# Data Exploration with Weight of Evidence and Information Value in R

## Introduction

Binary classification models are perhaps the most common use-case in predictive analytics. The reason is that many key client actions across a wide range of industries are binary in nature, such as defaulting on a loan, clicking on an ad, or terminating a subscription.

Prior to building a binary classification model, a common step is to perform variable screening and exploratory data analysis. This is the step where we get to know the data and weed out variables that are either ill-conditioned or simply contain no information that will help us predict the action of interest. Note that the purpose of this step should not to be confused with that of multiple-variable selection techniques, such as stepwise regression and lasso, where the final variables that go into the model are selected. Rather, this is a precursory step designed to ensure that the approaches deployed during the final modeling phases are set up for success.

The *weight of evidence* (WOE) and *information value* (IV) provide a great framework for for exploratory analysis and variable screening for binary classifiers. WOE and IV have been used extensively in the credit risk world for several decades, and the underlying theory dates back to the 1950s. However, it is still not widely used outside the credit risk world, and it is a fairly underserved area in R.

WOE and IV enable one to:

- Consider each variable's independent contribution to the outcome.
- Detect linear and non-linear relationships.
- Rank variables in terms of "univariate" predictive strength.
- Visualize the correlations between the predictive variables and the binary outcome.
- Seamlessly compare the strength of continuous and categorical variables without dummy-coding.
- Identify variables with too many missing values.
- Assess the predictive power of missing values.

In this paper we will provide a brief description of WOE and IV as well as the `Information` R package, which provides an easy way to perform variable screening and exploratory analysis with WOE and IV.

# What Are WOE and IV?

## Background

Weight of evidence (WOE) and information value originate from *information theory* where one of the goals is to measure *entropy* – i.e., the uncertainty involved in predicting the outcome of a random variable (see [2]). Thus this is a perfect framework for variable screening and exploratory analysis for predictive modeling.

## Setup

Let's say that we have a binary dependent variable $Y$ and a set of predictive variables $X_1, \dots, X_p$. As mentioned above, $Y$ can capture a wide range of outcomes, such as defaulting on a loan, clicking on an ad, or terminating a subscription.

WOE and IV play two distinct roles when analyzing data:

- WOE *describes the relationship* between a predictive variable and a binary target variable.
- IV *measures the strength* of that relationship.

## The Weight of Evidence

The WOE/IV framework is based on the following relationship:

$$\log \frac{P(Y=1|X_j)}{P(Y=0|X_j)} = \underbrace{\log \frac{P(Y=1)}{P(Y=0)}}_{\text{sample log-odds}} + \underbrace{\log \frac{f(X_j|Y=1)}{f(X_j|Y=0)}}_{\text{WOE}}.$$

This is saying that the conditional logit, given $X_j$, can be written as the overall log-odds (i.e., the "intercept") plus the *log-density ratio* – also known as the *weight of evidence*.

Note that the weight of evidence and the conditional log odds of $Y = 1$ are perfectly correlated since the "intercept" is constant. Hence, the greater the value of WOE, the higher the chance of observing $Y = 1$. In fact, when WOE is positive, the chance of of observing $Y = 1$ is greater than average (for the sample), and vice versa when WOE is negative. When WOE equals 0, the odds are simply equal to the sample average.

## Ties to Naive Bayes and Logistic Regression

Notice that the left-hand side of the equation above – i.e., the conditional log odds – is exactly what we are trying to predict in a *logistic regression* model. Hence, when building a logistic regression model – which is perhaps the most widely used technique for building binary classifiers – we are actually trying to estimate the

weight of evidence. This is another reason why looking at WOE for precursory modeling exploration is a good idea.

Last, but not least, the WOE framework has ties to the well-known naive Bayes classifier, given by (see [1]):

$$\log \frac{P(Y=1|x_1,\ldots,x_p)}{P(Y=0|x_1,\ldots,x_p)} = \log \frac{P(Y=1)}{P(Y=0)} + \sum_{j=1}^{p} \log \frac{f(X_j|Y=1)}{f(X_j|Y=0)}..$$

The naive Bayes model essentially says that the conditional log odds is equal to the sum of the individual weight of evidence vectors. The word "naive" comes from the fact that this model relies on the assumption that all predictors are conditionally independent given $Y$.

