

Naive Bayes with RFE

For this Naive Bayes model, I used RFE as variables reduction method, which selected only four variables out of 24 variables. The AUC is 73.6. From the ROC plot we can conclude that the max Balanced Accuracy is 70.8 with sensitivity of 87.18318 and specificity of 54.43669.

```
library(caret)
library(tidyr)
library(MASS)
library(e1071)
library(pROC)
```

Reading the data

```
data1 <- read.table(file = "C://Users/cs_mo/Downloads/ISYE7406/ProjectCreditCard/creditcards.csv", 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....

```
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
```

```
data_train[,3] <- as.numeric(data_train[,3])
data_train[,4] <- as.numeric(data_train[,4])
```

```
x_train <- (data_train[,1:23])
y_train <- data_train[,24]
```

```
data_test[,3] <- as.numeric(data_test[,3])
data_test[,4] <- as.numeric(data_test[,4])
x_test <- data_test[,1:23]
y_test <- data_test[,24]
```

RFE for variable selection. The results shows the top 4 variables that gives the highest accuracy. We can use these variables and built Naive Bayes model.

```
set.seed(7406)

control <- rfeControl(functions = nbFuncs,
                      method = "cv",

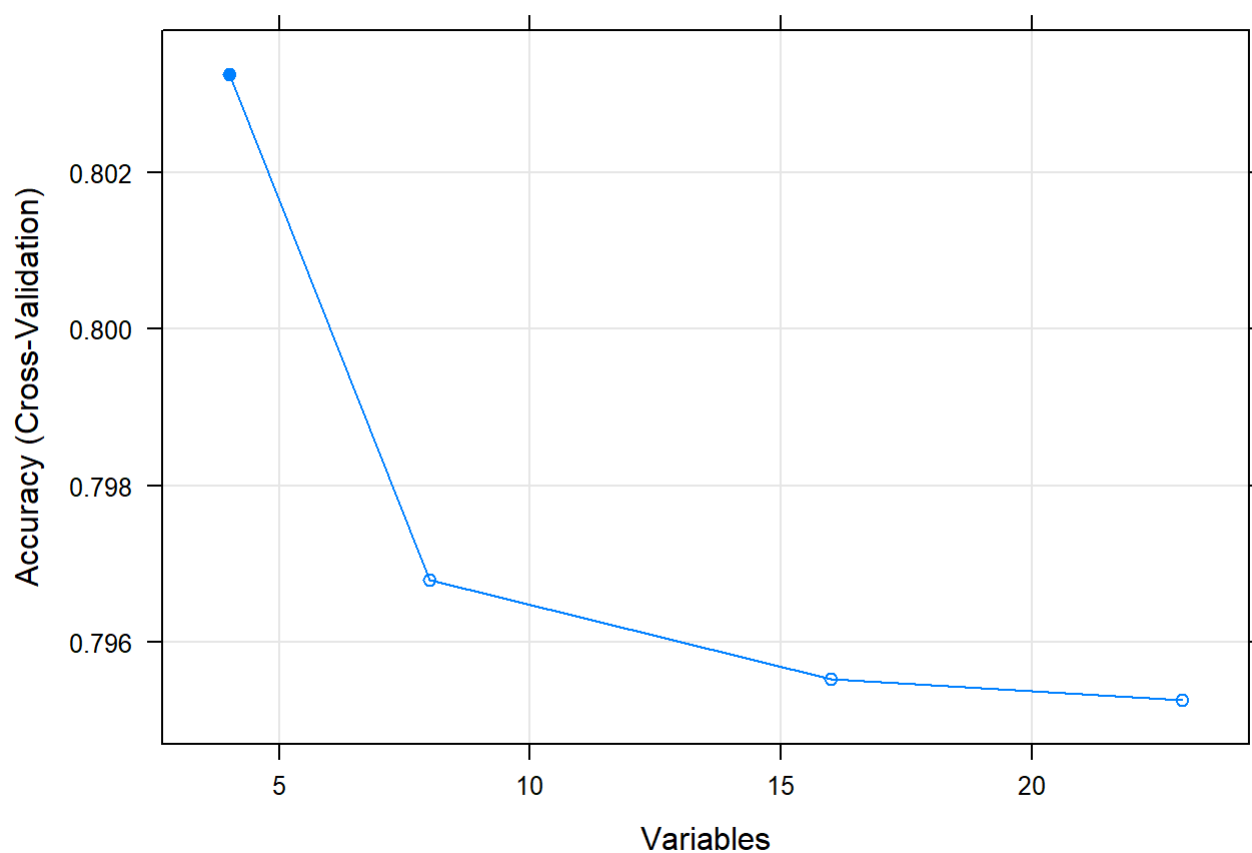
                      number = 5)
```

```
rfemodel <- rfe(x_train,
               as.factor(y_train),
               szes = c(1:23),
               rfeControl=control)
```

