way that makes portfolio default rate equal to selected central tendency (ct). Method "log.odds.ab" optimize intercept and slope of logit transformation in a way that makes portfolio default rate equal to selected central tendency (ct). For respecting selected constrain of minimum PD (min.pd), two-stage iterative procedure is implemented. Additional constrain of maximum PD (100%) is also implemented.

Value

The command rs.calibration returns a vector of calibrated PDs.

Examples

```
suppressMessages(library(PDtoolkit))
data(loans)
#estimate some dummy model
mod.frm <- `Creditability` ~ `Account Balance` + `Duration of Credit (month)` +
`Age (years)`
lr.mod <- glm(mod.frm, family = "binomial", data = loans)
summary(lr.mod)$coefficients
#model predictions
loans$pred <-  unname(predict(lr.mod, type = "response", newdata = loans))
#scale probabilities
loans$score <- scaled.score(probs = loans$pred, score = 600, odd = 50/1, pdo = 20)
#group scores into rating
loans$rating <- sts.bin(x = round(loans$score), y = loans$Creditability, y.type = "bina")[[2]]
#create rating scale
rs <- loans %>%
group_by(rating) %>%
summarise(no = n(),
    nb = sum(Creditability),
    ng = sum(1 - Creditability)) %>%
mutate(dr = nb / no)
rs
#calcualte portfolio default rate
sum(rs$dr * rs$no / sum(rs$no))
#calibrate rating scale to central tendency of 27% with minimum PD of 5%
ct <- 0.33
min.pd <- 0.05
rs$pd.scaling <- rs.calibration(rs = rs,
  dr = "dr",
  w = "no",
```

```
   ct = ct,
   min.pd = min.pd,
   method = "scaling")
rs$pd.log.a <- rs.calibration(rs = rs,
dr = "dr",
w = "no",
ct = ct,
min.pd = min.pd,
method = "log.odds.a")
rs$pd.log.ab <- rs.calibration(rs = rs,
 dr = "dr",
 w = "no",
 ct = ct,
 min.pd = min.pd,
 method = "log.odds.ab")
#checks
rs
sum(rs$pd.scaling * rs$no / sum(rs$no))
sum(rs$pd.log.a * rs$no / sum(rs$no))
sum(rs$pd.log.ab * rs$no / sum(rs$no))
```

---

scaled.score   *Scaling the probabilities*

Description

scaled.score performs scaling of the probabilities for a certain set up. User has to select three parameters (score, odd, pdo), while the probabilities (probs) are usually predictions of the final model.

Usage

```
scaled.score(probs, score = 600, odd = 50/1, pdo = 20)
```

Arguments

probs	Model predicted probabilities of default.
score	Specific score for selected odd (for argument odd). Default is 600.
odd	Odd (good/bad) at specific score (for argument score). Default is 50/1.
pdo	Points for double the odds. Default is 20.

Value

The command scaled.score returns a vector of scaled scores.

References

Siddiqi, N. (2012). Credit Risk Scorecards: Developing and Implementing Intelligent Credit Scoring, John Wiley & Sons, Inc.

Examples

```
suppressMessages(library(PDtoolkit))
data(loans)
#run stepMIV
res <- stepMIV(start.model = Creditability ~ 1,
   miv.threshold = 0.02,
   m.ch.p.val = 0.05,
   coding = "WoE",
   db = loans)
final.model <- res$model
summary(final.model)$coefficients
#overview of development data base
head(res$dev.db)
#predict probabilities using the final model
loans$probs <- predict(final.model, type = "response", newdata = res$dev.db)
#scale probabilities to scores
loans$score <- scaled.score(probs = loans$probs, score = 600, odd = 50/1, pdo = 20)
#check AUC of probabilities and score
auc.model(predictions = loans$probs, observed = loans$Creditability)
auc.model(predictions = loans$score, observed = ifelse(loans$Creditability == 0, 1, 0))
#note that higher score indicates lower probability of default
```

---

segment.vld   *Model segment validation*

Description

segment.vld performs model segment validation based on residuals. The main goal of this procedure is to identify segments where model in use overestimates or underestimates the observed default rate. The procedure consists of a few steps. The first step is to calculate the model residuals (observed default indicator minus estimated probability). Then, on obtained residuals, the regression tree is fitted for segment identification. Finally, one proportion test is applied in order to test overestimation or underestimation of the observed default rate within these segments. Results of this validation can indicate omission of some important risk factor(s) or some specific sub-portfolio for which model performs worse than for the rest of the portfolio.

