# Classification and Variable Selection Exercise

## Describing the data set

```
load("sample_data.RData")

summary(A)
```

```
##   Length Class      Mode
## X 200    data.frame list
## Y  77    factor     numeric
```

The data set "A" is a list with X (a data frame) and Y (a factor).

We have **77 observations**.

```
summary(A$Y)
```

```
## -1  1
## 40 37
```

The **target variable Y** is categorical, with two possible levels: -1 and 1.

The data set is balanced, i.e., half the observations have Y = -1 and the other half Y = 1.

In **X,** we have **200 explanatory variables**, labelled V1, V2, . . . , V200. All are numeric and with different ranges (*summary omitted*). No missing values detected.

I will place both Y and the X variables in the same data frame.

```
data_ad = A$X
data_ad$Y <- A$Y
```

## Methodology

This is a Classification problem. Due to the high number of candidates for explanatory variables, variable selection will be performed using Random Forests with the VSURF package. Then, Logistic Regression and Classification Tree models will be used to make predictions. I will compare the two models by calculating the accuracy, which is given by

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}.$$

Before starting, the data set will be split into Learning and Test sets.

## Splitting the data set

The models will be trained using the Learning set. Around 20% of the data will be kept separately on the Test set to test the models' predictions.

```r
splitProb <- c(0.8,0.2)
splitNames <-c("Learning","Test")

n = nrow(x=data_ad)

splitVector<- sample( x=splitNames, size=n, prob=splitProb, replace=TRUE )

table(splitVector)/n
```

```
## splitVector
##  Learning      Test
## 0.8051948 0.1948052
```

```r
#getting the indeces
indeces<-list(
  learning = which(x=(splitVector=="Learning")),
  test=which(x=(splitVector=="Test"))
  )
#splitting the data set
learningSet<-data_ad[indeces$learning,]
testSet<-data_ad[indeces$test,]
```

```r
summary(learningSet$Y)
```

```
## -1  1
## 29 33
```

## Variable Selection

Since there are 200 different explanatory variables, I will use variable selection using Random Forest to detect the most relevant variables for prediction.

```r
library('VSURF')
#Three steps variable selection procedure based on random forests for
#supervised classification and regression problems.
#First step ("thresholding step") is dedicated to eliminate irrelevant
#variables from the dataset. Second step ("interpretation step")
#aims to select all variables related to the response for interpretation purpose.
#Third step ("prediction step") refines the selection by eliminating redundancy in
#the set of variables selected by the second step, for prediction purpose.
set.seed(221921186)
Vy<-VSURF(Y~.,data=learningSet)
```

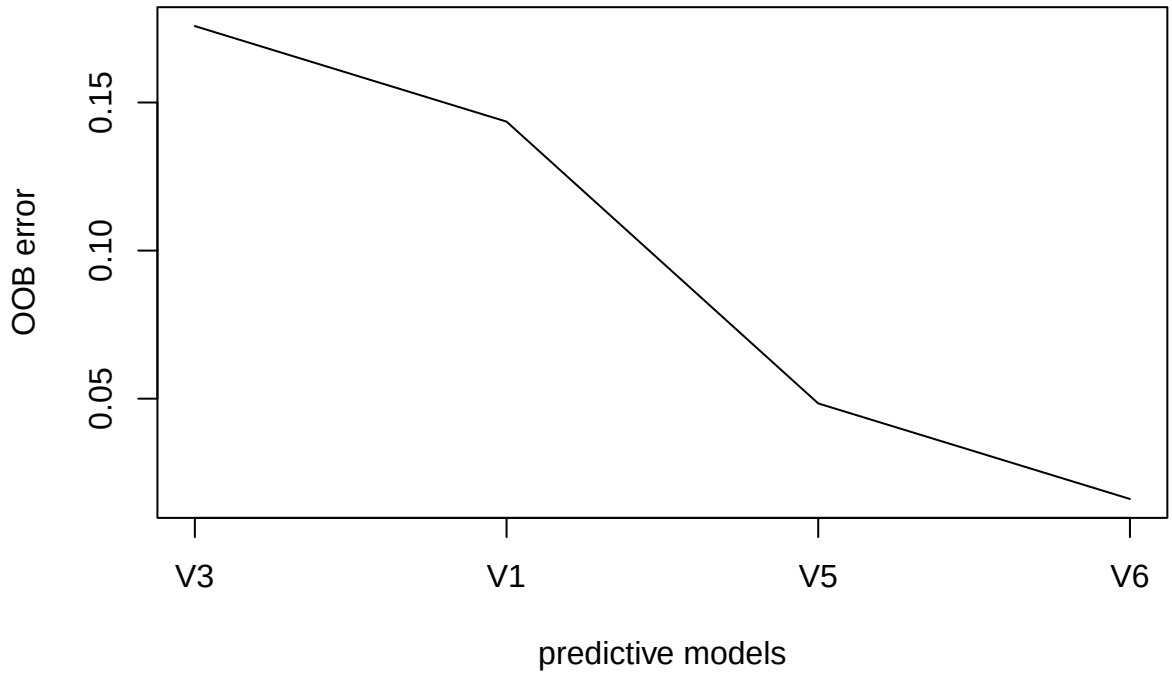
```
## Thresholding step
## Estimated computational time (on one core): 3.8 sec.
##   |                                                                      |
## Interpretation step (on 17 variables)
## Estimated computational time (on one core): between 3.4 sec. and  0 sec.
##   |                                                                      |
## Prediction step (on 5 variables)
## Maximum estimated computational time (on one core): 0 sec.
##   |                                                                      |
```

```
## Warning in VSURF.formula(Y ~ ., data = learningSet): VSURF with a formula-type call outputs selected
## which are indices of the input matrix based on the formula:
```

```
## you may reorder these to get indices of the original data
summary(Vy)
```

```
##
##  VSURF computation time: 5 secs
##
##  VSURF selected:
##  17 variables at thresholding step (in 4 secs)
##  5 variables at interpretation step (in 0.9 secs)
##  4 variables at prediction step (in 0.2 secs)
```

```
plot(Vy,step="pred",var.names=TRUE)
```




