Exercise 8

Konstantinos Vakalopoulos 12223236

2023-01-11

Premilinary work

First, the data (hcvdat1.csv) were loaded

```
bank <- read.csv2("C:/Users/vaka1/Desktop/bank.csv") #must be modified
```

and missing values were omitted.

```
bank <- na.omit(bank)
```

We transform all the character variables to factors and all int variables to numeric. Furthermore, the value "yes" was replaced by 1 and the value "no" to zero.

```
bank$job <- as.factor(bank$job)
bank$marital <- as.factor(bank$marital)
bank$education <- as.factor(bank$education)
bank$default <- as.factor(bank$default)
bank$housing <- as.factor(bank$housing)
bank$loan <- as.factor(bank$loan)
bank$contact <- as.factor(bank$contact)
bank$month <- as.factor(bank$month)
bank$poutcome <- as.factor(bank$poutcome)


bank$y <- ifelse(bank$y=="yes",1,0)
bank$y <- as.factor(bank$y)
```

```
bank$age <- as.numeric(bank$age)
bank$balance <- as.numeric(bank$balance)
bank$day <- as.numeric(bank$day)
bank$duration <- as.numeric(bank$duration)
bank$campaign <- as.numeric(bank$campaign)
bank$pdays <- as.numeric(bank$pdays)
bank$previous <- as.numeric(bank$previous)
```

Thus, for addition information about the data, the following commands were used

```
summary(bank)
```

```
##       age                 job          marital         education    default
##  Min.   :19.00   management :969   divorced: 528   primary  : 678   no :4445
##  1st Qu.:33.00   blue-collar:946   married :2797   secondary:2306   yes:  76
##  Median :39.00   technician :768   single  :1196   tertiary :1350
##  Mean   :41.17   admin.     :478                   unknown  : 187
##  3rd Qu.:49.00   services   :417
##  Max.   :87.00   retired    :230
##                  (Other)    :713
##     balance      housing     loan           contact          day
##  Min.   :-3313   no :1962   no :3830   cellular :2896   Min.   : 1.00
##  1st Qu.:   69   yes:2559   yes: 691   telephone: 301   1st Qu.: 9.00
##  Median :  444                         unknown  :1324   Median :16.00
##  Mean   : 1423                                          Mean   :15.92
##  3rd Qu.: 1480                                          3rd Qu.:21.00
##  Max.   :71188                                          Max.   :31.00
##
##      month         duration       campaign          pdays
##  may    :1398   Min.   :   4   Min.   : 1.000   Min.   : -1.00
##  jul    : 706   1st Qu.: 104   1st Qu.: 1.000   1st Qu.: -1.00
##  aug    : 633   Median : 185   Median : 2.000   Median : -1.00
##  jun    : 531   Mean   : 264   Mean   : 2.794   Mean   : 39.77
##  nov    : 389   3rd Qu.: 329   3rd Qu.: 3.000   3rd Qu.: -1.00
##  apr    : 293   Max.   :3025   Max.   :50.000   Max.   :871.00
##  (Other): 571
##     previous          poutcome    y
##  Min.   : 0.0000   failure: 490   0:4000
##  1st Qu.: 0.0000   other  : 197   1: 521
##  Median : 0.0000   success: 129
##  Mean   : 0.5426   unknown:3705
##  3rd Qu.: 0.0000
##  Max.   :25.0000
##
```

```
str(bank)
```

```
## 'data.frame':    4521 obs. of  17 variables:
##  $ age      : num  30 33 35 30 59 35 36 39 41 43 ...
##  $ job      : Factor w/ 12 levels "admin.","blue-collar",..: 11 8 5 5 2 5 7 10 3 8 ...
##  $ marital  : Factor w/ 3 levels "divorced","married",..: 2 2 3 2 2 3 2 2 2 2 ...
##  $ education: Factor w/ 4 levels "primary","secondary",..: 1 2 3 3 2 3 3 2 3 1 ...
##  $ default  : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ balance  : num  1787 4789 1350 1476 0 ...
##  $ housing  : Factor w/ 2 levels "no","yes": 1 2 2 2 2 1 2 2 2 2 ...
##  $ loan     : Factor w/ 2 levels "no","yes": 1 2 1 2 1 1 1 1 1 2 ...
##  $ contact  : Factor w/ 3 levels "cellular","telephone",..: 1 1 1 3 3 1 1 1 3 1 ...
##  $ day      : num  19 11 16 3 5 23 14 6 14 17 ...
##  $ month    : Factor w/ 12 levels "apr","aug","dec",..: 11 9 1 7 9 4 9 9 9 1 ...
##  $ duration : num  79 220 185 199 226 141 341 151 57 313 ...
##  $ campaign : num  1 1 1 4 1 2 1 2 2 1 ...
##  $ pdays    : num  -1 339 330 -1 -1 176 330 -1 -1 147 ...
##  $ previous : num  0 4 1 0 0 3 2 0 0 2 ...
##  $ poutcome : Factor w/ 4 levels "failure","other",..: 4 1 1 4 4 1 2 4 4 1 ...
##  $ y        : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
```

Below, it is presented the number of no and yes for the response variable y using the library ggplot2. Based on the histogram, it can be seen that the data are heavily imbalanced

```
library(ggplot2)
```

```
ggplot(bank, aes(x=y)) +
  geom_histogram(stat="count")
```