Usage

```
segment.vld(model, db, alpha = 0.05)
```

Arguments

model	Model in use, an object of class inheriting from "glm"
db	Modeling data with risk factors and target variable. Risk factors used for model development have to be of the same type (if WoE coding is used it has to be numeric with WoE values). Additionally, the rest of the risk factors (these that are supplied in db, but not used for model development) will be used for segment validation.
alpha	Significance level of p-value for one proportion test. Default is 0.05.

Value

The command segment.vld returns a list of three objects.
The first object (segment.model), returns regression tree results (rpart object).
The second object (segment.testing), is the data frame with segment overview and testing results.
The third object (segment.rules), is the data frame with average residual rate and rules for segment identification. This elements is returned, only if the segments are identified, otherwise it isNULL.

Examples

```
suppressMessages(library(PDtoolkit))
library(rpart)
data(loans)
#run stepMIV
res <- stepMIV(start.model = Creditability ~ 1,
   miv.threshold = 0.02,
   m.ch.p.val = 0.05,
   coding = "WoE",
   db = loans)
#check output elements
names(res)
#extract the final model
final.model <- res$model
#print coefficients
summary(final.model)$coefficients
#run segment validation procedure
seg.analysis <- segment.vld(model = final.model,
db = res$dev.db,
alpha = 0.05)
#check output elements
names(seg.analysis)
#print segment model - regression tree
seg.analysis$segment.model
#print segment summary and statistical testing
seg.analysis$segment.testing
#print segment identification rules
seg.analysis$segment.rules
```

---

staged.blocks   *Staged blocks regression*

Description

staged.blocks performs blockwise regression where the predictions of each blocks' model is used as an offset for the model of the following block.

Usage

```
staged.blocks(
  method,
```

```
  target,
  db,
  coding = "WoE",
  blocks,
  p.value = 0.05,
  miv.threshold = 0.02,
  m.ch.p.val = 0.05
)
```

Arguments

method	Regression method applied on each block. Available methods: "stepMIV", "stepFWD" or "stepRPC".
target	Name of target variable within db argument.
db	Modeling data with risk factors and target variable.
coding	Type of risk factor coding within the model. Available options are: "WoE" and "dummy". If "WoE" is selected, then modalities of the risk factors are replaced by WoE values, while for "dummy" option dummies (0/1) will be created for n-1 modalities where n is total number of modalities of analyzed risk factor.
blocks	Data frame with defined risk factor groups. It has to contain the following columns: rf and block.
p.value	Significance level of p-value for the estimated coefficient. For WoE coding this value is is directly compared to p-value of the estimated coefficient, while for dummy coding multiple Wald test is employed and its p-value is used for comparison with selected threshold (p.value). This argument is applicable only for "stepFWD" and "stepRPC" selected methods.
miv.threshold	MIV (marginal information value) entrance threshold applicable only for code"stepMIV" method. Only the risk factors with MIV higher than the threshold are candidate for the new model. Additional criteria is that MIV value should significantly separate good from bad cases measured by marginal chi-square test.
m.ch.p.val	Significance level of p-value for marginal chi-square test applicable only for code"stepMIV" method. This test additionally supports MIV value of candidate risk factor for final decision.

Value

The command staged.blocks returns a list of three objects.
The first object (model) is the list of the models of each block (an object of class inheriting from "glm").
The second object (steps), is the data frame with risk factors selected from the each block.
The third object (dev.db), returns the list of block's model development databases.

References

Anderson, R.A. (2021). Credit Intelligence & Modelling, Many Paths through the Forest of Credit Rating and Scoring, OUP Oxford

See Also

embedded.blocks, ensemble.blocks, stepMIV, stepFWD and stepRPC.

