Information Package Vignette

Kim Larsen

2016-04-08

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 variables that go into the final 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.

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 creating dummy variables.
- Seamlessly handle missing values without imputation.
- 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 for traditional binary classifiers as well as uplift models.

Information Theory

Weight of evidence (WOE) and information value are closely related to concepts from *information theory* where one of the goals is to understand the uncertainty involved in predicting the outcome of random events given varying degrees of knowledge of other variables (see [2], [3], and [4]). Thus this is a perfect framework for variable screening and exploratory analysis for predictive modeling.

Information theory kicked off when Claude Shannon (1948) wrote down an expression for the *entropy* of a probability distribution. Entropy is often described as the level of “disorder,” but in the context of information theory it is better to think of it as one’s level of uncertainty. High entropy means high uncertainty (you don’t know what tomorrow holds) and low entropy mean low uncertainty (there will be no surprises tomorrow).

Suppose a friend tosses a coin and asks you to predict whether the toss came up heads or tails. There’s a 50 percent probability of each possibility, so your prediction isn’t worth very much. This turns out to be the maximum amount of entropy for an event with two possible outcomes. Suppose instead that you know the coin is weighted so that it comes up heads 90 percent of the time. Now your job of making a prediction is easier – Shannon's formula tells you exactly how much information you've gained compared to the previous situation.

Mutual information is a way of summarizing much knowing the value of one random variable tells you about another random variable (Shannon + Weaver 1949). If X and Y are independent, the mutual information is zero. Any correlations (positive, negative, or nonlinear) will result in positive mutual information. It turns out the mutual information of X and Y is nicely expressed in terms of entropy (H):

$$\mathrm{MI}(X, Y) = H(X) + H(Y) - H(X, Y).$$

Mutual information is the difference in predictability that you get from knowing the joint distribution $p(X, Y)$ compared to knowing only the two marginal distributions $p(X)$ and $p(Y)$.

Information value (IV), which we will define below, is quite closely connected to mutual information (MI), and both are intended to do the same thing: summarize how much knowing the value of X in a given trial helps you predict the value of Y for the same trial.

Defining WOE and IV

Let's say that we have a binary dependent variable Y and a set of predictive variables $X_1, , X_p$. 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:

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

where $f(X_j|Y)$ denotes the conditional probability density function (or a discrete probability distribution if X_j is categorical).

This relationship says that the conditional logit of $P(Y=1)$, 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 above 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.

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

$$\log \frac{P(Y=1|X_1,\dots,X_p)}{P(Y=0|X_1,\dots,X_p)} = \log \frac{P(Y=1)}{P(Y=0)} + \sum_{j=1}^{p} \log \frac{f(X_j|Y=1)}{f(X_j|Y=0)}.$$

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, which is a highly optimistic assumption.

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

$$\log \frac{P(Y=1|X_1,\dots,X_p)}{P(Y=0|X_1,\dots,X_p)} = \log \frac{P(Y=1)}{P(Y=0)} + \sum_{j=1}^{p} \beta_j \log \frac{f(X_j|Y=1)}{f(X_j|Y=0)},$$

thus partly relaxing the assumption that all predictors in the model are independent. It should be noted that the underlying WOE vectors are still estimated univariately and that the coefficients merely function as scalars. For a more general model, GAM is a great choice (see [5]).

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:

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

Note that the IV is essentially a weighted “sum” of all the individual WOE values where the weights incorporate the *absolute difference* between the numerator and the denominator (WOE captures the relative difference). Generally, if $\mathrm{IV} < 0.05$ the variable has very little predictive power and will not add any meaningful predictive power to your model.

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 estimate.

Here is how it works: create a k by 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 $B_1, , B_k$ denote the bins for X_j, the WOE for X_j for bin i can be written as

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

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

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

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 to buy a product. Thus, 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). (Actual uplift modeling techniques are beyond the scope of this paper; we will stick to model exploration and variable screening.)

The net weight of evidence (NWOE) is the difference between the WOEs for the test group and the control group

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

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

$$10 \times \int \left(f(X_j|Y=1)_t f(X_j|Y=0)_c - f(X_j|Y=1)_c f(X_j|Y=0)_t \right) \times \log \frac{f(X_j|Y=1)_t f(X_j|Y=0)_c}{f(X_j|Y=1)_c f(X_j|Y=0)_t} dX_j.$$

where multiplication of 10 is simply used to make NIV look more like IV. Note that 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. To make the package as efficient as possible aggregations are done in data.table and creation of WOE vectors can be distributed across multiple cores.

