Package: woeBinning (via r-universe)

August 31, 2024

Type Package

Title Supervised Weight of Evidence Binning of Numeric Variables and Factors

Version 0.1.6

Date 2018-07-10

Author Thilo Eichenberg

Maintainer Thilo Eichenberg <te.r@gmx.net>

Description Implements an automated binning of numeric variables and factors with respect to a dichotomous target variable. Two approaches are provided: An implementation of fine and coarse classing that merges granular classes and levels step by step. And a tree-like approach that iteratively segments the initial bins via binary splits. Both procedures merge, respectively split, bins based on similar weight of evidence (WOE) values and stop via an information value (IV) based criteria. The package can be used with single variables or an entire data frame. It provides flexible tools for exploring different binning solutions and for deploying them to (new) data.

License GPL (>= 2)

LazyData TRUE

RoxygenNote 6.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2018-07-28 16:40:02 UTC

Contents

`germancredit` *German Credit Data*

Description

Credit data that classifies debtors described by a set of attributes as good or bad credit risks. See source link below for detailed information.

Usage

```
data(germancredit)
```

Format

A data frame with 21 variables (numeric and factors) and 1000 observations.

Source

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

Examples

```
# Load German credit data and create subset
data(germancredit)
df <- germancredit[, c('creditability', 'credit.amount', 'duration.in.month',
                  'savings.account.and.bonds', 'purpose')]
# Display structure of the subset (data frame)
str(df)
```

`woe.binning` *Binning via Fine and Coarse Classing*

Description

`woe.binning` generates a supervised fine and coarse classing of numeric variables and factors with respect to a dichotomous target variable. Its parameters provide flexibility in finding a binning that fits specific data characteristics and practical needs.

Usage

```
woe.binning(df, target.var, pred.var, min.perc.total,
            min.perc.class, stop.limit, abbrev.fact.levels, event.class)
```

Arguments

- `df` — Name of data frame with input data.
- `target.var` — Name of dichotomous target variable in quotes. Only target variables with two distinct values (e.g. 0, 1 or "Y", "N") are accepted; cases with NAs in the target variable will be ignored.
- `pred.var` — Name of predictor variable(s) to be binned in quotes. A single variable name can be provided, e.g. "varname1", or a list of variable names, e.g. c("varname1", "varname2"). Alternatively one can repeat the name of the input data frame; the function will be applied to all its variables apart from the target variable then. Numeric variables and factors are supported and may contain NAs.
- `min.perc.total` — For numeric variables this parameter defines the number of initial classes before any merging is applied. For example *min.perc.total=0.05* (5%) will result in 20 initial classes. For factors the original levels with a percentage below this limit are collected in a 'miscellaneous' level before the merging based on the *min.perc.class* and on the WOE starts. Increasing the *min.perc.total* parameter will avoid sparse bins. Accepted range: 0.0001-0.2; default: 0.05.
- `min.perc.class` — If a column percentage of one of the target classes within a bin is below this limit (e.g. below 0.01=1%) then the respective bin will be joined with others. In case of numeric variables adjacent predictor classes are merged. For factors respective levels (including sparse NAs) are assigned to a 'miscellaneous' level. Setting *min.perc.class*>0 may provide more reliable WOE values. Accepted range: 0-0.2; default: 0, i.e. no merging with respect to sparse target classes is applied.
- `stop.limit` — Stops WOE based merging of the predictor's classes/levels in case the resulting information value (IV) decreases more than *x%* (e.g. 0.05 = 5%) compared to the preceding binning step. *stop.limit=0* will skip any WOE based merging. Increasing the *stop.limit* will simplify the binning solution and may avoid overfitting. Accepted range: 0-0.5; default: 0.1.
- `abbrev.fact.levels` — Abbreviates the names of new (merged) factor levels via the base R `abbreviate` function in case the specified number of characters is exceeded. Accepted range: 0-1000; default: 200. 0 will prevent applying any abbreviation, i.e. only factor levels with more than 1000 characters will be truncated then. This option is particularly relevant in case one wants to generate dummy variables via the `woe.binning.deploy` function, because the factor levels will be part of the dummy variable names then.
- `event.class` — Optional parameter for specifying the class of the target event. This class typically indicates a negative event like a loan default or a disease. Use integers (e.g. 1) or characters in quotes (e.g. "bad"). This class will be represented by negative WOE values then.

Value

`woe.binning` generates an object containing the information necessary for studying and applying the realized binning solution. When saved it can be used with the functions `woe.binning.plot`, `woe.binning.table` and `woe.binning.deploy`.