Explanatory variables selected at prediction step:

```
variables = c()
for (i in Vy$varselect.pred){
  variables <- c(variables,colnames(learningSet)[i])
}
variables
```

```
## [1] "V3" "V1" "V5" "V6"
```

Correlation matrix:

```
cor(learningSet[,variables])
```

```
##            V3        V1        V5         V6
## V3 1.00000000 0.6423799 0.1179416 0.03008126
```

```
## V1 0.64237993 1.0000000 0.2020467 0.17818782
## V5 0.11794158 0.2020467 1.0000000 0.63452942
## V6 0.03008126 0.1781878 0.6345294 1.00000000
```

There is still high correlation between some variables selected.

I will increase the "nmj" in the VSURF function, which is a multiplicative constant that can be used to modulate the threshold.

```
library('VSURF')
set.seed(221921186)
Vy<-VSURF(Y~.,data=learningSet, nmj = 10)
```

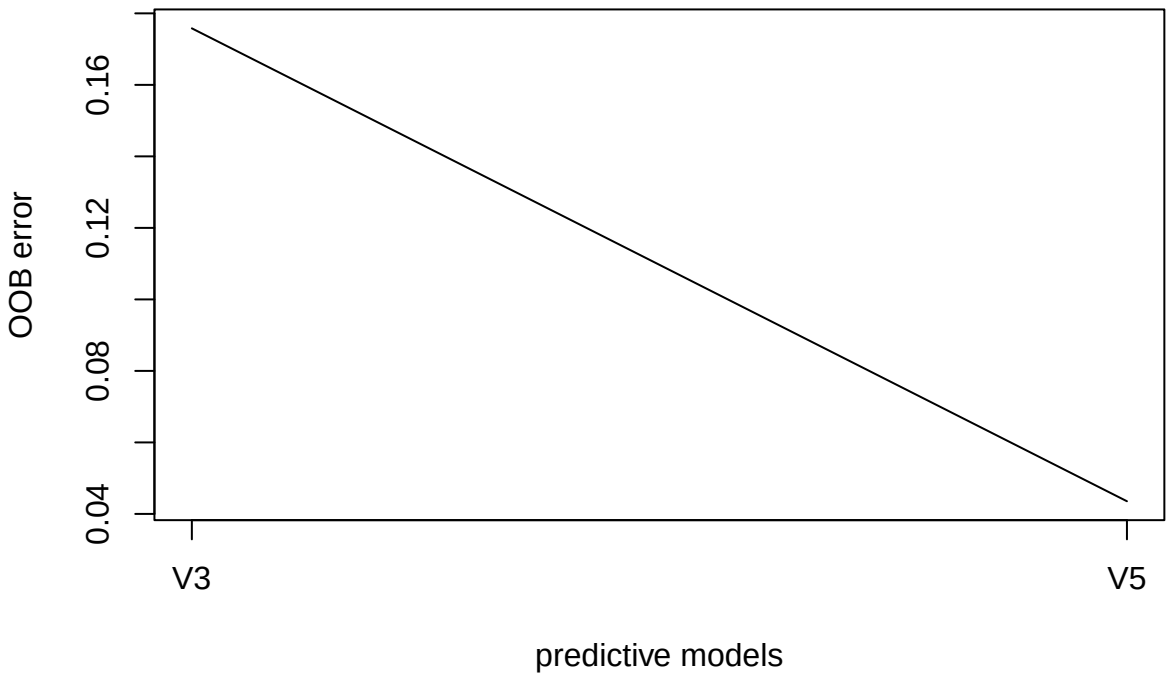
```
## Thresholding step
## Estimated computational time (on one core): 3.4 sec.
##   |                                                                              |
## Interpretation step (on 17 variables)
## Estimated computational time (on one core): between 0 sec. and  0 sec.
##   |                                                                              |
## Prediction step (on 5 variables)
## Maximum estimated computational time (on one core): 0.5 sec.
##   |                                                                              |
```

```
## Warning in VSURF.formula(Y ~ ., data = learningSet, nmj = 10): VSURF with a formula-type call outputs
## which are indices of the input matrix based on the formula:
## you may reorder these to get indices of the original data
```

```
summary(Vy)
```

```
##
##  VSURF computation time: 5 secs
##
##  VSURF selected:
##  17 variables at thresholding step (in 3.9 secs)
##  5 variables at interpretation step (in 0.8 secs)
##  2 variables at prediction step (in 0.2 secs)
```