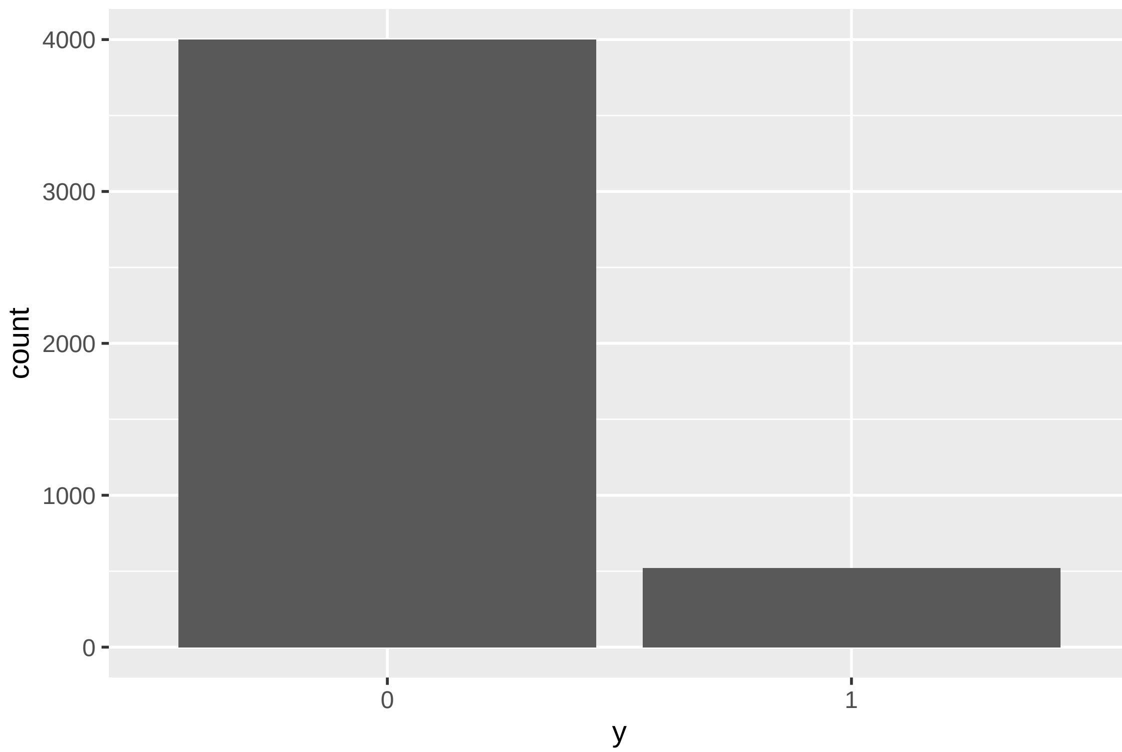

Figure 1: Frequency of the response variable

More information about the data are presented to the plots using the library Hmisc.

```
library(Hmisc)
```

```
hist.data.frame(bank[,which(sapply(bank, is.numeric)==TRUE)])
```

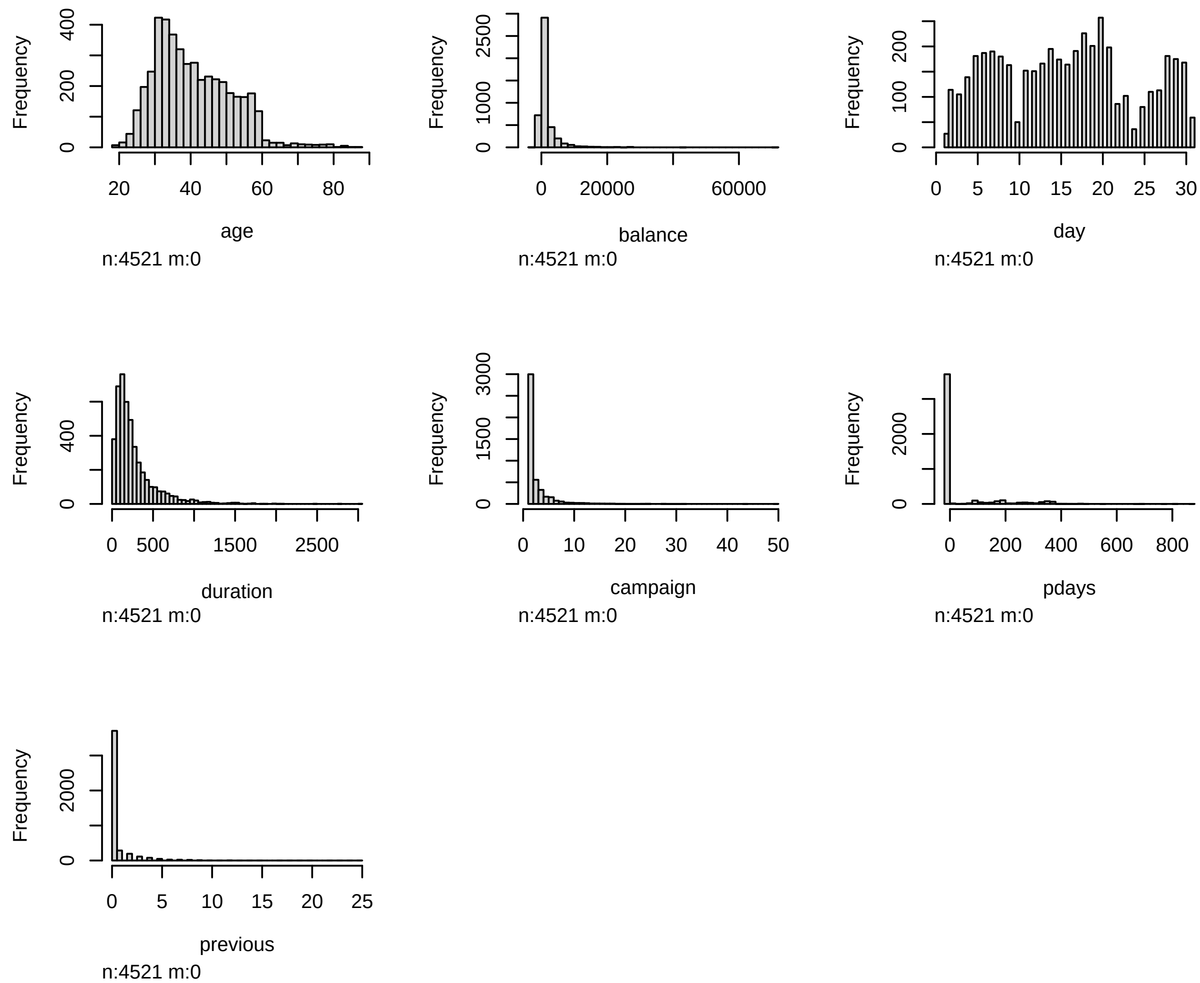

Figure 2: Histograms of the numeric variables

```
hist.data.frame(bank[,which(sapply(bank, is.numeric)==FALSE)])
```