In the credit scoring industry, a "semi-Naive" version of this model is quite popular. The idea is to transform the data into WOE vectors, and then use logistic regression to fit the model

$$\log \frac{P(Y=1|x_1,\ldots,x_p)}{P(Y=0|x_1,\ldots,x_p)} = \log \frac{P(Y=1)}{P(Y=0)} + \sum_{j=1}^{p} \beta_j \log \frac{f(X_j|Y=1)}{f(X_j|Y=0)},$$

thus partly relaxing the assumption that all predictors in the model are independent. However, it should be noted that the underlying WOE vectors are still estimated univariately and that the coefficients merely function as scalars. For a more flexible approach, the generalized additive model (GAM) is a great choice (see [2]).

### Measuring Univariate Variable Strength with the Information Value

We can leverage WOE to measure the predictive strength of $x_j$ – i.e., how well it helps us separate cases when $Y = 1$ from cases when \$Y = 0$. This is done through the *information value* (IV), which is defined like this:

$$\text{IV}_j = \int \log \frac{f(X_j|Y=1)}{f(X_j|Y=0)} \left(f(X_j|Y=1) - f(X_j|Y=0)\right) dx.$$

Note that the IV is essentially a weighted "sum" of all the individual WOE values where the weights incorporate differences between the numerators and the denominators. Generally, if $\text{IV} < 0.05$, the variable has very little predictive power.

### Estimating WOE

The most common approach to estimating the conditional densities needed to calculate WOE is to bin $X_j$ and then use a histogram-type estimator.

Here is how it works: create a $k \times 2$ table where $k$ is the number of bins, and the cells within the two columns count the number of records where $Y = 1$ and $Y = 0$, respectively. The bins are typically selected such that

the bins are roughly evenly sized with respect to the number of records in each bin (if possible). The conditional densities are then obtained by calculating the “column percentages” from this table. The typical number of bins used is 10-20. If $X_j$ is categorical, no binning is needed and the histogram estimator can be used directly. Moreover, missing values are treated as a separate bin and thus handled seamlessly.

If $B_1, \ldots, B_k$ denote the bins for $X_j$, the WOE for $X_j$ for bin $i$ can be written as

$$\text{WOE}_{ij} = \log \frac{P(X_j \in B_i | Y = 1)}{P(X_j \in B_i | Y = 0)},$$

which means that the IV for variable $X_j$ can be calculated as

$$\text{IV}_j = \sum_{i=1}^{k} (P(X_j \in B_i | Y = 1) - P(X_j \in B_i | Y = 0)) \times \text{WOE}_{ij}.$$

## Extensions to Exploratory Analysis for Uplift Models

Consider a direct marketing program where a *test group* received an offer of some sort, and the *control group* did not receive anything. The test and control groups are based on a random split. The lift of the campaign is defined as the difference in success rates between the test and control groups. In other words, the program can only be deemed successful if the offer outperforms the "do nothing" (a.k.a baseline) scenario.

The purpose of uplift models is to estimate the difference between the test and control groups, and then using the resulting model to target *persuadables* – i.e., potential or existing clients that are on the fence and need some sort of offer or contract to sign up or to buy a product. Actual uplift modeling techniques are beyond the scope of this paper; we will stick to model exploration and variable screening.

When preparing to build an uplift model, we cannot only focus on the log odds of $Y = 1$, we also need to consider the *log odds ratio* of $Y = 1$ for the test group versus the control group. This leads to the *net weight of evidence* (NWOE) and the *net information value* (NIV).

The net weight of evidence (NWOE) is the difference beteween the test group WOE ($\text{WOE}_t$(and the control group WOE ($\text{WOE}_c$)

$$\text{NWOE} = \text{WOE}_t - \text{WOE}_c,$$

which is equivalent to

$$\text{NWOE}_j = \log \frac{f(x_j | Y = 1)_t f(x_j | Y = 0)_c}{f(x_j | Y = 1)_c f(x_j | Y = 0)_t}.$$

The net information value for variable $X_j$ is then defined as

$$\int \left( f(X_j|Y=1)_t f(X_j|Y=0)_c - f(X_j|Y=1)_c f(X_j|Y=0)_t \right) \times \log \frac{f(X_j|Y=1)_t f(X_j|Y=0)_c}{f(X_j|Y=1)_c f(X_j|Y=0)_t} dx_j .$$

NWOE and NIV work just like IV and WOE, just from an uplift perspective. NIV measures the strength of a given variable, while NWOE describes the pattern of the relationship. Specifically, the higher the value of NIV, the better the given variable is at separating *self-selectors* – i.e., people who are self-motivated to buy – and *persuadables* that need to be motivated.