```
plot(Vy,step="pred",var.names=TRUE)
```




Explanatory variables selected this time:

```
variables = c()
for (i in Vy$varselect.pred){
  variables <- c(variables,colnames(learningSet)[i])
}
variables
```

```
## [1] "V3" "V5"
```

Adding the target variable to the list of variables:

```
variables <- c(variables, "Y")
```

New Learning Set:

```
learningSet= learningSet[,variables]
head(learningSet)
```

```
##            V3         V5 Y
## 45  1.0540762 -0.2922920 1
## 97 -0.3334613  1.5424999 1
## 65  1.1580159 -0.7351499 1
## 12  1.0059907  0.4954647 1
## 72  0.4349800  1.1937084 1
## 28  0.6158743  0.1034573 1
```

New Test Set:

```
testSet= testSet[,variables]
head(testSet)
```

```
##            V3         V5  Y
## 27  1.3118283 -1.0541512  1
## 60  1.1860290  0.6211801  1
## 79 -0.1773634  1.3850978  1
## 81 -0.3707854  0.6605784  1
## 42 -1.2495696  0.2225529 -1
## 69 -1.4738439 -0.2783526 -1
```

## Logistic Regression

Note: Some combinations for explanatory variables, such as V3 and V6, may sometimes lead to a warning by the glm function, saying that the algorithm did not converge and that fitted probabilities are numerically 0 and 1. This may happen when the variables chosen perfectly explain the target variable. In this case, predicted values perfectly match the real test values and accuracy is 100%.

```
model_lr <- glm(Y~., data = learningSet, family=binomial)
```

```
## Warning: glm.fit: algorithm did not converge
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
pred_lr = predict(model_lr, newdata=testSet, type = "response")
```

```
pred_lr <- ifelse(pred_lr >=.5, 1, -1)
pred_lr
```

```
## 27 60 79 81 42 69  6 15 37 77 64 48 78 38 41
##  1  1  1  1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
```

```
confusion_matrix_lr <- table(testSet$Y, pred_lr)
confusion_matrix_lr
```