There are a number of 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. The `Information` package is specifically designed to perform data exploration by producing easy-to-read tables and graphs.

Data 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. In addition, 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

- `create_infotables()` creates WOE or NWOE tables and outputs a variable-strength summary data.frame (IV or NIV).
- `plot_infotables` creates WOE or NWOE bar charts for one or more variables.

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 cross validation penalty for bin B_i is given by

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

and the total cross validation (CV) penalty is

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

where the probabilities used to form the weights are from the training data. The *adjusted IV* is then defined as

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

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

Note that if a certain bin is not present in the validation data, the penalty will be set to zero for that bin.

Example for a Traditional Binary Classification Problem

Ranking All Variables Using Adjusted IV

```
library(Information)
options(scipen=10)

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


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

### Ranking variables using penalized IV.
## Using ncore=2 because more than 2 is no allowed by CRAN in examples
## For real applications, leave ncore as NULL to get the default
IV <- create_infotables(data=train,
                       valid=valid,
                       y="PURCHASE",
                       parallel=FALSE)
```

[1] "Variable TREATMENT was removed because it has only 1 unique level"

```
knitr::kable(head(IV$Summary))
```

	Variable	IV	PENALTY	AdjIV
41	N_OPEN_REV_ACTS	1.0107695	0.1488150	0.8619545
2	TOT_HI_CRDT_CRDT_LMT	0.9345902	0.0870907	0.8474995
3	RATIO_BAL_TO_HI_CRDT	0.8232539	0.0819107	0.7413432
62	D_NA_M_SNC_MST_RCNT_ACT_OPN	0.6355466	0.0168795	0.6186671
27	M_SNC_OLDST_RETAIL_ACT_OPN	0.5573438	0.0887484	0.4685954
1	M_SNC_MST_RCNT_ACT_OPN	0.5026402	0.0726122	0.4300280

```
### Note that the elements of the list IV$Tables are named according to the variable names
names <- names(IV$Tables)
```

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.

```
knitr::kable(IV$Tables$N_OPEN_REV_ACTS)
```

N_OPEN_REV_ACTS	N	Percent	WOE	IV	PENALTY
[0,0]	1469	0.2954545	-2.0465968	0.6401443	0.0403333
[1,2]	958	0.1926790	-0.5900120	0.6958705	0.0415873
[3,3]	310	0.0623492	0.2033085	0.6986029	0.0471586
[4,5]	583	0.1172566	0.4419768	0.7244762	0.0707716
[6,8]	632	0.1271118	0.6148243	0.7810611	0.0824800
[9,11]	453	0.0911102	0.8815772	0.8692672	0.0836600
[12,48]	567	0.1140386	0.9883818	1.0107695	0.1488150