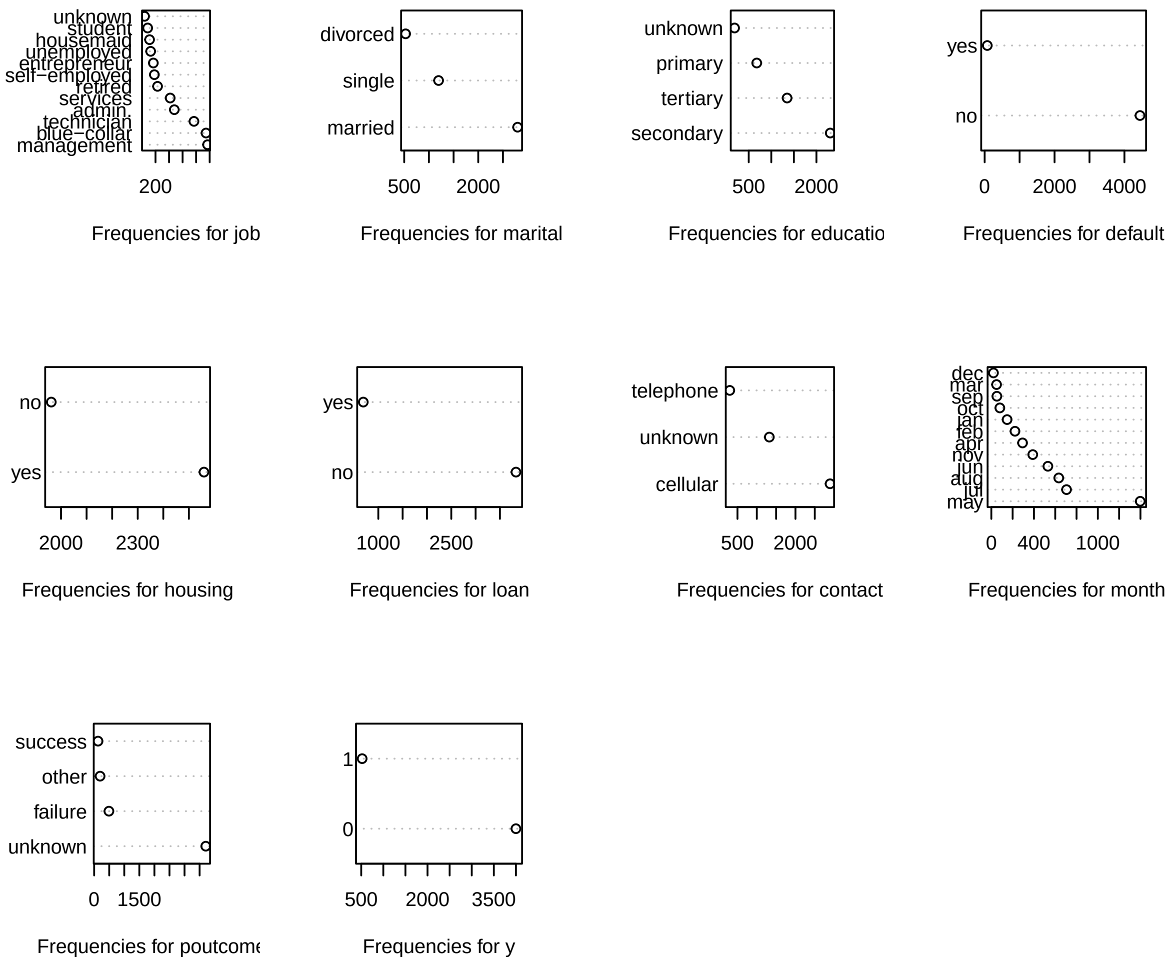

Figure 3: Frequencies of the categorical variables

Question 1(a)

The bank data set was split into 2/3 train and 1/3 test randomly.

```
set.seed(12223236)
n <- nrow(bank)
train <- sample(1:n,round(n*2/3))
test <- (1:n)[-train]
```

In R, from the package rpart, the function rpart() was used in order to compute the initial tree.

```
library(rpart)
model.tree <- rpart(y~.,data=bank, subset=train)
```

Question 1(b)

After creating the initial tree, the functions plot() and text() were used for the representation of the tree.

```
plot(model.tree)
text(model.tree)
```

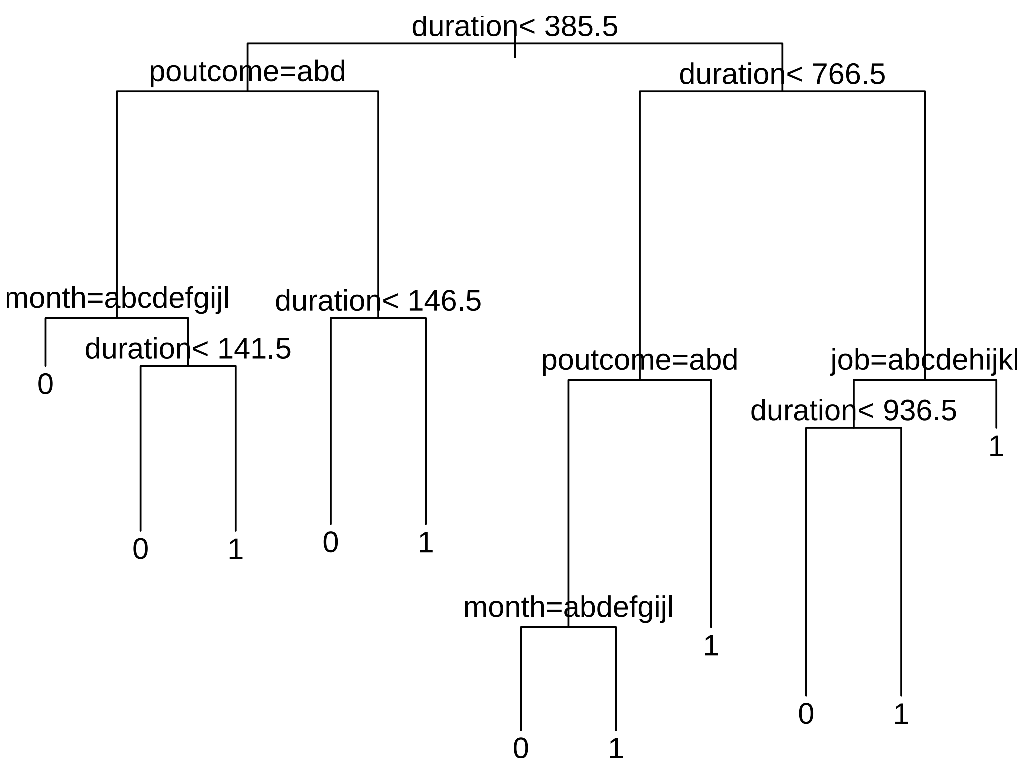

Figure 4: Initial Tree

However, there are wrong labels in the tree. Thus, for convenient purposes, the function prp() from the package rpart.plot was used for the tree representation.

```
rpart.plot::prp(model.tree)
```