Binning of Numeric Variables

Numeric variables (continuous and ordinal) are binned by merging initial classes with similar frequencies. The number of initial bins results from the *min.perc.total* parameter: min.perc.total will result in trunc(1/min.perc.total) initial bins, whereby *trunc* is needed to guarantee bins with similar frequencies. For example *min.perc.total=0.07* will cause trunc(14.3)=14 initial classes. Next, if *min.perc.class*>0, bins with sparse target classes will be merged with the next upper bin, and in case of the last bin with the next lower one. NAs have their own bin and will not be merged with others. Finally nearby bins with most similar weight of evidence (WOE) values are joined step by step until the information value (IV) decreases more than specified by a percentage value (*stop.limit* parameter) or until two bins are reached.

Binning of Factors

Factors (categorical variables) are binned by merging factor levels. As a start sparse levels (defined via the *min.perc.total* and *min.perc.class* parameters) are merged to a 'miscellaneous' level: if possible, respective levels (including sparse NAs) are bundled as 'misc. level pos.' (associated with positive WOE values), respectively as 'misc. level neg.' (associated with negative WOE values). In case a misc. level contains only NAs it will be named 'Missing'. Afterwards levels with similar WOE values are joined step by step until the information value (IV) decreases more than specified by a percentage value (*stop.limit* parameter) or until two bins are reached.

Adjustment of 0 Frequencies

In case the crosstab of the bins with the target classes contains frequencies = 0 the column percentages are adjusted to be able to compute the WOE and IV values: the offset 0.0001 (=0.01%) is added to each column percentage cell and the column percentages are recomputed then. This allows considering bins associated with one target class only, but may cause extreme WOE values for these bins. If a correction is not appropriate choose *min.perc.class*>0; bins with sparse target classes will be merged then before computing any WOE or IV value.

Handling of Missing Data

Cases with NAs in the target variable will be ignored. For predictor variables the following applies: in case NAs already occurred when generating the binning solution the code 'Missing' is displayed and a corresponding WOE value can be computed. (Note that factor NAs may be joined with other sparse levels to a 'miscellaneous' level - see above; only this 'miscellaneous' level will be displayed then.) In case NAs occur in the deployment scenario only 'Missing' is displayed for numeric variables and 'unknown' for factors; and the corresponding WOE values will be NA then, as well.

See Also

Other binning functions: `woe.tree.binning`

Examples

```
# Load German credit data and create subset
data(germancredit)
df <- germancredit[, c('creditability', 'credit.amount', 'duration.in.month',
```

```
                    'savings.account.and.bonds', 'purpose')]

# Bin a single numeric variable
binning <- woe.binning(df, 'creditability', 'duration.in.month',
                       min.perc.total=0.05, min.perc.class=0.01,
                       stop.limit=0.1, event.class='bad')

# Bin a single factor
binning <- woe.binning(df, 'creditability', 'purpose',
                       min.perc.total=0.05, min.perc.class=0, stop.limit=0.1,
                       abbrev.fact.levels=50, event.class='bad')

# Bin two variables (one numeric and one factor)
# with default parameter settings
binning <- woe.binning(df, 'creditability', c('credit.amount','purpose'))

# Bin all variables of the data frame (apart from the target variable)
# with default parameter settings
binning <- woe.binning(df, 'creditability', df)
```

`woe.binning.deploy` *Deployment of Binning*

Description

`woe.binning.deploy` applies the binning solution generated and saved via the `woe.binning` or `woe.tree.binning` function to (new) data.

Usage

```
woe.binning.deploy(df, binning, min.iv.total, add.woe.or.dum.var)
```

Arguments

`df`	Name of the data frame the binning solution - that was generated via the function `woe.binning` or `woe.tree.binning` - should be applied to. The variable names and types (numerical or factor) need to be identical to the ones used during the generation of the binning solution.
`binning`	Binning information generated from the `woe.binning` or `woe.tree.binning` function. Contains names of the input predictor variables and the corresponding binning, WOE and IV information, which is used to add a binned variable to a copy of the input data.
`min.iv.total`	If the IV total value of a binned variable falls below this limit (e.g. 0.1) it will not be added to the data. Just omit this parameter in case you would like to add all binned variables (default).

`add.woe.or.dum.var`
: *add.woe.or.dum.var*= *"woe"* adds an additional variable with WOE scores and = *"dum"* additional dummy variables for each (aggregated) level of the binned variable. In case of dummy variables make sure that you have set an appropriate *abbrev.fact.levels* parameter in the `woe.binning` or `woe.tree.binning` function to avoid too long variable names. In principle, only alphanumeric characters and dots (.) will be used for variable names. Just omit this parameter in case you don't need additional variables.