# The Information R Package

The information package is designed to perform exploratory data analysis and variable screening for binary classification models using WOE and IV. Aggregations are done in data.table, and creation of WOE vectors can be distributed across multiple cores. Thus the package is fairly fast.

The package can be downloaded from https://github.com/klarsen1/Information.

There are a number of great R packages available that support WOE and IV (see a partial list below), but they are either primarily designed to build WOE-based models – e.g., naive Bayes classifiers – or they do not support the uplift use-case. Hence, despite some redundancy, we saw the need to create the `Information` package.

## Date Used in Examples

The data is from an historical marketing campaign from the insurance industry and is automatically downloaded when you install the `Information` package. The data is stored in two `.RDA` files, one for the training dataset and one for the validation dataset. Each file has 68 predictive variables and 10k records.

The datasets contain two key indicators:

- `PURCHASE` This variable equals 1 if the client accepted the offer, and 0 otherwise
- `TREATMENT` This variable equals 1 if the client was in the test group (received the offer), and 0 otherwise.

## Key Functions

- `CreateTables()` creates WOE tables and IVs for all variables in the input dataframe.
- `PlotWOE()` plots the WOE vector for one variable.
- `MultiPlotWOE()` plots more than one WOE vector on one page for comparisons.

The same functions work for NWOE and NIV.

The examples below show how to call these functions

## External Cross Validation

The information package supports external cross validation to check that the WOE and NWOE vectors are stable.

Let's assume that we have a training and a validation dataset, and we calculate WOE for both datasets using the same bin cutoffs. The penalty for bin $B_i$ is given by

$$\text{penalty}_i = |\text{WOE}_{\text{train}} - \text{WOE}_{\text{valid}}|,$$

and the total penalty is

$$\sum_{i=1}^{k} |P(X_j \in B_i | Y = 1) - P(X_j \in B_i | Y = 0)| \times \text{penalty}_i.$$

The *adjusted IV* is then defined as

$$\text{NIV} = \text{IV} - \text{total penalty}.$$

The same approach can be applied to NIV for the uplift use-case.

## Example for a Traditional Binary Classification Problem

### Ranking All Variables Using Adjusted IV

```
library(Information)
library(gridExtra)

options(scipen=10)

### Loading the data
data(train, package="Information")
data(valid, package="Information")

### Exclude the control group
train <- subset(train, TREATMENT==1)
valid <- subset(valid, TREATMENT==1)

### Ranking variables using penalized IV
IV <- CreateTables(data=train,
                   valid=valid,
                   y="PURCHASE")

grid.table(head(IV$Summary), rows=NULL)
```

The strongest six variables:

| Variable | IV | PENALTY | AdjIV |
|---|---|---|---|
| N_OPEN_REV_ACTS | 1.0107695 | 0.08385690 | 0.9269126 |
| TOT_HI_CRDT_CRDT_LMT | 0.9345902 | 0.10269736 | 0.8318929 |
| RATIO_BAL_TO_HI_CRDT | 0.8232539 | 0.06544355 | 0.7578104 |
| D_NA_M_SNC_MST_RCNT_ACT_OPN | 0.6355466 | 0.07667477 | 0.5588718 |
| M_SNC_OLDST_RETAIL_ACT_OPN | 0.5573438 | 0.07840106 | 0.4789427 |
| M_SNC_MST_RCNT_ACT_OPN | 0.5026402 | 0.06044698 | 0.4421932 |

### Analyzing WOE Patterns

The `IV$Tables` object returned by `Information` is simply a list of `dataframes` that contains the WOE tables for all variables in the input dataset. Note that the `penalty` and `IV` columns are cumulative.

```
grid.table(IV$Tables$N_OPEN_REV_ACTS, rows=NULL)
```

| N_OPEN_REV_ACTS | N | Percent | WOE | IV | PENALTY |
|---|---|---|---|---|---|
| [0,0] | 1469 | 0.29545455 | –2.0465968 | 0.6401443 | 0.05703080 |
| [1,2] | 958 | 0.19267900 | –0.5900120 | 0.6958705 | 0.06226262 |
| [3,3] | 310 | 0.06234916 | 0.2033085 | 0.6986029 | 0.06514553 |
| [4,5] | 583 | 0.11725664 | 0.4419768 | 0.7244762 | 0.06767437 |
| [6,8] | 632 | 0.12711183 | 0.6148243 | 0.7810611 | 0.07159274 |
| [9,11] | 453 | 0.09111022 | 0.8815772 | 0.8692672 | 0.07683238 |
| [12,48] | 567 | 0.11403862 | 0.9883818 | 1.0107695 | 0.08385690 |

