RandomForest

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 4.0.3
```

```
## randomForest 4.6-14
```

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

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.0.5
```

```
## Loading required package: lattice
```

```
## Warning: package 'lattice' was built under R version 4.0.5
```

```
## Loading required package: ggplot2
```

```
## Warning: package 'ggplot2' was built under R version 4.0.5
```

```
##
## Attaching package: 'ggplot2'
```

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

```
library(tidyr)
```

Reading the data

```
data1 <- read.table(file = "C://Users/cs_mo/Downloads/ISYE7406/ProjectCreditCard/creditcards.cs
v", header= TRUE,sep= ",",skip = 1)
names(data1)[25] <- 'default'
head(data1)
```

```
##   ID LIMIT_BAL SEX EDUCATION MARRIAGE AGE PAY_0 PAY_2 PAY_3 PAY_4 PAY_5 PAY_6
## 1  1     20000   2         2        1  24     2     2    -1    -1    -2    -2
## 2  2    120000   2         2        2  26    -1     2     0     0     0     2
## 3  3     90000   2         2        2  34     0     0     0     0     0     0
## 4  4     50000   2         2        1  37     0     0     0     0     0     0
## 5  5     50000   1         2        1  57    -1     0    -1     0     0     0
## 6  6     50000   1         1        2  37     0     0     0     0     0     0
##   BILL_AMT1 BILL_AMT2 BILL_AMT3 BILL_AMT4 BILL_AMT5 BILL_AMT6 PAY_AMT1 PAY_AMT2
## 1      3913      3102       689         0         0         0        0      689
## 2      2682      1725      2682      3272      3455      3261        0     1000
## 3     29239     14027     13559     14331     14948     15549     1518     1500
## 4     46990     48233     49291     28314     28959     29547     2000     2019
## 5      8617      5670     35835     20940     19146     19131     2000    36681
## 6     64400     57069     57608     19394     19619     20024     2500     1815
##   PAY_AMT3 PAY_AMT4 PAY_AMT5 PAY_AMT6 default
## 1        0        0        0        0       1
## 2     1000     1000        0     2000       1
## 3     1000     1000     1000     5000       0
## 4     1200     1100     1069     1000       0
## 5    10000     9000      689      679       0
## 6      657     1000     1000      800       0
```

Removing 167 outliers as identified in the data exploration part

```
out <- boxplot.stats(data1$LIMIT_BAL)$out
out_ind <- which(data1$LIMIT_BAL %in% c(out))
mydata1 <- data1[-out_ind,]
dim(mydata1)
```

```
## [1] 29833    25
```

Cleaning up Marriage and Education feature

```
mydata1$MARRIAGE[mydata1$MARRIAGE == "0"] <- "3"
mydata1$EDUCATION[mydata1$EDUCATION== "6"]<-"4"
mydata1$EDUCATION[mydata1$EDUCATION== "5"]<-"4"
mydata1$EDUCATION[mydata1$EDUCATION== "0"]<-"4"
```

```
mydata1$default[mydata1$default=="0"] <- "ND"
mydata1$default[mydata1$default=="1"] <- "DEF"
```

Removing the ID column…

```
mydata <- mydata1[,2:25]
head(mydata)
```

```
##   LIMIT_BAL SEX EDUCATION MARRIAGE AGE PAY_0 PAY_2 PAY_3 PAY_4 PAY_5 PAY_6
## 1     20000   2         2        1  24     2     2    -1    -1    -2    -2
## 2    120000   2         2        2  26    -1     2     0     0     0     2
## 3     90000   2         2        2  34     0     0     0     0     0     0
## 4     50000   2         2        1  37     0     0     0     0     0     0
## 5     50000   1         2        1  57    -1     0    -1     0     0     0
## 6     50000   1         1        2  37     0     0     0     0     0     0
##   BILL_AMT1 BILL_AMT2 BILL_AMT3 BILL_AMT4 BILL_AMT5 BILL_AMT6 PAY_AMT1 PAY_AMT2
## 1      3913      3102       689         0         0         0        0      689
## 2      2682      1725      2682      3272      3455      3261        0     1000
## 3     29239     14027     13559     14331     14948     15549     1518     1500
## 4     46990     48233     49291     28314     28959     29547     2000     2019
## 5      8617      5670     35835     20940     19146     19131     2000    36681
## 6     64400     57069     57608     19394     19619     20024     2500     1815
##   PAY_AMT3 PAY_AMT4 PAY_AMT5 PAY_AMT6 default
## 1        0        0        0        0     DEF
## 2     1000     1000        0     2000     DEF
## 3     1000     1000     1000     5000      ND
## 4     1200     1100     1069     1000      ND
## 5    10000     9000      689      679      ND
## 6      657     1000     1000      800      ND
```

```
dim(mydata)
```

```
## [1] 29833    24
```