Examples

```
suppressMessages(library(PDtoolkit))
data(loans)
#identify numeric risk factors
num.rf <- sapply(loans, is.numeric)
num.rf <- names(num.rf)[!names(num.rf)%in%"Creditability" & num.rf]
#discretized numeric risk factors using ndr.bin from monobin package
loans[, num.rf] <- sapply(num.rf, function(x)
ndr.bin(x = loans[, x], y = loans[, "Creditability"])[[2]])
str(loans)
#create risk factor priority groups
rf.all <- names(loans)[-1]
set.seed(22)
blocks <- data.frame(rf = rf.all, block = sample(1:3, length(rf.all), rep = TRUE))
blocks <- blocks[order(blocks$block), ]
blocks
#method: stepMIV
res <- staged.blocks(method = "stepMIV",
   target = "Creditability",
   db = loans,
   coding = "WoE",
   blocks = blocks,
   miv.threshold = 0.02,
   m.ch.p.val = 0.05)
names(res)
nb <- length(res[["models"]])
res$models[[nb]]
auc.model(predictions = predict(res$models[[nb]], type = "response",
  newdata = res$dev.db[[nb]]),
    observed = res$dev.db[[nb]]$Creditability)

identical(unname(predict(res$models[[1]], type = "link", newdata = res$dev.db[[1]])),
    res$dev.db[[2]]$offset.vals)
identical(unname(predict(res$models[[2]], type = "link", newdata = res$dev.db[[2]])),
    res$dev.db[[3]]$offset.vals)
```

---

stepFWD   *Customized stepwise regression with p-value and trend check*

Description

stepFWD customized stepwise regression with p-value and trend check. Trend check is performed comparing observed trend between target and analyzed risk factor and trend of the estimated coefficients within the logistic regression. Note that procedure checks the column names of supplied db data frame therefore some renaming (replacement of special characters) is possible to happen. For details check help example.

Usage

```
stepFWD(
  start.model,
  p.value = 0.05,
  coding = "WoE",
  coding.start.model = TRUE,
  check.start.model = TRUE,
  db,
  offset.vals = NULL
)
```

Arguments

start.model	Formula class that represents starting model. It can include some risk factors, but it can be defined only with intercept (y ~ 1 where y is target variable).
p.value	Significance level of p-value of the estimated coefficients. For WoE coding this value is is directly compared to the p-value of the estimated coefficients, while for dummy coding multiple Wald test is employed and its p-value is used for comparison with selected threshold (p.value).
coding	Type of risk factor coding within the model. Available options are: "WoE" (default) and "dummy". If "WoE" is selected, then modalities of the risk factors are replaced by WoE values, while for "dummy" option dummies (0/1) will be created for n-1 modalities where n is total number of modalities of analyzed risk factor.
coding.start.model	
	Logical (TRUE or FALSE), if the risk factors from the starting model should be WoE coded. It will have an impact only for WoE coding option. Default is TRUE.
check.start.model	
	Logical (TRUE or FALSE), if risk factors from the starting model should be checked for p-value and trend in stepwise process. Default is TRUE. If FALSE is selected, then coding.start.model is forced to TRUE.
db	Modeling data with risk factors and target variable. All risk factors (apart from the risk factors from the starting model) should be categorized and as of character type.
offset.vals	This can be used to specify an a priori known component to be included in the linear predictor during fitting. This should be NULL or a numeric vector of length equal to the number of cases. Default is NULL.

Value

The command stepFWD returns a list of four objects.
The first object (model), is the final model, an object of class inheriting from "glm".
The second object (steps), is the data frame with risk factors selected at each iteration.
The third object (warnings), is the data frame with warnings if any observed. The warnings refer to the following checks: if risk factor has more than 10 modalities, if any of the bins (groups) has less than 5% of observations and if there are problems with WoE calculations.
The final, fourth, object dev.db returns the model development database.

Examples

```
suppressMessages(library(PDtoolkit))
data(loans)
#identify numeric risk factors
num.rf <- sapply(loans, is.numeric)
num.rf <- names(num.rf)[!names(num.rf)%in%"Creditability" & num.rf]
#discretized numeric risk factors using ndr.bin from monobin package
loans[, num.rf] <- sapply(num.rf, function(x)
ndr.bin(x = loans[, x], y = loans[, "Creditability"])[[2]])
str(loans)
res <- stepFWD(start.model = Creditability ~ 1,
    p.value = 0.05,
    coding = "dummy",
    db = loans)
summary(res$model)$coefficients
rf.check <- tapply(res$dev.db$Creditability,
 res$dev.db$Value_Savings_Stocks,
 mean)
rf.check
diff(rf.check)
res$steps
head(res$dev.db)
```

---

stepMIV	*Stepwise logistic regression based on marginal information value (MIV)*

Description

`stepMIV` performs stepwise logistic regression based on MIV.