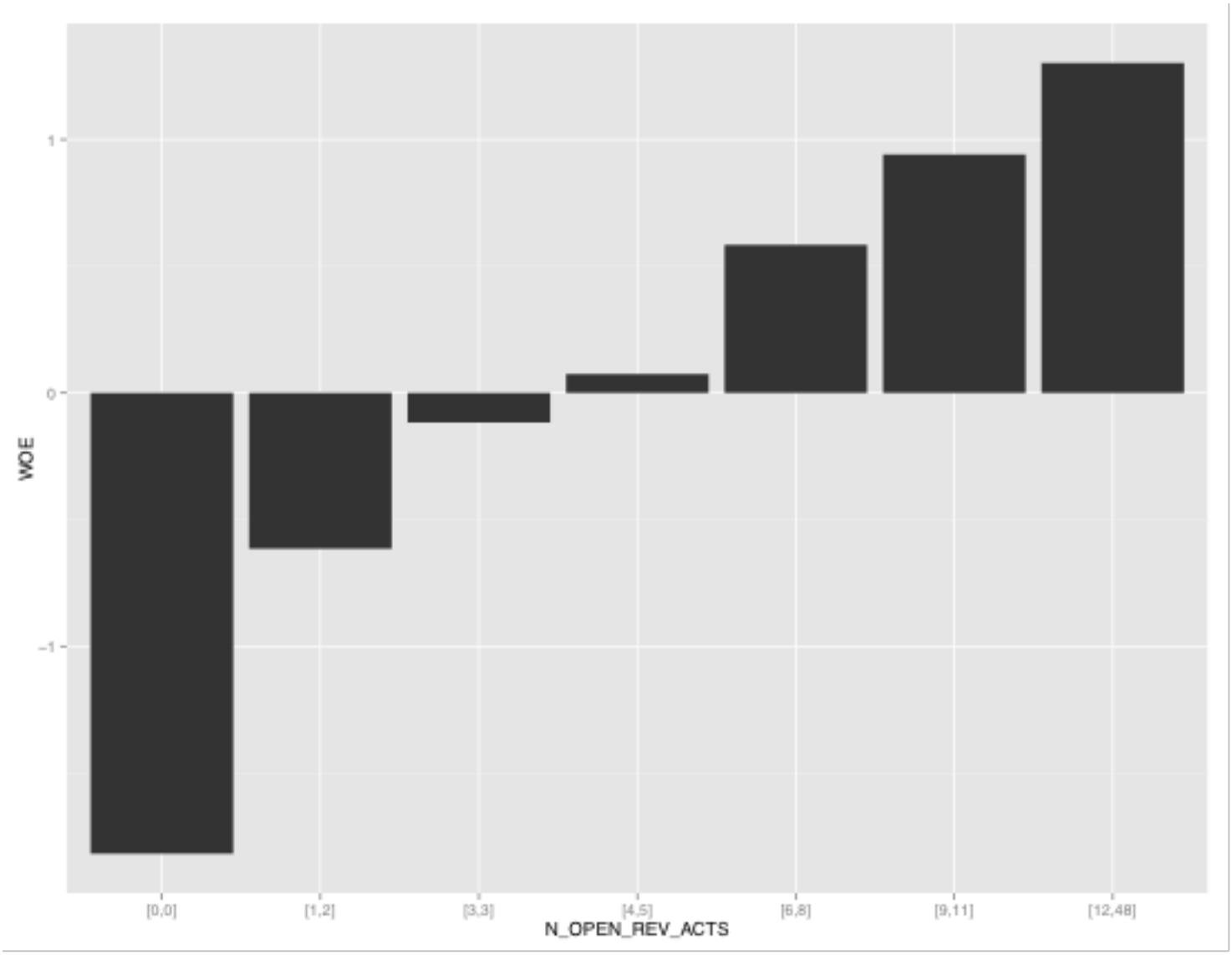
The table shows that the odds of `PURCHASE=1` increases as `N_OPEN_REV_ACTS` increases, although the relationship is not linear.