```
print(rfemodel)
```

```
##
## Recursive feature selection
##
## Outer resampling method: Cross-Validated (5 fold)
##
## Resampling performance over subset size:
##
## Variables Accuracy Kappa AccuracySD KappaSD Selected
##      4  0.8033 0.2326  0.003256 0.02121      *
##      8  0.7968 0.1928  0.004061 0.02901
##     16  0.7955 0.1873  0.004839 0.03993
##     23  0.7953 0.1872  0.005872 0.04211
##
## The top 4 variables (out of 4):
##    PAY_0, PAY_2, PAY_3, LIMIT_BAL
```

```
plot(rfemodel, type=c("g", "o"))
```



```
red_df <- x_train[,6:8]
red_df$LIMIT_BAL <- x_train[,1]
head(red_df)
```

```
##    PAY_0 PAY_2 PAY_3 LIMIT_BAL
## 1     2     2   -1    20000
## 2    -1     2    0   120000
## 3     0     0    0    90000
## 4     0     0    0    50000
## 5    -1     0   -1    50000
## 7     0     0    0   500000
```

```
red_model <- naiveBayes(red_df,as.factor(y_train),laplace =1)
red_pred <- predict(red_model, x_test)
red_cf <- confusionMatrix(red_pred,as.factor(y_test))
red_cf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  DEF   ND
##           DEF  446  273
##           ND   557 3199
##
##           Accuracy : 0.8145
##           95% CI : (0.8028, 0.8258)
##           No Information Rate : 0.7759
##           P-Value [Acc > NIR] : 1.324e-10
##
##           Kappa : 0.407
##
## Mcnemar's Test P-Value : < 2.2e-16
##
##           Sensitivity : 0.44467
##           Specificity : 0.92137
##           Pos Pred Value : 0.62031
##           Neg Pred Value : 0.85170
##           Prevalence : 0.22413
##           Detection Rate : 0.09966
##           Detection Prevalence : 0.16067
##           Balanced Accuracy : 0.68302
##
##           'Positive' Class : DEF
##
```

```
red_pred1 <- predict(red_model,x_test, type= "raw",index =2 )
head(red_pred1)
```

```
##           DEF           ND
## [1,] 0.11101405 0.88898595
## [2,] 0.92469628 0.07530372
## [3,] 0.08036284 0.91963716
## [4,] 0.10392840 0.89607160
## [5,] 0.11101405 0.88898595
## [6,] 0.55672161 0.44327839
```

CF with threshold value of 0.90 As we can see that this improves the model's sensitivity and balanced accuracy. However, it reduces the overall accuracy of the model.