Splitting the data: 85% training and 15% testing

```
set.seed(7406)
flag<- sort(sample(1:29833,4475))
data_train  <- mydata[-flag,]
data_test <- mydata[flag,]
dim(data_train)
```

```
## [1] 25358    24
```

```
dim(data_test)
```

```
## [1] 4475   24
```

```
head(data_train)
```

```
##   LIMIT_BAL SEX EDUCATION MARRIAGE AGE PAY_0 PAY_2 PAY_3 PAY_4 PAY_5 PAY_6
## 1     20000   2         2        1  24     2     2    -1    -1    -2    -2
## 2    120000   2         2        2  26    -1     2     0     0     0     2
## 3     90000   2         2        2  34     0     0     0     0     0     0
## 4     50000   2         2        1  37     0     0     0     0     0     0
## 5     50000   1         2        1  57    -1     0    -1     0     0     0
## 7    500000   1         1        2  29     0     0     0     0     0     0
##   BILL_AMT1 BILL_AMT2 BILL_AMT3 BILL_AMT4 BILL_AMT5 BILL_AMT6 PAY_AMT1 PAY_AMT2
## 1      3913      3102       689         0         0         0        0      689
## 2      2682      1725      2682      3272      3455      3261        0     1000
## 3     29239     14027     13559     14331     14948     15549     1518     1500
## 4     46990     48233     49291     28314     28959     29547     2000     2019
## 5      8617      5670     35835     20940     19146     19131     2000    36681
## 7    367965    412023    445007    542653    483003    473944    55000    40000
##   PAY_AMT3 PAY_AMT4 PAY_AMT5 PAY_AMT6 default
## 1        0        0        0        0     DEF
## 2     1000     1000        0     2000     DEF
## 3     1000     1000     1000     5000      ND
## 4     1200     1100     1069     1000      ND
## 5    10000     9000      689      679      ND
## 7    38000    20239    13750    13770      ND
```

```
table(mydata$default)
```

```
##
##   DEF    ND
##  6617 23216
```

finding the best mtry value: 4

```
set.seed(7406)
#bestmtry <- tuneRF(data_train[,1:23],as.factor(data_train[,24]), stepFactor=1.5, improve=1e-5,
 ntree=500)
#print(bestmtry)
```