Note that the `Information` package attempts to create evenly-sized bins in terms of the number of subjects in each group. However, this is not always possible due to ties in the data, as with `N_OPEN_REV_ACTS` which has ties at 0. If the variable under consideration is categorical, its distinct categories will show up as rows in the WOE table. Moreover, if the variable has missing values, the WOE table will contain a separate "NA" row which can be used to gauge the impact of missing values. Thus, the framework seamlessly handles missing values and categorical variables without any dummy-coding or imputation .

**Plotting WOE Patterns**

We can also plot this pattern for better visualization:

```
PlotWOE(IV, "N_OPEN_REV_ACTS")
```

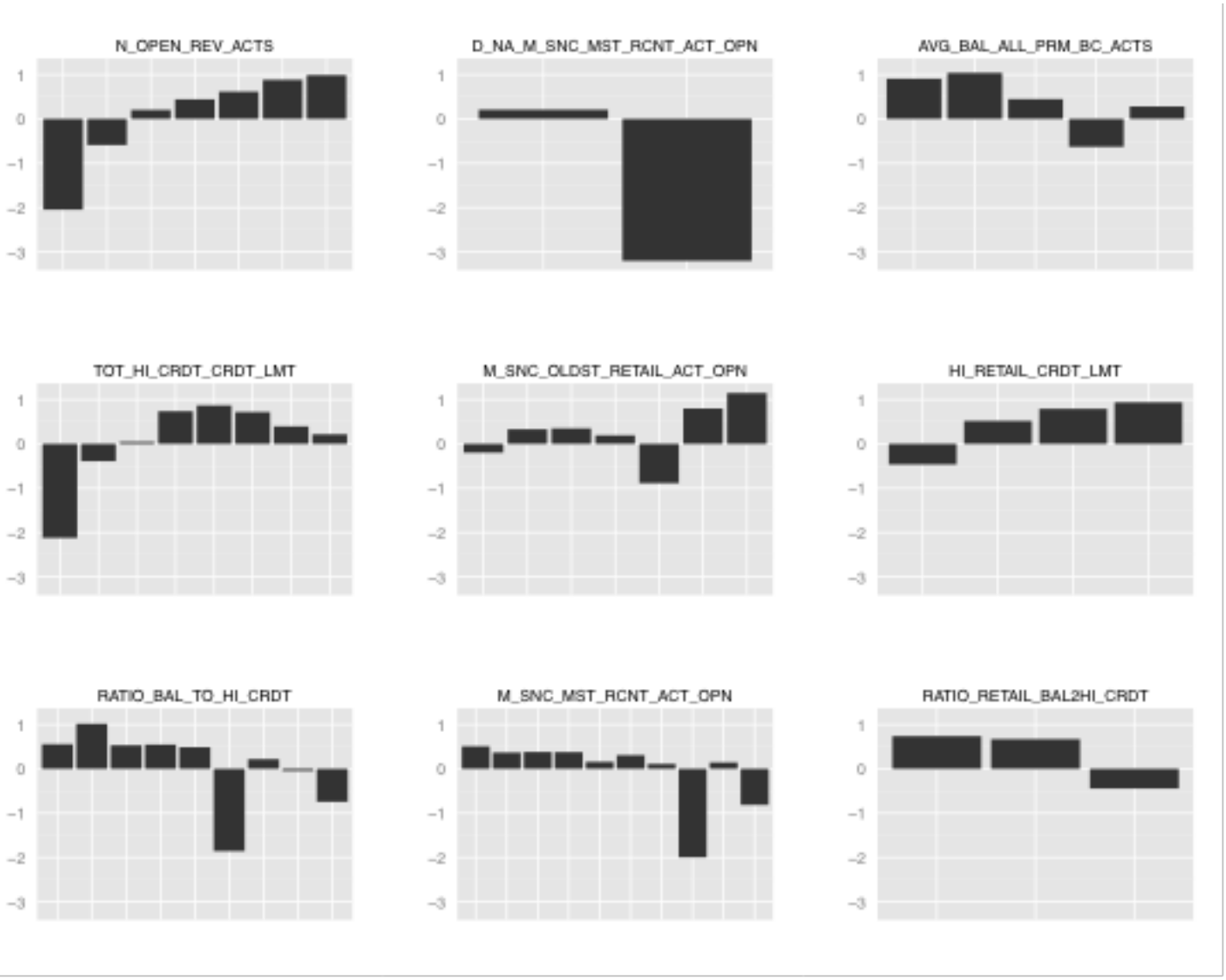



If we want to plot all variables on separate pages, we can loop through all variables:

```
n <- names(IV$Tables)
for (i in 1:length(n)){
   PlotWOE(IV, n[i])
}
```