General Procedure

`woe.binning.deploy` applies the binning information that was generated from the `woe.binning` or `woe.tree.binning` function to a data frame. In this data frame the names of the variables to be binned need to be identical to the ones used with the `woe.binning` or `woe.tree.binning` function. For each variable a binned version will be added. Optionally a variable with associated weight of evidence (WOE) values or corresponding dummy variables (one dummy variable for each final bin) are provided.

Handling of Missing Data

In case NAs already occurred during the `woe.binning` or `woe.tree.binning` binning process the code 'Missing' is displayed and a corresponding WOE value can be computed. In case NAs only occur in the deployment scenario 'Missing' is displayed for numeric variables and 'unknown' for factors; and the corresponding WOE values will be NAs then, as well.

Handling of Unknown Factor Levels

For factor levels that have not been provided in generating the binning solution via the `woe.binning` or `woe.tree.binning` function a new factor level 'unknown' is displayed and the corresponding WOE value will be NA.

Examples

```
# Load German credit data and create a subset
data(germancredit)
df <- germancredit[, c('creditability', 'credit.amount', 'duration.in.month',
                  'savings.account.and.bonds', 'purpose')]

# Bin all variables of the data frame (apart from the target variable)
# with default parameter settings
binning <- woe.binning(df, 'creditability', df)

# Deploy the binning solution to the data frame
# (add all binned variables and corresponding WOE variables)
df.with.binned.vars.added <- woe.binning.deploy(df, binning,
                                                add.woe.or.dum.var='woe')

# Deploy the binning solution to the data frame
# (add binned variables with IV>=0.1 and corresponding dummy variables)
df.with.binned.vars.added <- woe.binning.deploy(df, binning,
                                                min.iv.total=0.1,
```

```
                                        add.woe.or.dum.var='dum')
```

`woe.binning.plot` *Visualization of Binning*

Description

woe.binning.plot visualizes the binning solution generated and saved via the woe.binning or woe.tree.binning function.

Usage

```
woe.binning.plot(binning, multiple.plots, plot.range)
```

Arguments

binning	Binning information generated from the woe.binning or woe.tree.binning function. Contains names of the input predictor variables and the corresponding binning, WOE and IV information, which is used to generate the WOE and IV plots.
multiple.plots	In case the binning solution contains several predictor variables they will be visualized via multiple plots (max. four WOE plots per graph window). Use *multiple.plots=FALSE* to avoid this and to display single plots in separate windows.
plot.range	Range of variables that should be plotted in quotes. For example "1:10" will generate WOE plots and one IV plot for the ten variables with the highest IV values, "11:20" for the next ten variables and so on. Just omit this parameter to visualize all binned variables (default).

Details

For each predictor variable woe.binning.plot generates a weight of evidence (WOE) plot. In case of multiple predictors an additional plot with variables ranked via the information value (IV) will be displayed.

Examples

```
# Load German credit data
data(germancredit)
df <- germancredit

# Bin all variables of the data frame (apart from the target variable)
# with default parameter settings
binning <- woe.binning(df, 'creditability', df)

# Plot all binned variables as multiple plots
woe.binning.plot(binning)
```

```
# Plot only the first four binned variables with the highest IV value
# as multiple plots
woe.binning.plot(binning, plot.range='1:4')

# Plot the binned variables in single plots
woe.binning.plot(binning, multiple.plots=FALSE)
```

---

woe.binning.table *Tabulation of Binning*

Description

woe.binning.table tabulates the binning solution generated and saved via the woe.binning or woe.tree.binning function.

Usage

```
woe.binning.table(binning)
```

Arguments

binning	Binning information generated from the woe.binning or woe.tree.binning function. Contains names of the input predictor variables and the corresponding binning, counts, WOE and IV information, which is used to generate the tables.

Details

For each predictor variable woe.binning.table generates a table (data frame). This table contains the final bin labels, total counts, total distribution (column percentages), counts for the first and the second target class, distribution of the first and the second target class (column percentages), rate (row percentages) of the target event specified via the *event.class* parameter in the woe.binning or woe.tree.binning function, as well as weight of evidence (WOE) and information values (IV).