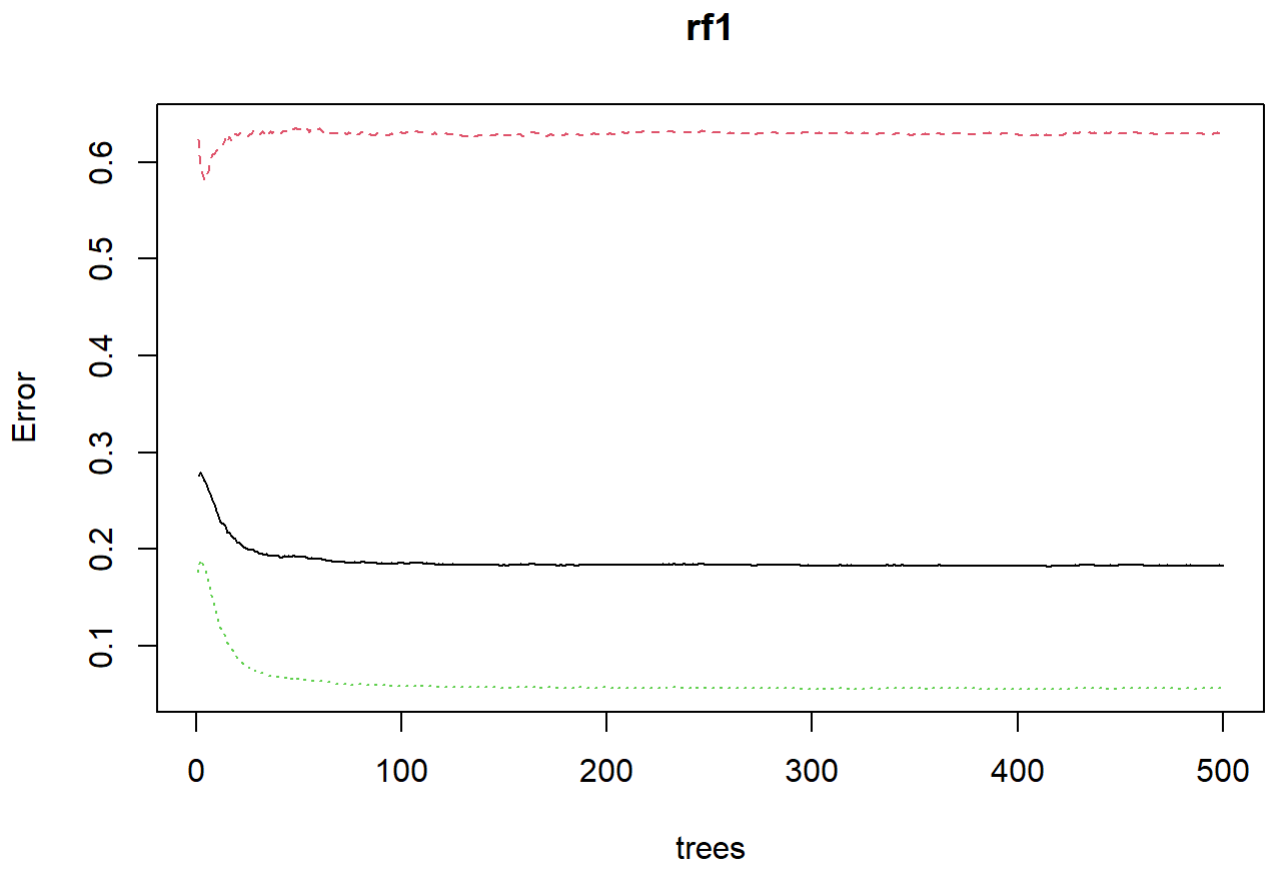
RandomForest did not see any improvement after removing outliers.

```
set.seed(7406)
rf1 <- randomForest(as.factor(default) ~., data = data_train, ntree= 500, replace = TRUE, mtry =
4)
print(rf1)
```

```
##
## Call:
##  randomForest(formula = as.factor(default) ~ ., data = data_train,      ntree = 500, replace
= TRUE, mtry = 4)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 18.31%
## Confusion matrix:
##      DEF    ND class.error
## DEF 2084  3530   0.6287852
## ND  1114 18630   0.0564222
```

After 100 trees there is no improvement in the accuracy.

```
plot(rf1)
```



Predicting values using type as “response” we can see that model’s accuracy is 82% but sensitivity is only 68% and balance accuracy is 76%. We need to improve the sensitivity as

misclassifying Defaulter might be costlier than misclassifying the non-defaulter.

```
ptr <- predict(rf1, data_test[,1:23])
cfr <- confusionMatrix(as.factor(data_test[,24]),as.factor(ptr))
cfr
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  DEF   ND
##        DEF  389  614
##        ND   183 3289
##
##                Accuracy : 0.8219
##                  95% CI : (0.8104, 0.833)
##     No Information Rate : 0.8722
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.3956
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.68007
##             Specificity : 0.84269
##          Pos Pred Value : 0.38784
##          Neg Pred Value : 0.94729
##              Prevalence : 0.12782
##          Detection Rate : 0.08693
##    Detection Prevalence : 0.22413
##       Balanced Accuracy : 0.76138
##
##        'Positive' Class : DEF
##
```

Using probability as type to change the threshold value to improve sensitivity of the model

```
pt <- predict(rf1, data_test[,1:23], type= "prob",index =2 )
```