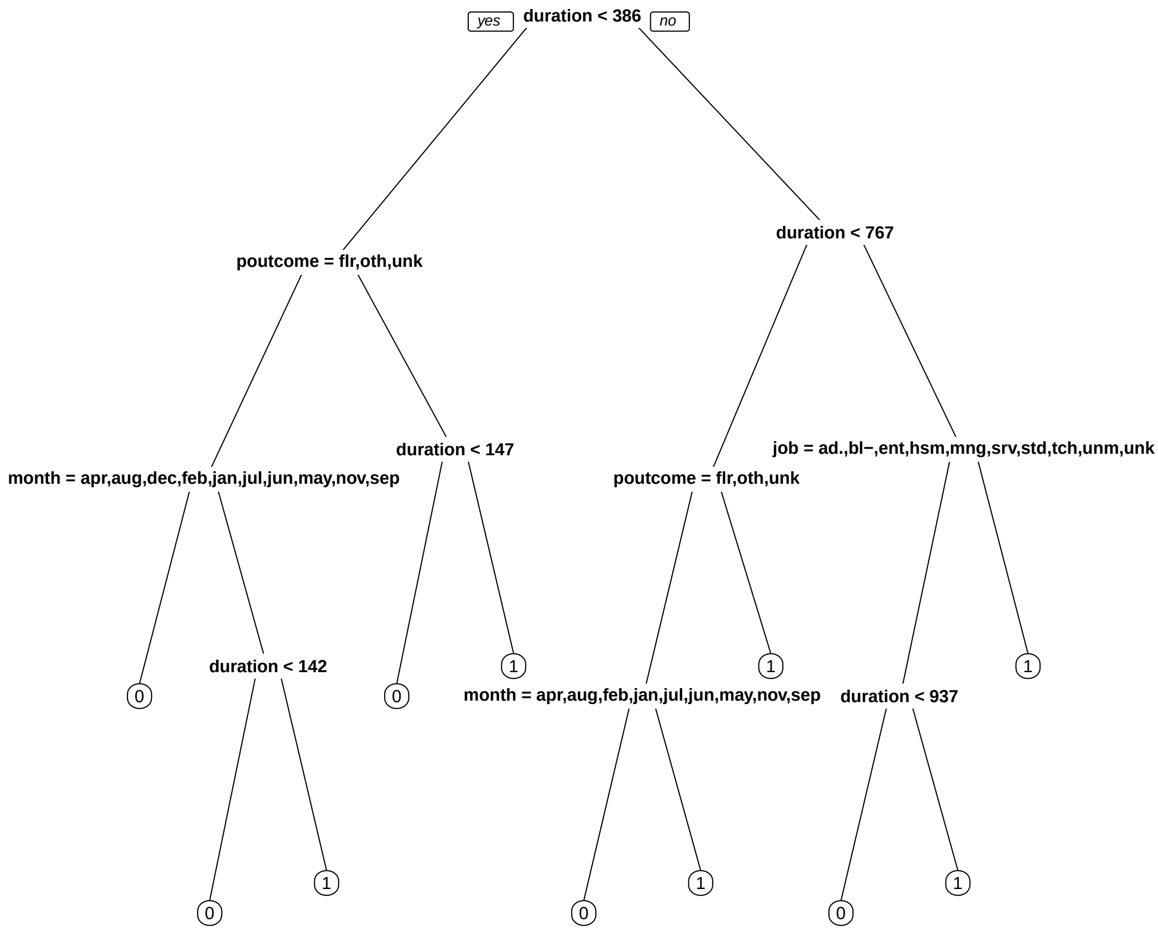

Figure 5: Initial Tree

The tree model was created using only 4 variables: duration, poutcome, month and job. The depth of the tree is 4. We start from the root node, which is the variable duration. This indicates that the variable duration is significant to the model. If duration is lower than 386 we move to the node where we check if the variable poutcome is equal to "failure" or "other" or "unknown". If duration is greater or equals to 386 we move to next node on the right where we check again the value of the variable duration. We continue the same procedure until we reach the leaf nodes. The leaf nodes are responsible for the final classification of the observations. Thus, for the classification of unknown observations, we follow the branches/edges of the tree based, of course, on the conditions/internal nodes until we reach the leaf nodes where the final classification takes place.

Question 1(c)

In this question of the exercise the class variable "y" for test set was predicted and then the misclassification rate was calculated.

```
predicted.classes <- predict(model.tree,newdata=bank[test,], type="class")
actual.classes <- bank[test,]$y
TAB <- table(actual.classes,predicted.classes)
TAB
```

```
##                predicted.classes
## actual.classes    0    1
##               0 1293   36
##               1  128   50
```

```
misclassification.error <- 1-sum(diag(TAB))/sum(TAB)
cat("The misclassification rate is: ", misclassification.error)
```

```
## The misclassification rate is:  0.1088255
```

Question 1(d)

Cross validation was implemented by using the functions printcp() and plotcp() in order to find the optimal cost complexity for a smaller pruned tree.

```
printcp(model.tree)
```

```
##
## Classification tree:
## rpart(formula = y ~ ., data = bank, subset = train)
##
## Variables actually used in tree construction:
## [1] duration job      month    poutcome
##
## Root node error: 343/3014 = 0.1138
##
## n= 3014
##
##         CP nsplit rel error  xerror     xstd
## 1 0.024295      0   1.00000 1.00000 0.050830
## 2 0.018950      5   0.86297 0.89796 0.048481
## 3 0.014577      7   0.82507 0.91545 0.048897
## 4 0.011662      8   0.81050 0.94752 0.049645
## 5 0.010000     10   0.78717 0.94169 0.049510
```

```
optimal.cp <- model.tree$cptable[which.min(model.tree$cptable[,'xerror']),'CP']
optimal.cp
```

```
## [1] 0.01895044
```

The printcp() function provide the cross validation error for each split based on different values of the cost complexity parameter. The one with least cross-validated error (xerror) is the optimal value of CP given by the printcp() function. In our case, the optimal cost complexity is: 0.01895044.

```
plotcp(model.tree)
```