Examples

```
# Load German credit data and create a subset
data(germancredit)
df <- germancredit[, c('creditability', 'credit.amount', 'duration.in.month',
                  'savings.account.and.bonds', 'purpose')]

# Bin all variables of the data frame (apart from the target variable)
# with default parameter settings
binning <- woe.binning(df, 'creditability', df)

# Tabulate the binned variables
tabulate.binning <- woe.binning.table(binning)
tabulate.binning
```

```
## Not run:

# Plot a layouted table (using the gridExtra library) for a specific
# variable (in this example for the first binned variable
# with the highest IV value)
library(gridExtra)
grid.table(tabulate.binning[[1]],
          theme = ttheme_default(core=list(bg_params=
                  list(fill=c(rep(c('grey95','grey90'),
                  length.out=nrow(tabulate.binning[[1]])-1),
                  '#BCC7BD')),fg_params=list(cex=0.8)),
                  colhead=list(fg_params=list(cex=0.8))),
                  rows=NULL)

## End(Not run)
```

`woe.tree.binning` *Binning via Tree-Like Segmentation*

Description

`woe.tree.binning` generates a supervised tree-like segmentation of numeric variables and factors with respect to a dichotomous target variable. Its parameters provide flexibility in finding a binning that fits specific data characteristics and practical needs.

Usage

```
woe.tree.binning(df, target.var, pred.var, min.perc.total,
                 min.perc.class, stop.limit, abbrev.fact.levels, event.class)
```

Arguments

`df`	Name of data frame with input data.
`target.var`	Name of dichotomous target variable in quotes. Only target variables with two distinct values (e.g. 0, 1 or “Y”, “N”) are accepted; cases with NAs in the target variable will be ignored.
`pred.var`	Name of predictor variable(s) to be binned in quotes. A single variable name can be provided, e.g. “varname1”, or a list of variable names, e.g. c(“varname1”, “varname2”). Alternatively one can repeat the name of the input data frame; the function will be applied to all its variables apart from the target variable then. Numeric variables and factors are supported and may contain NAs.
`min.perc.total`	For numeric variables this parameter defines the number of initial classes before any merging or tree-like splitting is applied. For example *min.perc.total=0.05* (5%) will result in 20 initial classes. For factors the original levels with a percentage below this limit are collected in a ‘miscellaneous’ level before the merging based on the *min.perc.class* and the tree-like splitting based on the WOE

values starts. Increasing the *min.perc.total* parameter will avoid sparse bins. Accepted range: 0.0001-0.2; default: 0.01.

`min.perc.class` If a column percentage of one of the target classes within a bin is below this limit (e.g. below 0.01=1%) then the respective bin will be joined with others. In case of numeric variables adjacent predictor classes are merged. For factors respective levels (including sparse NAs) are assigned to a 'miscellaneous' level. Setting *min.perc.class*>0 may provide more reliable WOE values. Accepted range: 0-0.2; default: 0, i.e. no merging with respect to sparse target classes is applied.

`stop.limit` Stops WOE based segmentation of the predictor's classes/levels in case the resulting information value (IV) increases less than *x%* (e.g. 0.05 = 5%) compared to the preceding binning step. Increasing the *stop.limit* will simplify the binning solution and may avoid overfitting. Accepted range: 0-0.5; default: 0.1.

`abbrev.fact.levels`
Abbreviates the names of new (merged) factor levels via the base R `abbreviate` function in case the specified number of characters is exceeded. Accepted range: 0-1000; default: 200. 0 will prevent applying any abbreviation, i.e. only factor levels with more than 1000 characters will be truncated then. This option is particularly relevant in case one wants to generate dummy variables via the `woe.binning.deploy` function, because the factor levels will be part of the dummy variable names then.

`event.class` Optional parameter for specifying the class of the target event. This class typically indicates a negative event like a loan default or a disease. Use integers (e.g. 1) or characters in quotes (e.g. "bad"). This class will be represented by negative WOE values then.

Value

`woe.tree.binning` generates an object with the information necessary for studying and applying the realized binning solution. When saved it can be used with the functions `woe.binning.plot`, `woe.binning.table` and `woe.binning.deploy`.

Binning of Numeric Variables

Numeric variables (continuous and ordinal) are binned beginning with initial classes with similar frequencies. The number of initial bins results from the *min.perc.total* parameter: min.perc.total will result in trunc(1/min.perc.total) initial bins, whereby *trunc* is needed to guarantee bins with similar frequencies. For example *min.perc.total=0.07* will cause trunc(14.3)=14 initial classes. Next, if *min.perc.class*>0, bins with sparse target classes will be merged with the next upper bin, and in case of the last bin with the next lower one. NAs have their own bin and will not be merged with others. Finally the actual tree-like procedure starts: binary splits iteratively assign nearby classes with similar weight of evidence (WOE) values to segments in a way that maximizes the resulting information value (IV). The procedure stops when the IV increases less then specified by a percentage value (*stop.limit* parameter).