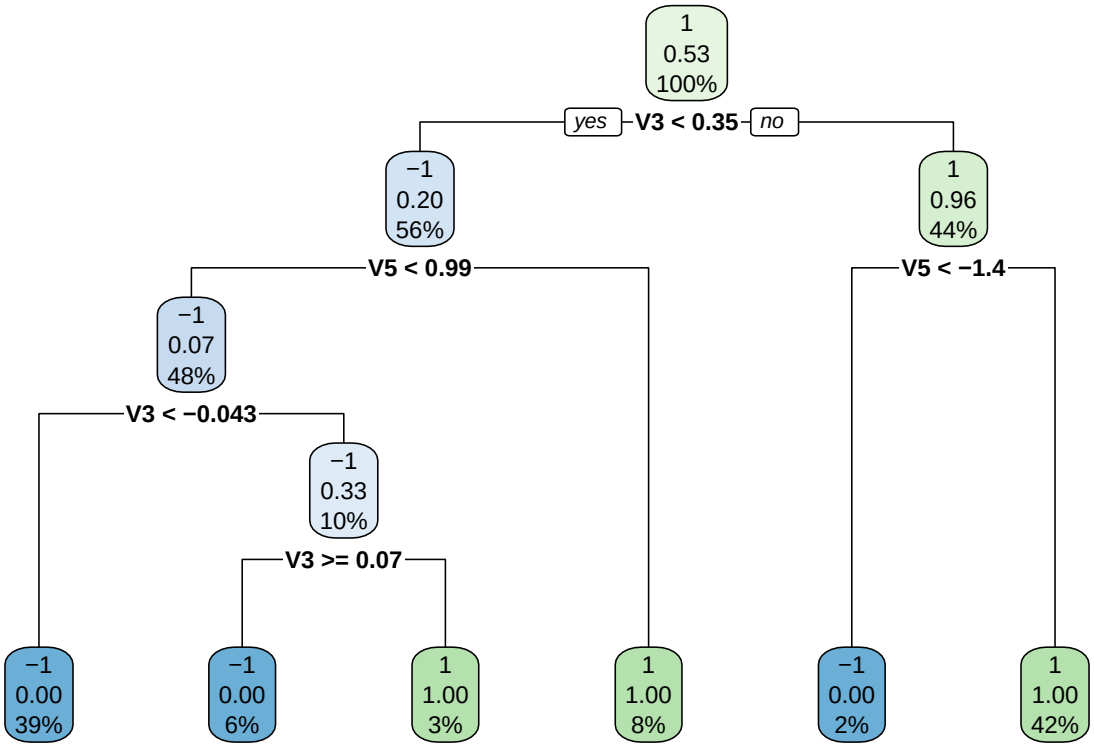
```
##     pred_lr
##      -1  1
##   -1 11  0
##   1   0  4
```

## Classification Tree

First, the maximal tree is obtained, maintaining the complexity parameter CP low.

The minimum number of observations in a node in order that an split is attempted is set as 2.

```
library(rpart)
library(rpart.plot)
#maximal tree
tree_max=rpart(Y~.,data=learningSet,  method = "class", minsplit = 2, cp = 10^(-9))
rpart.plot(tree_max)
```




Then, the maximal tree is pruned using the 1-SE rule.