Usage

```
stepMIV(
  start.model,
  miv.threshold,
  m.ch.p.val,
  coding,
  coding.start.model = FALSE,
  db,
  offset.vals = NULL
)
```

Arguments

start.model   Formula class that represent starting model. It can include some risk factors, but it can be defined only with intercept (y ~ 1 where y is target variable).

miv.threshold	MIV entrance threshold. Only the risk factors with MIV higher than the threshold are candidate for the new model. Additional criteria is that MIV value should significantly separate good from bad cases measured by marginal chi-square test.
m.ch.p.val	Significance level of p-value for marginal chi-square test. This test additionally supports MIV value of candidate risk factor for final decision.
coding	Type of risk factor coding within the model. Available options are: "WoE" and "dummy". If "WoE" is selected, then modalities of the risk factors are replaced by WoE values, while for "dummy" option dummies (0/1) will be created for n-1 modalities where n is total number of modalities of analyzed risk factor.
coding.start.model	
	Logical (TRUE or FALSE), if risk factors from the starting model should be WoE coded. It will have an impact only for WoE coding option. Default value is FALSE.
db	Modeling data with risk factors and target variable. All risk factors should be categorized as of character type.
offset.vals	This can be used to specify an a priori known component to be included in the linear predictor during fitting. This should be NULL or a numeric vector of length equal to the number of cases. Default is NULL.

Value

The command stepMIV returns a list of five objects.
The first object (model), is the final model, an object of class inheriting from "glm".
The second object (steps), is the data frame with risk factors selected at each iteration.
The third object (miv.iter), is the data frame with iteration details.
The fourth object (warnings), is the data frame with warnings if any observed. The warnings refer to the following checks: if risk factor has more than 10 modalities, if any of the bins (groups) has less than 5% of observations and if there are problems with WoE calculations.
The final, fifth, object dev.db object dev.db returns the model development database.

References

Scallan, G. (2011). Class(ic) Scorecards: Selecting Characteristics and Attributes in Logistic Regression, Edinburgh Credit Scoring Conference, downloaded from here.

Examples

```
suppressMessages(library(PDtoolkit))
data(loans)
#identify numeric risk factors
num.rf <- sapply(loans, is.numeric)
num.rf <- names(num.rf)[!names(num.rf)%in%"Creditability" & num.rf]
#discretized numeric risk factors using ndr.bin from monobin package
loans[, num.rf] <- sapply(num.rf, function(x)
ndr.bin(x = loans[, x], y = loans[, "Creditability"])[[2]])
str(loans)
#run stepMIV
res <- stepMIV(start.model = Creditability ~ 1,
   miv.threshold = 0.02,
```

```
    m.ch.p.val = 0.05,
    coding = "WoE",
    coding.start.model = FALSE,
    db = loans)
#check output elements
names(res)
#extract the final model
final.model <- res$model
#print coefficients
summary(final.model)$coefficients
#print steps of stepwise
res$steps
#print head of all iteration details
head(res$miv.iter)
#print warnings
res$warnings
#print head of coded development data
head(res$dev.db)
#calculate AUC
auc.model(predictions = predict(final.model, type = "response", newdata = res$dev.db),
    observed = res$dev.db$Creditability)
```

---

stepRPC   *Stepwise logistic regression based on risk profile concept*

Description

`stepRPC` customized stepwise regression with p-value and trend check which additionally takes into account the order of supplied risk factors per group when selects a candidate for the final regression model. Trend check is performed comparing observed trend between target and analyzed risk factor and trend of the estimated coefficients within the logistic regression. Note that procedure checks the column names of supplied db data frame therefore some renaming (replacement of special characters) is possible to happen. For details, please, check the help example.