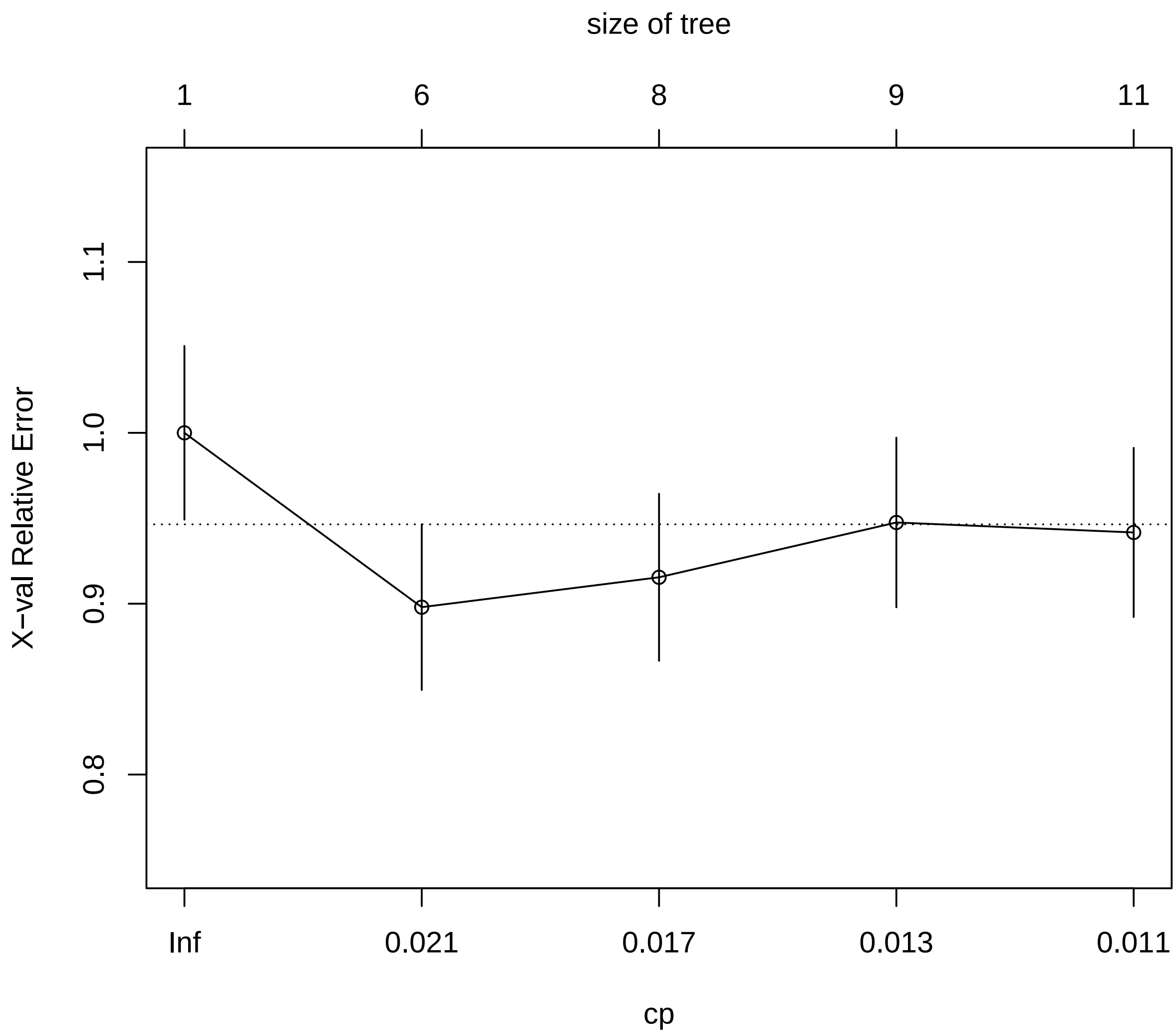

Figure 6: Cost Complexity

A graphical depiction of the cross-validated error summary is provided via plotcp(). To show the deviation, the cp values are plotted against the geometric mean until the smallest value is attained.

Question 1(e)

Based in the question 1(d) the optimal cost complexity is 0.0189 This specific cp value was used for pruning the initial tree using the function prune().

```
model.tree.pruned <- prune(model.tree,cp=optimal.cp)
```

For the tree visualization, the function prp(), as mentioned above, was used.

```
rpart.plot::prp(model.tree.pruned)
```

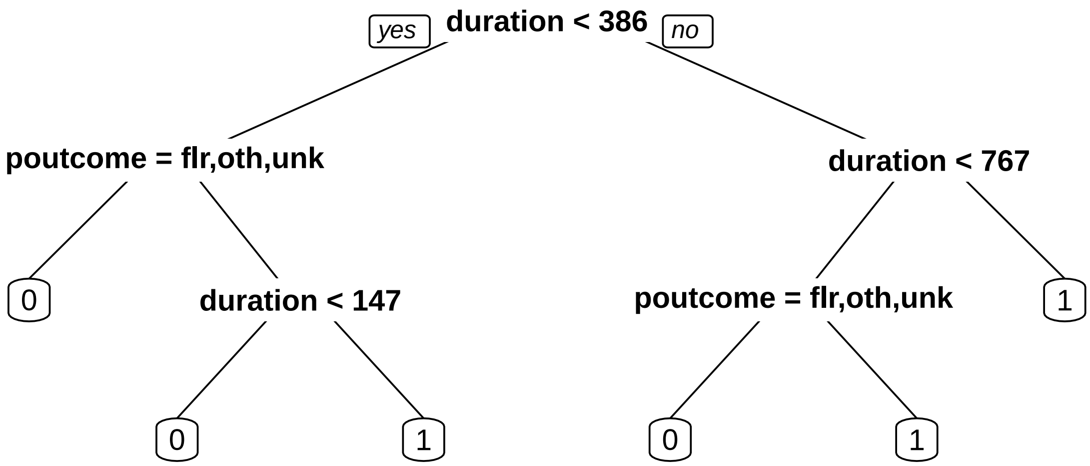

Figure 7: Pruned Tree

The pruned tree is noticeable smaller, with depth equals to 3, compared to the initial tree. The tree model uses only two variables (duration and poutcome) where the variable duration still remains the most important variable. This is due to fact that this variables is in the root of the pruned tree.

