Exercise 10

Business Analytics and Data Science WS16/17

Weight of Evidence (WoE)

The *weight of evidence* is a supervised (i.e. based on the target variable) approach to project a categorical variable with many levels onto one numeric variable. The general idea is to replace each level by a numeric value indicating which class this level more likely to be associated with. Since this is a supervised technique, overfitting is a potential problem. It is advisable to split a woe training set from the overall training data to calculate the woe values. Because the loans data set is very small, we will refrain from doing so in this exercise.

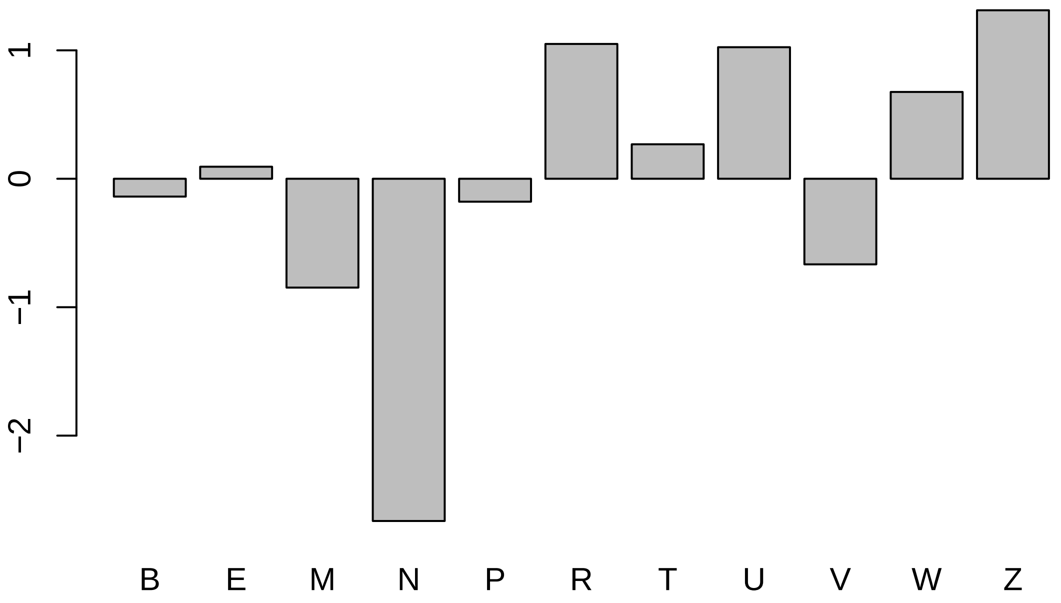
1. Calculate weight of evidence for variables **EMPS_A** and **RES** on the **woe.data** using function **woe()** from package **klaR**. Remember that the formula for the woe of each factor level is

$$woe_{level} = ln\left(\frac{p(BAD)_{level}}{p(GOOD)_{level}}\right).$$

   Because both the denominator and the argument to the natural log cannot be 0, the math will require at least one observation for each combination of level and target category to work. Use argument **zeroadj = 0.5** to correct with an *additive constant to be added for a level with 0 observations in a class* (see R help).
2. Plot the weight of evidence for the levels of **EMPS_A**. It is always advisable to check created data for plausibility. Do the weights generally fit your expectations?

```
## $good
##   B   E   M   N   P   R   T   U   V   W   Z
##  16  71  13   4 303  39  66   4 141  17   3
##
## $bad
##  B  E  M  N  P  R  T  U  V  W  Z
##  5 28  2  0 91 40 31  4 26 12  4
```

```
## $EMPS_A
##           B           E           M           N           P           R
##  0.13895242 -0.09372302  0.84760379  2.66468106  0.17867491 -1.04951620
##           T           U           V           W           Z
## -0.26853085 -1.02419839  0.66646496 -0.67589170 -1.31188046
##
## $RES
##           F           N           O           P           U
##  0.04802697 -0.94122822  0.02892440  0.13284296  0.09673071
```



Filter

Filter approaches determine the value of each variable individually and are usually fast to compute. Which scoring method to use depends on the type of the target variable and the variables used for prediction. We'll look at the Fisher score and information value, two filter for categorical target variables.

- Define a function **fisherScore()** that calculates the Fisher score based on one numeric vector and a factor vector of the classes.

$$FisherScore = \frac{|\bar{x}_G| - |\bar{x}_B|}{\sqrt{s_G^2 + s_B^2}}$$

- Apply the score function to calculate the Fisher score for all numeric variables in the data set.
- Use function **woe()** from package **klaR** to calculate the weight of evidence of the categorical variables in the data set. The resulting object includes the information value for each of the variables calculated as

$$IV = \sum_{cat} \left((p(BAD)_{cat} - p(GOOD)_{cat}) \cdot WOE_{cat} \right).$$

- Which variables should we drop based on these results? NOTE: There are some rules of thumb in the lecture slides.