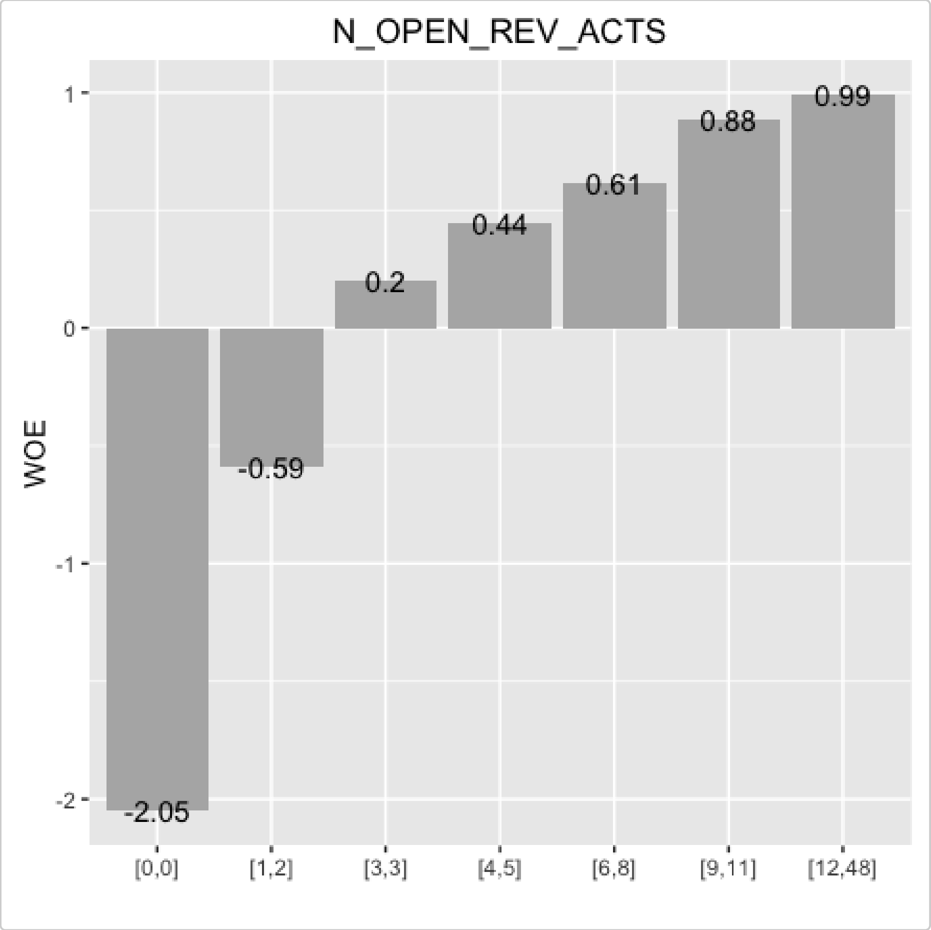
The table shows that the odds of `PURCHASE=1` increases as `N_OPEN_REV_ACTS` increases, although the relationship is not linear.

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

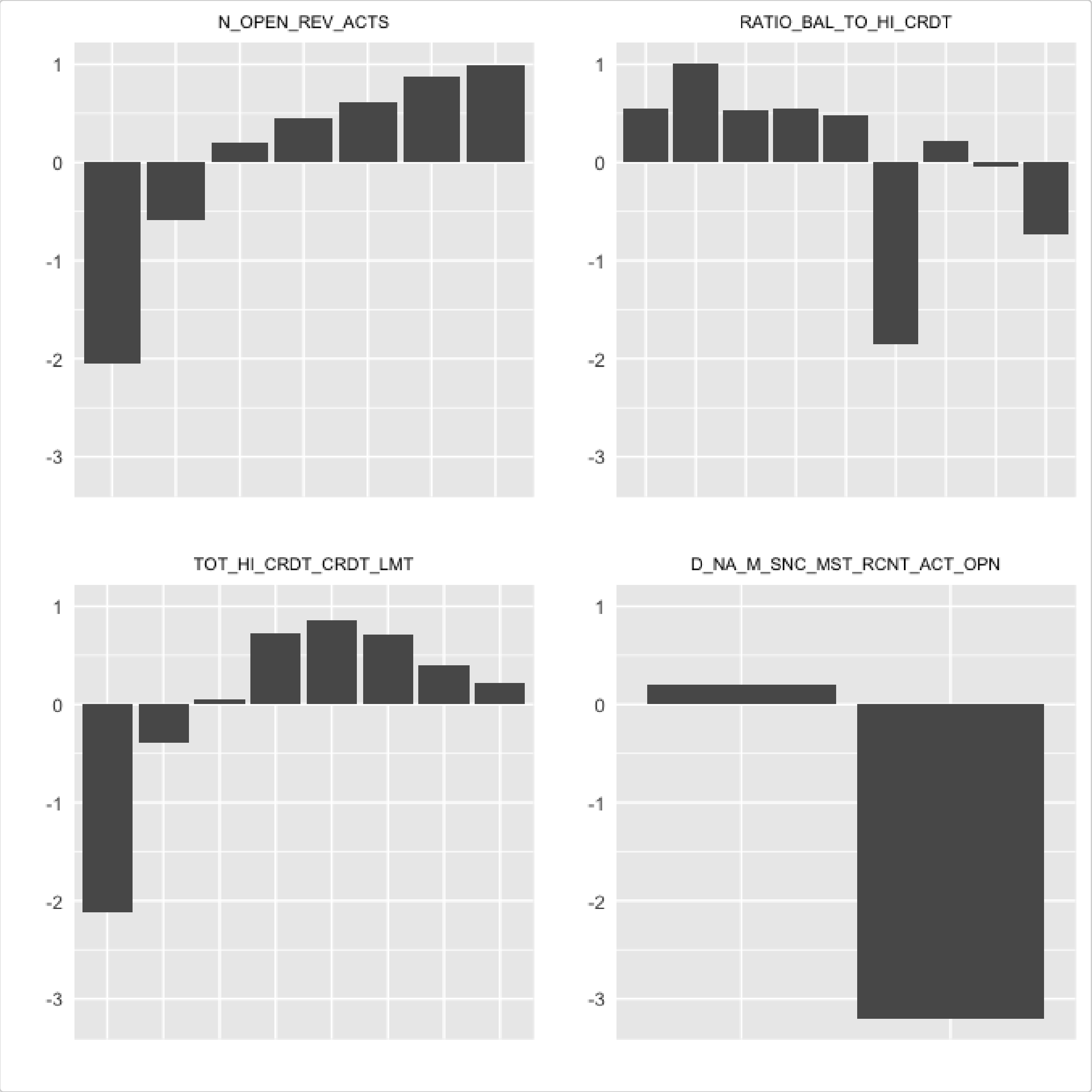
We can also plot this pattern for better visualization:

```
plot_infotables(IV, "N_OPEN_REV_ACTS", show_values=TRUE)
```