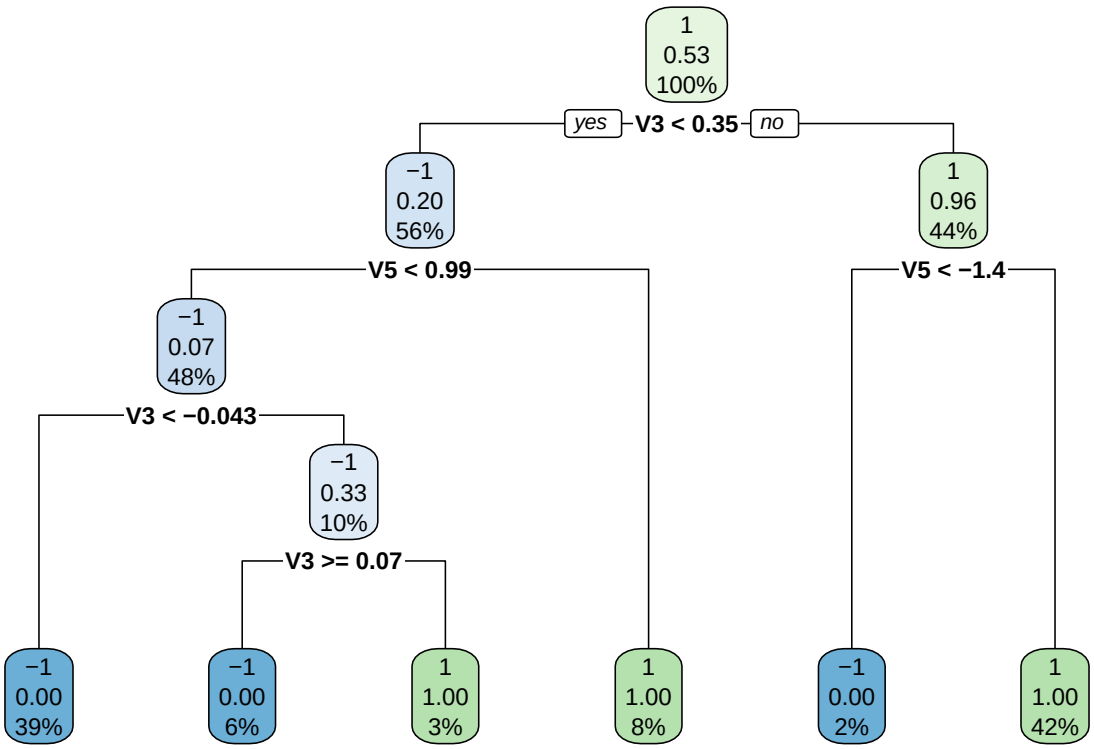
```
finalcart=function(T)
{
  P=printcp(T)
  CV=P[,4] #crossvalidation error (CV)
  a=which(CV==min(CV)) #finding the row with the smallest CV
  s=P[a,4]+P[a,5] #adding the standard deviation - the new threshold used in the 1SE rule
  ss=min(s) #in case s is a vector (several values are the min)
  b=which(CV<=ss)
  d=b[1] #selected value of CP
  Tf=prune(T,cp=P[d,1],  method = "class") #pruning the maximal tree using the selected CP
  finalcart=Tf
}
```

```
tree1 = finalcart(tree_max)
```

```
##
## Classification tree:
## rpart(formula = Y ~ ., data = learningSet, method = "class",
##     minsplit = 2, cp = 10^(-9))
##
## Variables actually used in tree construction:
## [1] V3 V5
##
## Root node error: 29/62 = 0.46774
##
```

```
## n= 62
##
##           CP nsplit rel error   xerror     xstd
## 1 7.2414e-01      0   1.00000 1.000000 0.135476
## 2 1.7241e-01      1   0.27586 0.482759 0.113525
## 3 3.4483e-02      2   0.10345 0.206897 0.080274
## 4 1.0000e-09      5   0.00000 0.068966 0.047973
```

```
rpart.plot(tree1)
```




```
pred_tree<- predict(tree1, method = "class", newdata = testSet)
```

The output is 0 and 1s. Converting it back to the original -1 and 1 levels.

```
predict_tree_converter <- function(predictions_tree, set){
  results = c()
  for (i in 1:nrow(set)){
    if (predictions_tree[i,1] == 0){
      results= c(results,1)
    }
    else{
      results = c(results,-1)
    }
  }
  return(results)
}
```

```
pred_tree = predict_tree_converter(pred_tree, testSet)
```

```
confusion_matrix_tree <- table(testSet$Y, pred_tree)
confusion_matrix_tree
```

```
##     pred_tree
##      -1  1
##   -1 10  1
##   1   1  3
```

## Discussion and Conclusion

Calculating the accuracy of predictions obtained with each model.

```
#function to calculate accuracy
accuracy <- function(results, set){
corrects = 0
for(i in 1:nrow(set)){
  if (results[i] == set$Y[i]){
    corrects = corrects + 1
  }
}
corrects
accuracy = 100*corrects/nrow(set)
return(accuracy)
}
```

Logistic Regression accuracy:

```
accuracy(pred_lr,testSet)
```

```
## [1] 100
```

Classification Tree accuracy:

```
accuracy(pred_tree,testSet)
```

```
## [1] 86.66667
```

The chosen model is the **Logistic Regression** model. The algorithms were ran several times, splitting the data in different ways. While the accuracy of predictions of Classification Tree strongly depends on how the data is split, the accuracy for the Logistic Regression model is much more stable and always close or equal to 100%.

The variable selection was essential. It showed that of 200 variables, only a small number are really relevant. The difficulty is that some of these relevant variables are correlated to each other. For example, V1, V2 and V3 are significantly correlated, and V5 and V6 as well. Therefore, depending of how data is split, the algorithm might select different combinations of these explanatory variables. Most frequently, **V3 and V6**, **V3 and V5** or **V2 and V5** yield the best results.