Question 1(f)

In this question of the exercise the class variable "y" for test set was predicted and then the misclassification rate was calculated for the pruned tree.

```
predicted.classes <- predict(model.tree.pruned,newdata=bank[test,], type="class")
actual.classes <- bank[test,]$y
TAB <- table(actual.classes,predicted.classes)
TAB
```

```
##                predicted.classes
## actual.classes    0    1
##              0 1293   36
##              1  114   64
```

```r
misclassification.error <- 1-sum(diag(TAB))/sum(TAB)
cat("The misclassification rate is: ", misclassification.error)
```

```
## The misclassification rate is:  0.0995355
```

It is observed that the misclassification rate has been decreased from 0.1088 to 0.0995. Also, based on the two confusion matrices 14 observations from the minority class "1" were predicted correctly from the pruned tree compared to the initial full tree.

Question 1(g)

There are two techniques that can possibly reduce the misclassification rate. The first strategy is to split the data using the stratification technique. With this technique the data set split into train and test set maintaining the same proportions of samples in each class as shown in the original data set. Below, using the caret package, the data were split using the stratification.

```r
model.tree <- rpart(y~.,data=bank, subset=train)
predicted.classes <- predict(model.tree,newdata=bank[test,], type="class")
actual.classes <- bank[test,]$y
TAB <- table(actual.classes,predicted.classes)

misclassification.error <- 1-sum(diag(TAB))/sum(TAB)
cat("The misclassification rate for the intitial tree is: ", misclassification.error)
```

```
## The misclassification rate for the intitial tree is:  0.1088255
```

```r
cat("\n")
```

```r
cat("The misclassification rate for the intitial tree for the yes clients is: ",
    1 - TAB[2,2]/(TAB[1,2]+TAB[2,1]+TAB[2,2]))
```

```
## The misclassification rate for the intitial tree for the yes clients is:  0.7663551
```

```r
cat("\n")
```

```r
set.seed(12223236)
train_str <- caret::createDataPartition(bank$y, times = 1,p=2/3, list=F) #stratified separation
test_str <- (1:n)[-train_str]

model.tree.statified <- rpart(y~.,data=bank, subset=train_str)

predicted.classes <- predict(model.tree.statified,newdata=bank[test,], type="class")
actual.classes <- bank[test,]$y
TAB <- table(actual.classes,predicted.classes)

misclassification.error <- 1-sum(diag(TAB))/sum(TAB)
cat("The misclassification rate for the stratified tree is: ", misclassification.error)
```

```
## The misclassification rate for the stratified tree is:  0.08427339
```

```r
cat("\n")
```

```r
cat("The misclassification rate for the stratified tree for the yes clients is: ",
    1 - TAB[2,2]/(TAB[1,2]+TAB[2,1]+TAB[2,2]))
```

```
## The misclassification rate for the stratified tree for the yes clients is:  0.6165049
```

Therefore, the overall and the "yes" clients misclassification rate has been slightly decreased.

The second strategy to assign weights in each observation based on the class that it belongs. Thus, the weights will depend mainly from the class of the observation. Basically, the weight will be small if the observation belongs to majority class and high if the observation belongs to the minority class.The entire idea of the weights is to penalize the minority class for misclassifying itself by increasing class weight while simultaneously decreasing class weight for the majority class.

The formula for the weight assignments is: wj=n_samples / (n_classes * n_samplesj), where wj is the weight for each class(j signifies the class), n_samples the total number of samples or rows in the dataset, n_classes the total number of unique classes in the target, n_samplesj is the total number of rows of the respective class. Thus the final weights are for the two classes:

```r
w0 <- length(train)/(2*table(bank[train,]$y)[1]) #weight for the 0 or class "no"
w1 <- length(train)/(2*table(bank[train,]$y)[2]) #weight for the 1 or class "yes"
w <- ifelse(bank[train,]$y == 1, w1, w0)

model.tree.weights <- rpart(y~.,data=bank[train,], weights = w)

predicted.classes <- predict(model.tree.weights,newdata=bank[test,], type="class")
actual.classes <- bank[test,]$y
TAB <- table(actual.classes,predicted.classes)

misclassification.error <- 1-sum(diag(TAB))/sum(TAB)
cat("The misclassification rate for the weighted tree is: ", misclassification.error)
```

```
## The misclassification rate for the weighted tree is:  0.2329131
```

```r
cat("\n")
```

```r
cat("The misclassification rate for the weighted tree for the yes clients is: ",
    1 - TAB[2,2]/(TAB[1,2]+TAB[2,1]+TAB[2,2]))
```

```
## The misclassification rate for the weighted tree for the yes clients is:  0.6992032
```

In conclusion, the "yes" clients misclassification rate has been decreased. However, the overall misclassification rate has been increased up to 13% which is a huge a difference compared to the initial tree.

Question 2(a)

From the package randomForest, the function randomForest() was used in order to create a model based on the train set. also, the misclassification rate was calculated.

```
library(randomForest)
```

```
## randomForest 4.7-1.1

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
model.random.forest <- randomForest(y~.,data=bank, subset=train)


predictions <- predict(model.random.forest,newdata=bank[test,], type="class")
actual <- bank[test,]$y
TAB.forest <- table(actual,predictions)
TAB.forest
```

```
##       predictions
## actual    0    1
##      0 1301   28
##      1  125   53
```

```
misclassification.error.forest <- 1-sum(diag(TAB.forest))/sum(TAB.forest)
cat("The misclassification rate is: ", misclassification.error.forest)
```

```
## The misclassification rate is:  0.1015262
```

```
cat("\n")
```

```
cat("The misclassification rate for the yes clients is: ",
    1 - TAB[2,2]/(TAB[1,2]+TAB[2,1]+TAB[2,2]))
```

```
## The misclassification rate for the yes clients is:  0.6992032
```

Question 2(b)

The option importance=TRUE in the function randomForest() was used. Furthermore, the plot and the varImpPlot was used.

```
model.random.forest <- randomForest(y~.,data=bank, subset=train,importance = TRUE)
```

```
plot(model.random.forest)
legend("right", legend = c("OOB Error", "FPR", "FNR"),lty = c(1,2,3), col = c("black", "red", "green"))
```