Or, for better visualization, we can do a multiplot to compare WOE patterns. Here we plot the first nine variables on one page. Note that we can plot as many variables as we want; MultiPlotWOE will simply spread the plots over multiple pages (max of nine plots per page).

```
MultiPlotWOE(IV, IV$Summary$Variable[1:9])
```




## Omitting Cross Validation

To run IVs without external cross validation, simply omit the validation dataset:

```
IV <- CreateTables(data=train,
                   y="PURCHASE")
```

## Changing the Number of Bins

The default number of bins is 10, but we can choose a different number if we desire more granularity. Note that the IV formula is fairly invariant to the number of bins. Also, note that the bins are selected such that the bins are evenly sized, to the extent that it is possible (depending on the number of ties in the data).

```
IV <- CreateTables(data=train,
                   valid=valid,
                   y="PURCHASE",
                   bins=20)
grid.table(IV$Tables$N_OPEN_REV_ACTS,
           gp=gpar(fontsize=12),
           rows=NULL)
```

| N_OPEN_REV_ACTS | N | Percent | WOE | IV | PENALTY |
|---|---|---|---|---|---|
| [0,0] | 1469 | 0.29545455 | −2.0465968 | 0.6401443 | 0.05703080 |
| [1,2] | 958 | 0.19267900 | −0.5900120 | 0.6958705 | 0.06226262 |
| [3,3] | 310 | 0.06234916 | 0.2033085 | 0.6986029 | 0.06514553 |
| [4,5] | 583 | 0.11725664 | 0.4419768 | 0.7244762 | 0.06767437 |
| [6,8] | 632 | 0.12711183 | 0.6148243 | 0.7810611 | 0.07159274 |
| [9,11] | 453 | 0.09111022 | 0.8815772 | 0.8692672 | 0.07683238 |
| [12,48] | 567 | 0.11403862 | 0.9883818 | 1.0107695 | 0.08385690 |

## Uplift Example

For an uplift model, we have to include both the test group and the control group.

```
data(train, package="Information")
data(valid, package="Information")
```

When calling the `CreateTables` function, all we have to do is specify the variable that identifies the test and control groups.

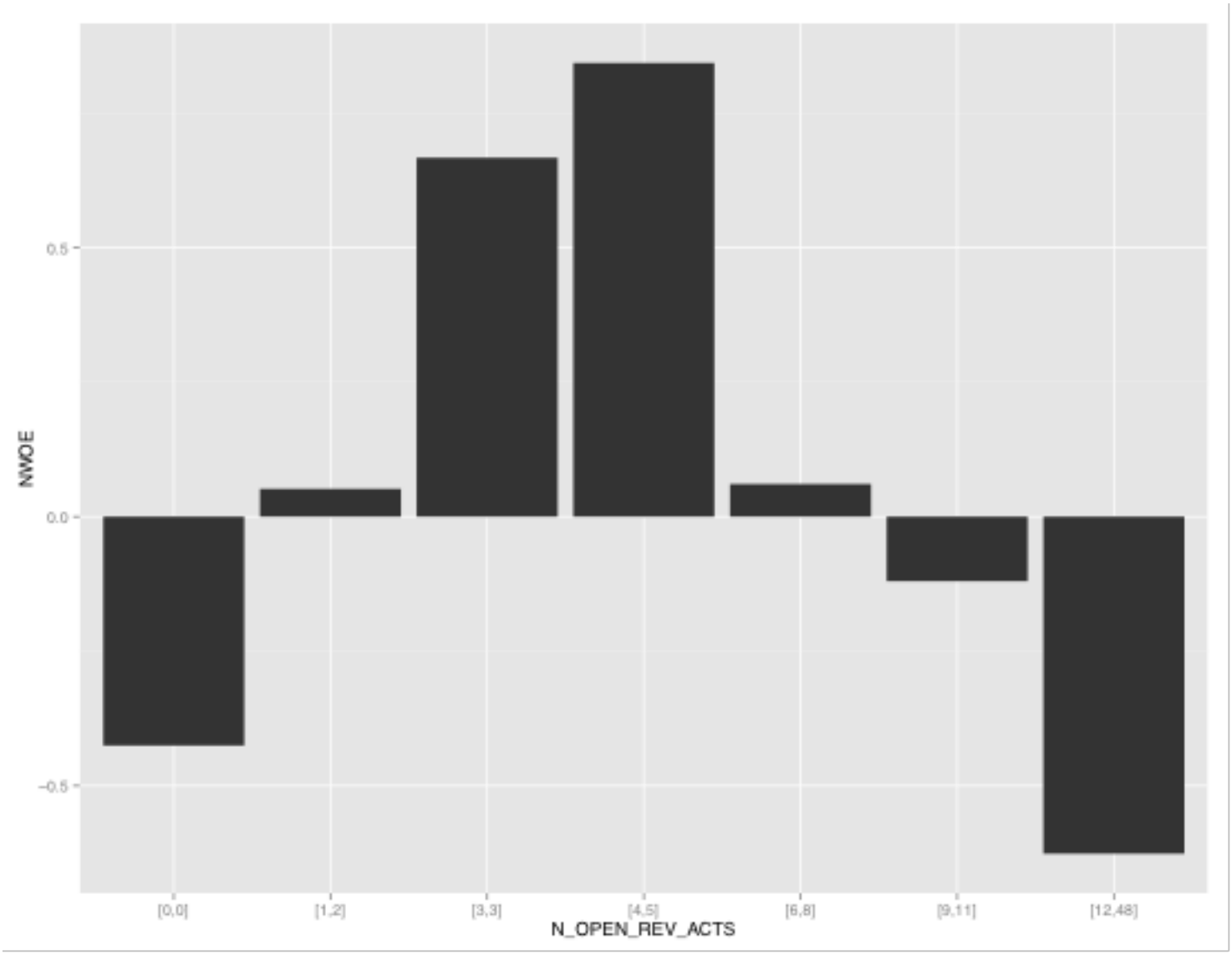
```
NIV <- CreateTables(data=train,
                    valid=valid,
                    y="PURCHASE",
                    trt="TREATMENT")

grid.table(head(NIV$Summary),
           rows=NULL,
           gp=gpar(fontsize=12))
```