Information Value

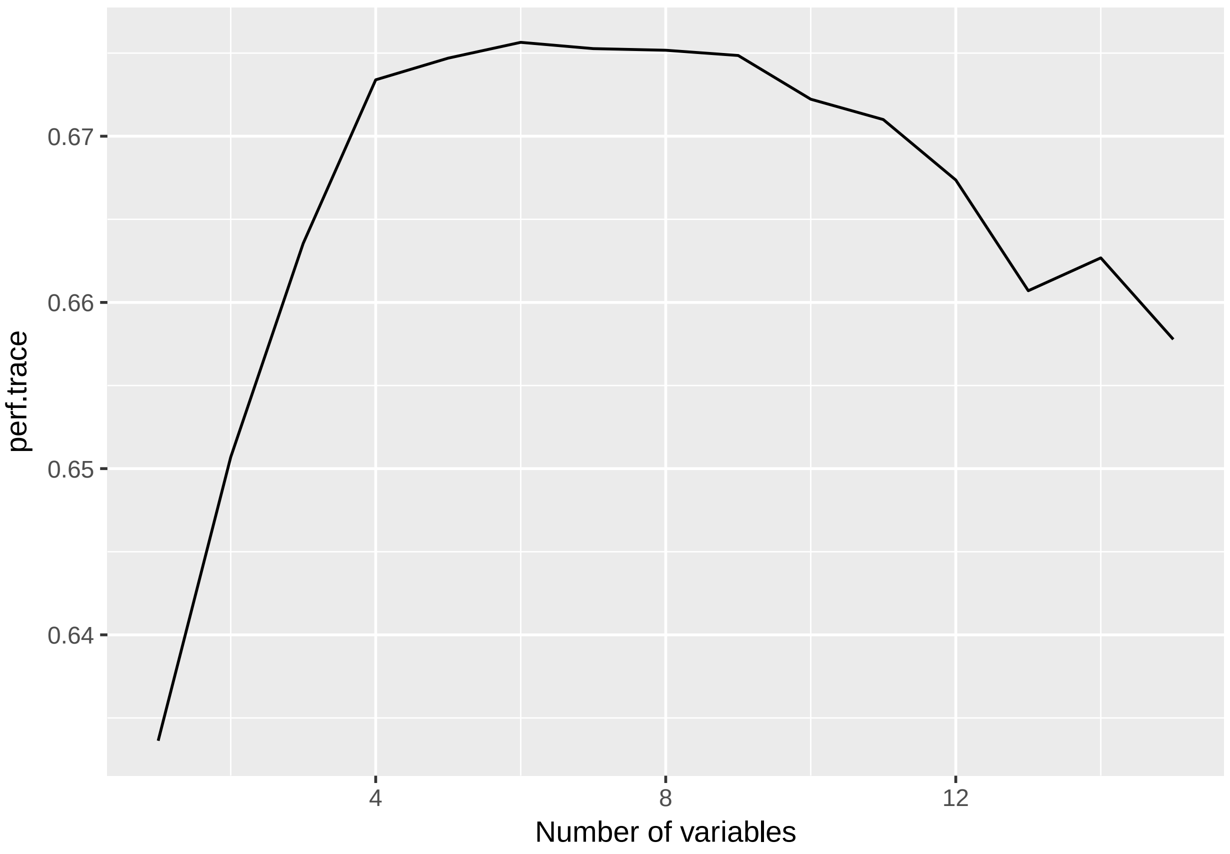
```
##     EMPS_A        RES
## 0.25839900 0.05959541
```

Wrapper

We will build a simple piece of code to do stepwise forward selection. We will then start with an empty model and consider the performance of models including each of the variables. We will keep the model with the best variable and in the next step consider each of the remaining variables in addition to the one that we selected already. We repeat this process until including any of the remaining variables does not increase the performance further.

Write this approach down in pseudo-code to make a blueprint that we can translate into real R code. Really, write it down.
One formulation could be: - Create a vector of all variables S not used in the model and P used in the model (empty at beginning) - For at most as many rounds as there are variables: - For each variable s in set of variables S: - Train model on all variables in current P and *including* s and make prediction - Save performance measure - If performance of one or more variables is better than best previous performance: - Select best-performing variable from P and add variable to set P Else: - Stop - Output set of variables to include P

Write some code to implement the search procedure for a model of your choice. See also mlr's implementation of sequential backward search (sbs) for an efficient implementation.



```
## [1] "dINC_A"  "RES"     "dINC_SP" "dOUTCC"  "dHVAL"   "nDEP"
```