Usage

```
stepRPC(
  start.model,
  risk.profile,
  p.value = 0.05,
  coding = "WoE",
  coding.start.model = TRUE,
  check.start.model = TRUE,
  db,
  offset.vals = NULL
)
```

Arguments

start.model	Formula class that represents the starting model. It can include some risk factors, but it can be defined only with intercept (y ~ 1 where y is target variable).
risk.profile	Data frame with defined risk profile. It has to contain the following columns: rf and group. Column group defines order of groups that will be tested first as a candidate for the regression model. Risk factors selected in each group are kept as a starting variables for the next group testing. Column rf contains all candidate risk factors supplied for testing.
p.value	Significance level of p-value of the estimated coefficients. For WoE coding this value is is directly compared to the p-value of the estimated coefficients, while for dummy coding multiple Wald test is employed and its value is used for comparison with selected threshold (p.value).
coding	Type of risk factor coding within the model. Available options are: "WoE" and "dummy". If "WoE" is selected, then modalities of the risk factors are replaced by WoE values, while for "dummy" option dummies (0/1) will be created for n-1 modalities where n is total number of modalities of analyzed risk factor.
coding.start.model	
	Logical (TRUE or FALSE), if the risk factors from the starting model should be WoE coded. It will have an impact only for WoE coding option.
check.start.model	
	Logical (TRUE or FALSE), if risk factors from the starting model should checked for p-value and trend in stepwise process.
db	Modeling data with risk factors and target variable. All risk factors (apart from the risk factors from the starting model) should be categorized and as of character type.
offset.vals	This can be used to specify an a priori known component to be included in the linear predictor during fitting. This should be NULL or a numeric vector of length equal to the number of cases. Default is NULL.

Value

The command stepRPC returns a list of four objects.
The first object (model), is the final model, an object of class inheriting from "glm".
The second object (steps), is the data frame with risk factors selected at each iteration.
The third object (warnings), is the data frame with warnings if any observed. The warnings refer to the following checks: if risk factor has more than 10 modalities, if any of the bins (groups) has less than 5% of observations and if there are problems with WoE calculations.
The final, fourth, object dev.db returns the model development database.

Examples

```
suppressMessages(library(PDtoolkit))
data(loans)
#identify numeric risk factors
num.rf <- sapply(loans, is.numeric)
num.rf <- names(num.rf)[!names(num.rf)%in%"Creditability" & num.rf]
#discretized numeric risk factors using ndr.bin from monobin package
```

```
loans[, num.rf] <- sapply(num.rf, function(x)
ndr.bin(x = loans[, x], y = loans[, "Creditability"])[[2]])
str(loans)
#create risk factor priority groups
rf.all <- names(loans)[-1]
set.seed(591)
rf.pg <- data.frame(rf = rf.all, group = sample(1:3, length(rf.all), rep = TRUE))
head(rf.pg)
#bring AUC for each risk factor in order to sort them within groups
bva <- bivariate(db = loans, target = "Creditability")[[1]]
rf.auc <- unique(bva[, c("rf", "auc")])
rf.pg <- merge(rf.pg, rf.auc, by = "rf", all.x = TRUE)
#prioritized risk factors
rf.pg <- rf.pg[order(rf.pg$group, rf.pg$auc), ]
rf.pg <- rf.pg[order(rf.pg$group), ]
rf.pg
res <- stepRPC(start.model = Creditability ~ 1,
   risk.profile = rf.pg,
   p.value = 0.05,
   coding = "WoE",
   db = loans)
summary(res$model)$coefficients
res$steps
head(res$dev.db)
```

---

univariate   *Univariate analysis*

Description

univariate returns the univariate statistics for risk factors supplied in data frame db.
For numeric risk factors univariate report includes:

- rf: Risk factor name.
- rf.type: Risk factor class. This metric is always equal to numeric.
- bin.type: Bin type - special or complete cases.
- bin: Bin type. If a sc.method argument is equal to "together", then bin and bin.type have the same value. If the sc.method argument is equal to "separately", then the bin will contain all special cases that exist for analyzed risk factor (e.g. NA, NaN, Inf).
- pct: Percentage of observations in each bin.
- cnt.unique: Number of unique values per bin.
- min: Minimum value.
- p1, p5, p25, p50, p75, p95, p99: Percentile values.
- avg: Mean value.
- avg.se: Standard error of the mean.
- max: Maximum value.

- neg: Number of negative values.

- pos: Number of positive values.

- cnt.outliers: Number of outliers. Records above or below `Q75`±`1.5 * IQR`, where `IQR = Q75 - Q25`.

- sc.ind: Special case indicator. It takes value 1 if share of special cases exceeds `sc.threshold` otherwise 0.