| Variable | NIV | PENALTY | AdjNIV |
|---|---|---|---|
| N_OPEN_REV_ACTS | 0.23453611 | 0.079766143 | 0.15476997 |
| D_NA_M_SNC_OLDST_MRTG_ACT_OPN | 0.09140998 | 0.004117210 | 0.08729277 |
| M_SNC_MSTRCNT_MRTG_ACT_UPD | 0.09072817 | 0.003556086 | 0.08717209 |
| M_SNC_MSTREC_INSTL_TRD_OPN | 0.09622970 | 0.025260288 | 0.07096942 |
| MRTG_2_CURRENT_BAL | 0.07030452 | 0.005079736 | 0.06522478 |
| MRTG_1_MONTHLY_PAYMENT | 0.13837290 | 0.084556887 | 0.05381601 |

Note that we cannot compare the scales of NIV and IV. Moreover, there is no known, rule-of-thumb cutoff for the NIV. Hence, we have to use it solely as a ranking statistic and make a judgement call.

Interestingly, `N_OPEN_REV_ACTS` is also the most predictive variable from an uplift perspective. However, the NWOE pattern is quite different from the WOE pattern, and suggests a u-shaped pattern. This illustrates how the story can be very different when modeling the incremental effect as opposed to simply building a model to estimate the chance of $Y = 1$ following a treatment.

| N_OPEN_REV_ACTS | Percent | Treatment | Control | NWOE | WOE_t | WOE_c | NIV | PENALTY |
|---|---|---|---|---|---|---|---|---|
| [0,0] | 0.2931 | 1469 | 1462 | –0.42526740 | –2.0465968 | –1.6213294 | 0.03606464 | 0.04538061 |
| [1,2] | 0.1919 | 958 | 961 | 0.05159896 | –0.5900120 | –0.6416109 | 0.03670672 | 0.04649485 |
| [3,3] | 0.0672 | 310 | 362 | 0.66687280 | 0.2033085 | –0.4635643 | 0.05511137 | 0.05349898 |
| [4,5] | 0.1176 | 583 | 593 | 0.84291922 | 0.4419768 | –0.4009424 | 0.15430105 | 0.05536948 |
| [6,8] | 0.1288 | 632 | 656 | 0.06059508 | 0.6148243 | 0.5542292 | 0.15511473 | 0.05693023 |
| [9,11] | 0.0885 | 453 | 432 | –0.11986332 | 0.8815772 | 1.0014406 | 0.15679026 | 0.05698007 |
| [12,48] | 0.1129 | 567 | 562 | –0.62619770 | 0.9883818 | 1.6145795 | 0.23453611 | 0.07976614 |




## Combining IV With the ClustOfVar Package

Variable clustering divides a set of numeric variables into mutually exclusive clusters. The algorithm attempts to generate clusters such that

- the correlations between variables assigned to the same cluster are maximized.
- the correlations between variables in different clusters are minimized.