Binning of Factors

Factors (categorical variables) are binned via factor levels. As a start sparse levels (defined via the *min.perc.total* and *min.perc.class* parameters) are merged to a 'miscellaneous' level: if possible, respective levels (including sparse NAs) are bundled as 'misc. level pos.' (associated with positive WOE values), respectively as 'misc. level neg.' (associated with negative WOE values). In case a misc. level contains only NAs it will be named 'Missing'. Afterwards the actual tree-like procedure starts: binary splits iteratively assign levels with similar WOE values to segments in a way that maximizes the resulting information value (IV). The procedure stops when the IV increases less then specified by a percentage value (*stop.limit* parameter).

Adjustment of 0 Frequencies

In case the crosstab of the bins with the target classes contains frequencies = 0 the column percentages are adjusted to be able to compute the WOE and IV values: the offset 0.0001 (=0.01%) is added to each column percentage cell and the column percentages are recomputed then. This allows considering bins associated with one target class only, but may cause extreme WOE values for these bins. If a correction is not appropriate choose *min.perc.class*>0; bins with sparse target classes will be merged then before computing any WOE or IV value.

Handling of Missing Data

Cases with NAs in the target variable will be ignored. For predictor variables the following applies: in case NAs already occurred when generating the binning solution the code 'Missing' is displayed and a corresponding WOE value can be computed. (Note that factor NAs may be joined with other sparse levels to a 'miscellaneous' level - see above; only this 'miscellaneous' level will be displayed then.) In case NAs occur in the deployment scenario only 'Missing' is displayed for numeric variables and 'unknown' for factors; and the corresponding WOE values will be NA then, as well.

See Also

Other binning functions: `woe.binning`

Examples

```
# Load German credit data and create subset
data(germancredit)
df <- germancredit[, c('creditability', 'credit.amount', 'duration.in.month',
                  'savings.account.and.bonds', 'purpose')]

# Bin a single numeric variable
binning <- woe.tree.binning(df, 'creditability', 'duration.in.month',
                           min.perc.total=0.01, min.perc.class=0.01,
                           stop.limit=0.1, event.class='bad')

# Bin a single factor
binning <- woe.tree.binning(df, 'creditability', 'purpose',
                           min.perc.total=0.05, min.perc.class=0, stop.limit=0.1,
                           abbrev.fact.levels=50, event.class='bad')

# Bin two variables (one numeric and one factor)
# with default parameter settings
binning <- woe.tree.binning(df, 'creditability', c('credit.amount','purpose'))
```

```
# Bin all variables of the data frame (apart from the target variable)
# with default parameter settings
binning <- woe.tree.binning(df, 'creditability', df)
```

---

woeBinning *Package for Supervised Weight of Evidence Binning*

Description

This package generates, visualizes, tabulates and deploys a supervised weight of evidence (WOE) binning of variables.

Details

The package `woeBinning` automates the process of binning of numeric variables and factors with respect to a dichotomous target variable. Additionally, it visualizes the realized binning solution, tabulates it and deploys it to (new) data. All functions can be used with single variables or an entire data frame.

Binning Functions

- `woe.binning` generates a supervised fine and coarse classing of numeric variables and factors.
- `woe.tree.binning` generates a supervised tree-like segmentation of numeric variables and factors.
- `woe.binning.plot` visualizes the binning solution generated and saved via `woe.binning` or `woe.tree.binning`.
- `woe.binning.table` tabulates the binning solution generated and saved via `woe.binning` or `woe.tree.binning`.
- `woe.binning.deploy` deploys the binning solution generated and saved via `woe.binning` or `woe.tree.binning` to (new) data.

References

Siddiqi, N. 2006: *Credit Risk Scorecards: Developing and Implementing Intelligent Credit Scoring*. Hoboken, New Jersey: John Wiley & Sons.

Anderson, R. 2007: *The Credit Scoring Toolkit: Theory and Practice for Retail Credit Risk Management and Decision Automation*. Oxford / New York: Oxford University Press.

Examples

```
# Load German credit data and create subset
data(germancredit)
df <- germancredit[, c('creditability', 'credit.amount', 'duration.in.month',
                  'savings.account.and.bonds', 'purpose')]

# Bin all variables of the data frame (apart from the target variable)
```

```
# with default parameter settings
binning <- woe.binning(df, 'creditability', df)

# Plot the binned variables
woe.binning.plot(binning)

# Tabulate the binned variables
tabulate.binning <- woe.binning.table(binning)
tabulate.binning

# Deploy the binning solution to the data frame
# (i.e. add binned variables and corresponding WOE variables)
df.with.binned.vars.added <- woe.binning.deploy(df, binning,
                                               add.woe.or.dum.var='woe')
```

Index