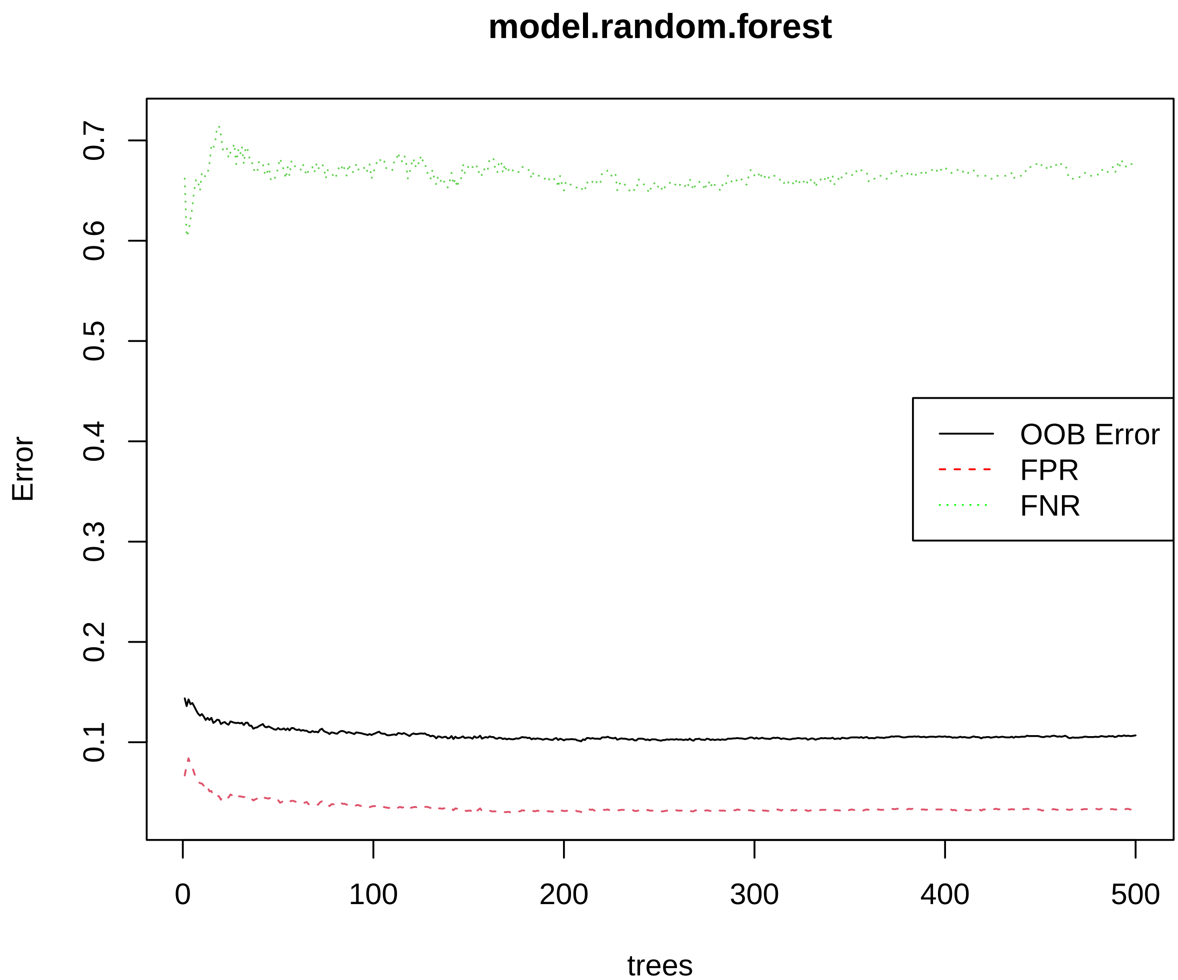

Figure 8: Random Forest Importance

The plot above presents the Out Of Bag error (black line) when the number of trees increase. An interesting thing that can be seen is that the OOB error is stabilized for number of trees equals to 500 which is the default value for the randomForest function. The green line is the False Negative Rate and is high compared to red line, which is the False Positive Rate. The reason why the green line has high error values is due to the imbalance of the data.

```
varImpPlot(model.random.forest)
```