For categorical risk factors univariate report includes:

- rf: Risk factor name.

- rf.type: Risk factor class. This metric is equal to one of: `character`, `factor` or `logical`.

- bin.type: Bin type - special or complete cases.

- bin: Bin type. If a `sc.method` argument is equal to `"together"`, then `bin` and `bin.type` have the same value. If the `sc.method` argument is equal to `"separately"`, then the `bin` will contain all special cases that exist for analyzed risk factor (e.g. `NA`, `NaN`, `Inf`).

- pct: Percentage of observations in each `bin`.

- cnt.unique: Number of unique values per `bin`.

- sc.ind: Special case indicator. It takes value 1 if share of special cases exceeds `sc.threshold` otherwise 0.

Usage

```
univariate(
  db,
  sc = c(NA, NaN, Inf),
  sc.method = "together",
  sc.threshold = 0.2
)
```

Arguments

db	Data frame of risk factors supplied for univariate analysis.
sc	Vector of special case elements. Default values are `c(NA, NaN, Inf)`.
sc.method	Define how special cases will be treated, all together or in separate bins. Possible values are `"together"`, `"separately"`.
sc.threshold	Threshold for special cases expressed as percentage of total number of observations. If `sc.method` is set to `"separately"`, then percentage for each special case will be summed up.

Value

The command `univariate` returns the data frame with explained univariate metrics for numeric, character, factor and logical class of risk factors.

Examples

```
suppressMessages(library(PDtoolkit))
data(gcd)
gcd$age[100:120] <- NA
gcd$age.bin <- ndr.bin(x = gcd$age, y = gcd$qual, y.type = "bina")[[2]]
gcd$age.bin <- as.factor(gcd$age.bin)
gcd$maturity.bin <- ndr.bin(x = gcd$maturity, y = gcd$qual, y.type = "bina")[[2]]
gcd$amount.bin <- ndr.bin(x = gcd$amount, y = gcd$qual, y.type = "bina")[[2]]
gcd$all.miss1 <- NaN
gcd$all.miss2 <- NA
gcd$tf <- sample(c(TRUE, FALSE), nrow(gcd), rep = TRUE)
#create date variable to confirm that it will not be processed by the function
gcd$dates <- Sys.Date()
str(gcd)
univariate(db = gcd)
```

---

woe.tbl	*Weights of evidence (WoE) table*

Description

woe.tbl calculates WoE and information value for given target variable and risk factor along with accompanied metrics needed for their calculation. WoE table reports:

- bin: Risk factor group (bin).
- no: Number of observations per bin.
- ng: Number of good cases (where target is equal to 0) per bin.
- nb: Number of bad cases (where target is equal to 1) per bin.
- pct.o: Percentage of observations per bin.
- pct.g: Percentage of good cases (where target is equal to 0) per bin.
- pct.b: Percentage of bad cases (where target is equal to 1) per bin.
- dr: Default rate per bin.
- so: Number of all observations.
- sg: Number of all good cases.
- sb: Number of all bad cases.
- dist.g: Distribution of good cases per bin.
- dist.b: Distribution of bad cases per bin.
- woe: WoE value.
- iv.b: Information value per bin.
- iv.s: Information value of risk factor (sum of individual bins' information values).

Usage

```
woe.tbl(tbl, x, y, y.check = TRUE)
```

Arguments

`tbl`	Data frame which contains target variable (y) and analyzed risk factor (x).
`x`	Selected risk factor.
`y`	Selected target variable.
`y.check`	Logical, if target variable (y) should be checked for 0/1 values. Default value is `TRUE`. Change of this parameter to `FALSE` can be handy for calculation of WoE based on model predictions. Concretely, it is used only in calculation of marginal information value (MIV) in `stepMIV`.

Value

The command `woe.tbl` returns the data frame with WoE and information value calculations along with accompanied metrics.

See Also

`bivariate` for automatic bivariate analysis.

Examples

```
suppressMessages(library(PDtoolkit))
data(gcd)
#categorize numeric risk factors
gcd$age.bin <- woe.bin(x = gcd$age, y = gcd$qual, y.type = "bina")[[2]]
#generate woe table
woe.tbl(tbl = gcd, x = "age.bin", y = "qual")
```

Index