CF with threshold value of 0.65. As we can see that this improves the model's sensitivity and balanced accuracy. However, it reduces the overall accuracy of the model.

```
tr_0.65 <- ifelse(pt[,1]>0.65,"DEF","ND")
table(tr_0.65)
```

```
## tr_0.65
##  DEF   ND
##  283 4192
```

```
cf1 <- confusionMatrix(as.factor(data_test[,24]),as.factor(tr_0.65))
cf1
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  DEF   ND
##        DEF  219  784
##        ND    64 3408
##
##                Accuracy : 0.8105
##                  95% CI : (0.7987, 0.8219)
##     No Information Rate : 0.9368
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2684
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.77385
##             Specificity : 0.81298
##          Pos Pred Value : 0.21834
##          Neg Pred Value : 0.98157
##              Prevalence : 0.06324
##          Detection Rate : 0.04894
##    Detection Prevalence : 0.22413
##       Balanced Accuracy : 0.79341
##
##        'Positive' Class : DEF
##
```

CF with threshold value of 0.75

```
tr_0.75 <- ifelse(pt[,1]>0.75, "DEF","ND")
cf2 <- confusionMatrix(as.factor(data_test[,24]),as.factor(tr_0.75))
cf2
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  DEF   ND
##        DEF  109  894
##        ND    27 3445
##
##                Accuracy : 0.7942
##                  95% CI : (0.782, 0.806)
##     No Information Rate : 0.9696
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1457
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.80147
##             Specificity : 0.79396
##          Pos Pred Value : 0.10867
##          Neg Pred Value : 0.99222
##              Prevalence : 0.03039
##          Detection Rate : 0.02436
##    Detection Prevalence : 0.22413
##       Balanced Accuracy : 0.79772
##
##        'Positive' Class : DEF
##
```

CF with threshold value of 0.90. It gives the maximum Balanced accuracy.

```
tr_0.90 <- ifelse(pt[,1]>0.90,"DEF","ND")
cf3 <- confusionMatrix(as.factor(data_test[,24]),as.factor(tr_0.90))
cf3
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  DEF   ND
##        DEF   25  978
##        ND     2 3470
##
##                Accuracy : 0.781
##                  95% CI : (0.7686, 0.793)
##     No Information Rate : 0.994
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0372
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.925926
##             Specificity : 0.780126
##          Pos Pred Value : 0.024925
##          Neg Pred Value : 0.999424
##              Prevalence : 0.006034
##          Detection Rate : 0.005587
##    Detection Prevalence : 0.224134
##       Balanced Accuracy : 0.853026
##
##        'Positive' Class : DEF
##
```

We can also build the ROC plot and compare the sensitivity and specificity for different threshold values.

```
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'
```

```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
par(pyt ="s")
```