Using this algorithm, we can replace a large set of variables by a single member of each cluster, often with little loss of information. The question is which member to choose from a given cluster. One option is to choose the variable that has the highest multiple correlation with the variables within its cluster, and the lowest correlation with variables outside the cluster. A more meaningful choice for a predictive modeling is to choose the variable that has the highest information value. The following code shows how to do this.

```
library(ClustOfVar)
library(reshape2)
library(plyr)

data(train, package="Information")
data(valid, package="Information")
train <- subset(train, TREATMENT==1)
valid <- subset(valid, TREATMENT==1)

tree <- hclustvar(train[,!(names(train) %in% c("PURCHASE", "TREATMENT"))])
nvars <- length(tree[tree$height<0.7])
part_init<-cutreevar(tree,nvars)$cluster
kmeans<-kmeansvar(X.quanti=train[,!(names(train) %in% c("PURCHASE", "TREATMENT"))],in
it=part_init)
clusters <- cbind.data.frame(melt(kmeans$cluster), row.names(melt(kmeans$cluster)))
names(clusters) <- c("Cluster", "Variable")
clusters <- join(clusters, IV$Summary, by="Variable", type="left")
clusters <- clusters[order(clusters$Cluster),]
clusters$Rank <- ave(-clusters$AdjIV, clusters$Cluster, FUN=rank)
selected_members <- subset(clusters, Rank==1)
selected_members$Rank <- NULL
```

Using variable clustering in combination with IV cuts the number of variables from 68 to 21:

```
nrow(selected_members)
> 21
nrow(clusters)
> 69
```

Here are the first rows from the resulting variable ranking table:

```
grid.table(head(selected_members),
           rows=NULL,
           gp=gpar(fontsize=12))
```

| Cluster | Variable | IV | PENALTY | AdjIV |
|---|---|---|---|---|
| 1 | M_SNC_MST_RCNT_ACT_OPN | 0.5026402 | 0.06044698 | 0.4421932 |
| 2 | TOT_HI_CRDT_CRDT_LMT | 0.9345902 | 0.10269736 | 0.8318929 |
| 3 | RATIO_BAL_TO_HI_CRDT | 0.8232539 | 0.06544355 | 0.7578104 |
| 4 | AGRGT_BAL_ALL_XCLD_MRTG | 0.2718228 | 0.05601568 | 0.2158071 |
| 5 | N_OF_SATISFY_FNC_REV_ACTS | 0.2883329 | 0.04703612 | 0.2412968 |
| 6 | D_NA_M_SNC_MST_RCNT_ACT_OPN | 0.6355466 | 0.07667477 | 0.5588718 |

# Summary

- As stated in the introduction, *weight of evidence* (WOE) and *information value* (IV) provide a great framework for performing exploratory analysis and variable screening prior to building a binary classifier (e.g., logistic regression). It seamlessly handles missing values and character variables, and the output is easy to interpret.
- The purpose of exploratory analysis and variable screening is to get to know the data and assess "univariate" predictive strength, *before* we deploy more sophisticated variable selection approaches.
- The information package is specifically written to perform this type of analysis using parallel processing. It also supports exploratory analysis for uplift models, a growing area within marketing analytics. The information package is not designed to transfer data into WOE vectors for Naive Bayes models, although this feature could be added later.

# References

[1] Hastie, Trevor , Tibshirani, Robert and Friedman, Jerome. (1986), Elements of Statistical Learning, Second Edition, Springer, 2009.

[2] Kullback S., Information Theory and Statistics, John Wiley & Sons, 1959.

### Other Packages that support WOE/IV

woe package, https://cran.r-project.org/web/packages/woe/woe.pdf.

riv package (not to be confused with the riv packake in CRAN), http://www.r-bloggers.com/r-credit-scoring-woe-information-value-in-woe-package/.

klaR package, https://cran.r-project.org/web/packages/klaR/klaR.pdf.