If we input a vector of variables, plot_infotables() will plot multiple variables on the same page for comparison purposes. If we are plotting more than nine variables,it will simply spread the plots over multiple pages. Here we are plotting the top four variables:

```
plot_infotables(IV, IV$Summary$Variable[1:4], same_scales=TRUE)
```



If the goal is to plot multiple variables individually, as opposed to a comparison-grid, we can loop through the variable names and create individual plots:

```
# Get the names and loop through to create individual plots
names <- names(IV$Tables)
plots <- list()
for (i in 1:length(names)){
   plots[[i]] <- plot_infotables(IV, names[i])
}

# Showing the top 18 variables
plots[1:4]
```

Omitting Cross Validation

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

```
IV <- create_infotables(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).

```
# Note that this example is running on parallel mode
IV <- create_infotables(data=train,
                   valid=valid,
                   y="PURCHASE",
                   bins=20)
```

Uplift Example

For an uplift model we have to include both the test group and the control group in our dataset:

```
## Read the datasets, including the control group this time
data(train, package="Information")
data(valid, package="Information")
```

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

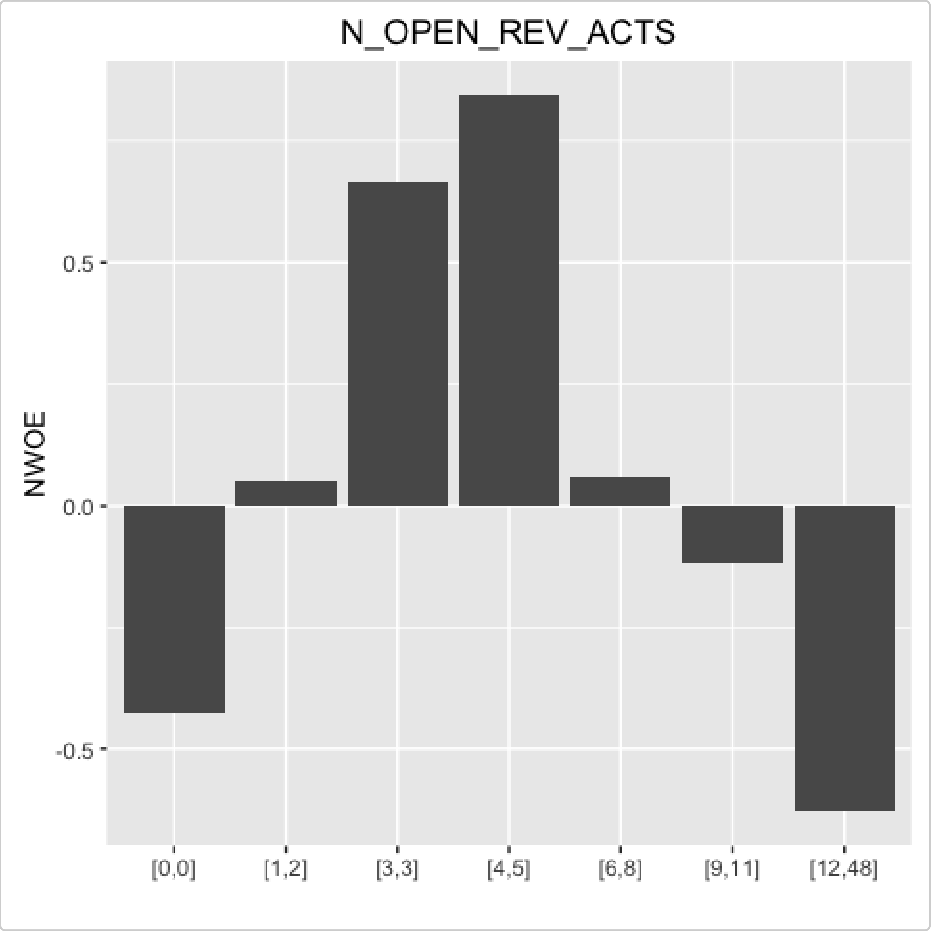
```
## This time use cross validation
NIV <- create_infotables(data=train,
                  valid=valid,
                  y="PURCHASE",
                  trt="TREATMENT",
                  parallel=FALSE)