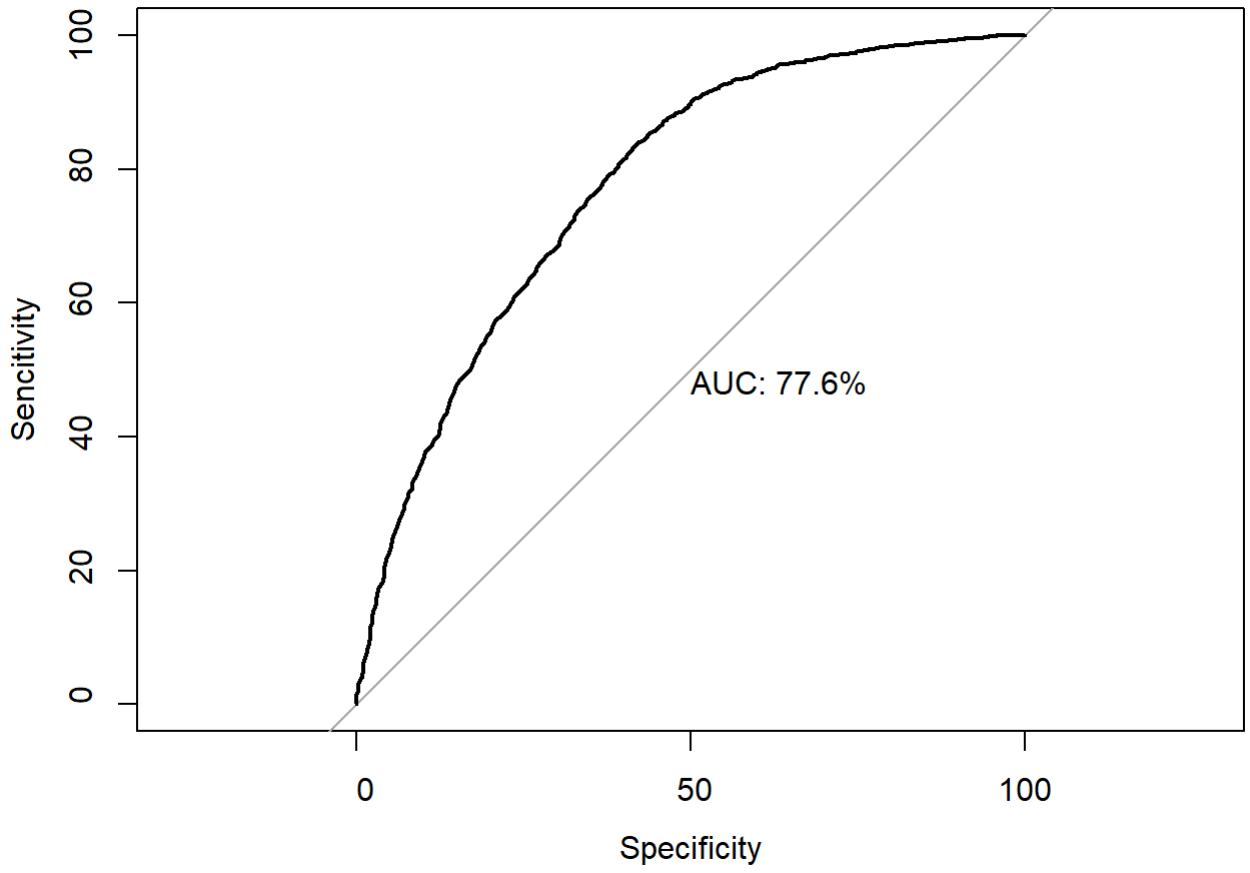
```
## Warning in par(pyt = "s"): "pyt" is not a graphical parameter
```

```
roc(data_test[,24], pt[,1], plot = TRUE, legacy.axes = T, percent = TRUE,
    print.auc =TRUE,

    #auc.polygon = TRUE,
    xlab= "Specificity",
    ylab= "Sencitivity"
    #xlab ="False Positive Percentage",
    #ylab =" True positive Percentage"
    )
```

```
## Setting levels: control = DEF, case = ND
```

```
## Setting direction: controls > cases
```




```
##
## Call:
## roc.default(response = data_test[, 24], predictor = pt[, 1],     percent = TRUE, plot = TRUE,
legacy.axes = T, print.auc = TRUE,     xlab = "Specificity", ylab = "Sencitivity")
##
## Data: pt[, 1] in 1003 controls (data_test[, 24] DEF) > 3472 cases (data_test[, 24] ND).
## Area under the curve: 77.62%
```

```
roc.info <- roc(data_test[,24], pt[,1], plot = FALSE, legacy.axes = TRUE)
```

```
## Setting levels: control = DEF, case = ND
```

```
## Setting direction: controls > cases
```

```
auc(roc.info)
```

```
## Area under the curve: 0.7762
```

```
roc.df <- data.frame(sensitivity = roc.info$sensitivities*100,
                     specificity =(roc.info$specificities)*100,
                     thresholds = roc.info$thresholds)
```

```
roc.df$Balance <- ((roc.df$sensitivity + roc.df$specificity)/2)
head(roc.df)
```

```
##   sensitivity specificity thresholds  Balance
## 1    100.0000   0.0000000        Inf 50.00000
## 2    100.0000   0.0997009      0.997 50.04985
## 3    100.0000   0.4985045      0.994 50.24925
## 4     99.9712   0.4985045      0.989 50.23485
## 5     99.9712   0.5982054      0.984 50.28470
## 6     99.9712   0.6979063      0.976 50.33455
```

Printing the top 10 records with the highest Balance accuracy.

```
df <- roc.df[with(roc.df,order(-Balance)),]
head(df)
```

```
##     sensitivity specificity thresholds  Balance
## 300    83.92857    57.92622      0.289 70.92740
## 301    83.72696    58.12562      0.287 70.92629
## 304    83.20853    58.62413      0.281 70.91633
## 303    83.35253    58.42473      0.283 70.88863
## 311    80.90438    60.81755      0.267 70.86096
## 302    83.49654    58.22532      0.285 70.86093
```