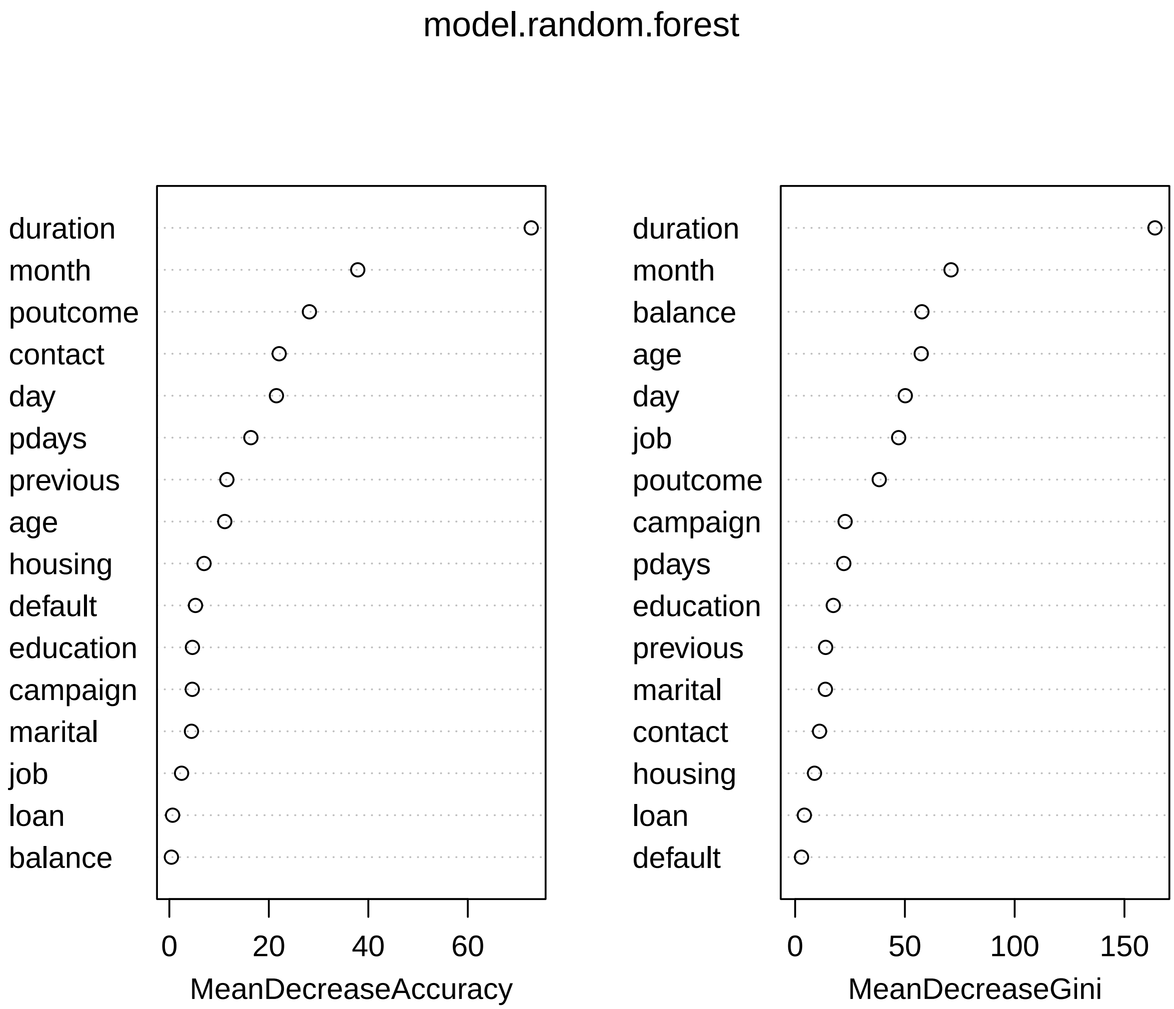

Figure 9: Variable Importance

The left plot, where the x axis is the MeanDecreaseAccuracy, indicates how much the accuracy decreases if we remove that variable. It is observed that removing the variable duration the accuracy decreases by a lot compared to the variable loan. The right plot, where the x axis is the MeanDecreaseGini, indicates the importance of each variable based on the gini impurity index, used for the calculation of splits in tree. The variable duration is the most important variable and it remains the same as in the left plot. However, the least important variable changes and it is the variable default.

Question 2(c)

In this question, we try to improve the misclassification error of the "yes" clients (by keeping the overall misclassificatione error still small) with different strategies. We modify the parameters: sampsize, classwt, cutoff and strata in the randomForest() function.

Sampsize

The parameter sampsize defines the size or the sizes of sample to draw for the creations of the trees. In our case where the data are imbalances, we could define the number of samples that belong to class "yes" or class "no", through the parameter sampsize.

We randomly select 100 samples for the class "no" and 100 samples for the class "yes" and check if the misclassification error for the class "yes" is reduced.

```
model.random.forest <- randomForest(y~.,data=bank, subset=train,sampsize = c(100,100))

predictions <- predict(model.random.forest,newdata=bank[test,], type="class")
actual <- bank[test,]$y
TAB.forest <- table(actual,predictions)
TAB.forest
```

```
##       predictions
## actual    0    1
##      0 1029  300
##      1   20  158
```

```
misclassification.error.forest <- 1-sum(diag(TAB.forest))/sum(TAB.forest)
cat("The misclassification rate is: ", misclassification.error.forest)
```

```
## The misclassification rate is:  0.2123424
```

```
cat("\n")
```

```
cat("The misclassification rate for the yes clients is: ",
    1 - TAB[2,2]/(TAB[1,2]+TAB[2,1]+TAB[2,2]))
```

```
## The misclassification rate for the yes clients is:  0.6992032
```

Classwt

The parameter classwt assigns weights to the two classes. The weights that were used are from the question 1(g)

```
model.random.forest <- randomForest(y~.,data=bank[train,],classwt = c(w0,w1))

predictions <- predict(model.random.forest,newdata=bank[test,], type="class")
actual <- bank[test,]$y
TAB.forest <- table(actual,predictions)
TAB.forest
```

```
##       predictions
## actual    0    1
##      0 1319   10
##      1  151   27
```

```
misclassification.error.forest <- 1-sum(diag(TAB.forest))/sum(TAB.forest)
cat("The misclassification rate is: ", misclassification.error.forest)
```

```
## The misclassification rate is:  0.1068348
```

```
cat("\n")
```

```
cat("The misclassification rate for the yes clients is: ",
    1 - TAB[2,2]/(TAB[1,2]+TAB[2,1]+TAB[2,2]))
```

```
## The misclassification rate for the yes clients is:  0.6992032
```

Cutoff

The cutoff parameter is only used for classfication. More specifically, the cutoff is a A vector of length equal to number of classes. The 'winning' class for an observation is the one with the maximum ratio of proportion of votes to cutoff. Default is 1/k where k is the number of classes (k=2 in our case) (i.e., majority vote wins). We randomly select cutoff = c(0.3,0.7). Thus:

```
model.random.forest <- randomForest(y~.,data=bank, subset=train,cutoff = c(0.3,0.7))

predictions <- predict(model.random.forest,newdata=bank[test,], type="class")
actual <- bank[test,]$y
TAB.forest <- table(actual,predictions)
TAB.forest
```

```
##       predictions
## actual    0    1
##      0 1325    4
##      1  164   14
```

```
misclassification.error.forest <- 1-sum(diag(TAB.forest))/sum(TAB.forest)
cat("The misclassification rate is: ", misclassification.error.forest)
```

```
## The misclassification rate is:  0.1114798
```

```
cat("\n")
```

```
cat("The misclassification rate for the yes clients is: ",
    1 - TAB[2,2]/(TAB[1,2]+TAB[2,1]+TAB[2,2]))
```

```
## The misclassification rate for the yes clients is:  0.6992032
```

Strata

The strata parameter is responsible for the stratified sampling.

```r
model.random.forest <- randomForest(y~.,data=bank[train,],strata = bank$y[train])

predictions <- predict(model.random.forest,newdata=bank[test,], type="class")
actual <- bank[test,]$y
TAB.forest <- table(actual,predictions)
TAB.forest
```

```
##       predictions
## actual    0    1
##      0 1298   31
##      1  121   57
```

```r
misclassification.error.forest <- 1-sum(diag(TAB.forest))/sum(TAB.forest)
cat("The misclassification rate is: ", misclassification.error.forest)
```

```
## The misclassification rate is:  0.1008626
```

```r
cat("\n")
```

```r
cat("The misclassification rate for the yes clients is: ",
    1 - TAB[2,2]/(TAB[1,2]+TAB[2,1]+TAB[2,2]))
```

```
## The misclassification rate for the yes clients is:  0.6992032
```

Overall, non of the above approaches improve the misclassification rate for the yes clients. The error rate remanes the same with all different strategies.