```
tr_0.90 <- ifelse(red_pred1[,1]>0.90,"DEF","ND")
table(tr_0.90)
```

```
## tr_0.90
## DEF ND
## 441 4034
```

```
cf1 <- confusionMatrix(as.factor(y_test),as.factor(tr_0.90))
cf1
```

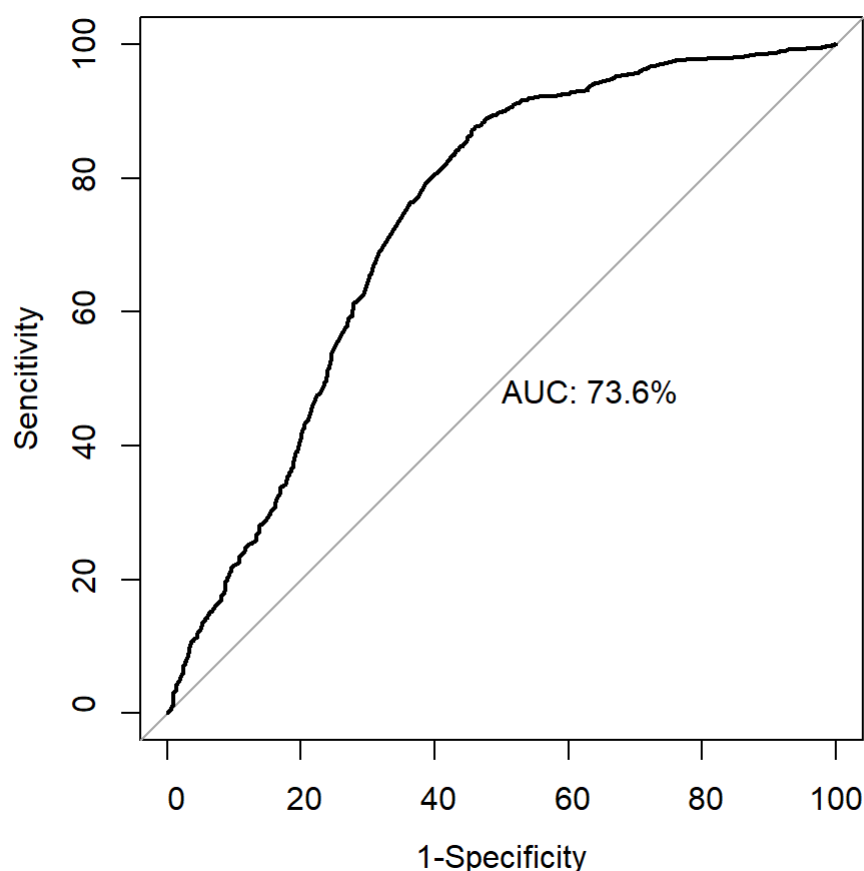
```
## Confusion Matrix and Statistics
```

```
##
##           Reference
## Prediction DEF ND
##           DEF 296 707
##           ND 145 3327
##
##           Accuracy : 0.8096
##           95% CI : (0.7978, 0.821)
## No Information Rate : 0.9015
## P-Value [Acc > NIR] : 1
##
##           Kappa : 0.3164
##
## McNemar's Test P-Value : <2e-16
##
##           Sensitivity : 0.67120
##           Specificity : 0.82474
##           Pos Pred Value : 0.29511
##           Neg Pred Value : 0.95824
##           Prevalence : 0.09855
##           Detection Rate : 0.06615
##           Detection Prevalence : 0.22413
##           Balanced Accuracy : 0.74797
##
##           'Positive' Class : DEF
##
```

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

```
par(pty = "s")
roc(y_test, red_pred1[,1], plot = TRUE, legacy.axes = T, percent = TRUE,
    print.auc = TRUE,

    #auc.polygon = TRUE,
    xlab= "1-Specificity",
    ylab= "Sensitivity"
    #xlab = "False Positive Percentage",
    #ylab = " True positive Percentage"
    )
```



```
##
## Call:
## roc.default(response = y_test, predictor = red_pred1[, 1], percent = TRUE, plot = TRUE, legacy.axes = T, print.auc = TRUE, xlab = "1-Specificity", ylab = "Sensitivity")
##
## Data: red_pred1[, 1] in 1003 controls (y_test DEF) > 3472 cases (y_test ND).
## Area under the curve: 73.55%
```

```
roc.info1 <- roc(y_test, red_pred1[,1], plot = FALSE, legacy.axes = TRUE)
auc(roc.info1)
```

```
## Area under the curve: 0.7355
```

```
roc.df1 <- data.frame(sensitivity = roc.info1$sensitivities*100,
                     specificity =(roc.info1$specificities)*100,
                     thresholds = roc.info1$thresholds)
```

```
 #(roc.df1)
```

```
roc.df1$Balance <- ((roc.df1$sensitivity + roc.df1$specificity)/2)
head(roc.df1)
```



```
##      sensitivity specificity thresholds  Balance
## 1    100.00000    0.0000000          Inf 50.00000
## 2     99.97120    0.0997009            1 50.03545
## 3     99.97120    0.1994018            1 50.08530
## 4     99.94240    0.1994018            1 50.07090
## 5     99.94240    0.2991027            1 50.12075
## 6     99.91359    0.2991027            1 50.10635
```

Printing the top 10 records with the highest Balance accuracy.

```
df1 <- roc.df1[with(roc.df1,order(-Balance)),]
head(df1)
```

```
##      sensitivity specificity thresholds  Balance
## 366     87.18318     54.43669   0.2040761 70.80993
## 367     87.15438     54.43669   0.2023805 70.79553
## 365     87.21198     54.33699   0.2058032 70.77449
## 359     87.50000     54.03789   0.2239782 70.76894
## 358     87.67281     53.83848   0.2258698 70.75565
## 360     87.47120     54.03789   0.2224196 70.75454
```