knitr::kable(head(NIV$Summary))
```

	Variable	NIV	PENALTY	AdjNIV
41	N_OPEN_REV_ACTS	0.2345361	0.0797661	0.1547700
65	D_NA_M_SNC_OLDST_MRTG_ACT_OPN	0.0914100	0.0041172	0.0872928
14	M_SNC_MSTRCNT_MRTG_ACT_UPD	0.0907282	0.0035561	0.0871721
11	M_SNC_MSTREC_INSTL_TRD_OPN	0.0962297	0.0252603	0.0709694
49	MRTG_2_CURRENT_BAL	0.0703045	0.0050797	0.0652248
48	MRTG_1_MONTHLY_PAYMENT	0.1383729	0.0845569	0.0538160

```r
knitr::kable(NIV$Tables$N_OPEN_REV_ACTS)
```

N_OPEN_REV_ACTS	Percent	Treatment	Control	NWOE	WOE_t	WOE_c	NIV	PENALTY
[0,0]	0.2931	1469	1462	-0.4252674	-2.0465968	-1.6213294	0.0360646	0.0453806
[1,2]	0.1919	958	961	0.0515990	-0.5900120	-0.6416109	0.0367067	0.0464948
[3,3]	0.0672	310	362	0.6668728	0.2033085	-0.4635643	0.0551114	0.0534990
[4,5]	0.1176	583	593	0.8429192	0.4419768	-0.4009424	0.1543011	0.0553695
[6,8]	0.1288	632	656	0.0605951	0.6148243	0.5542292	0.1551147	0.0569302
[9,11]	0.0885	453	432	-0.1198633	0.8815772	1.0014406	0.1567903	0.0569801
[12,48]	0.1129	567	562	-0.6261977	0.9883818	1.6145795	0.2345361	0.0797661

```r
plot_infotables(NIV, "N_OPEN_REV_ACTS")
```



Note that we cannot compare the scales of NIV and IV. Moreover, there is no 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. Moreover, note that 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.

Combining IV Analysis With Variable Clustering

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.

In R, we can do this with the `ClustOfVar` package.

```
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"))],init=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$Rank <- stats::ave(-clusters$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.
print(selected_members, row.names=FALSE)
```

Summary

- 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 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 information value originates from information theory and is closely related to the concept of *mutual information (see [3], [4]).
- 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 and Sons, 1959.

[3] Shannon, C.E., A Mathematical Theory of Communication, Bell System Technical Journal, 1948.

[4] Shannon, CE. and Weaver, W. The Mathematical Theory of Communication. Univ of Illinois Press, 1949.

[5] GAM: the Predictive Modeling Silver Bullet, (via)

Other Packages that support information theory

woe package, (link)

klaR package, (link)

infotheo package, (link)