MINI PROJECT – RANDOM FOREST

Submitted By

Pratheeba R

Contents

1. Project Objective

This case is about a bank (Thera Bank) which has a growing customer base. Majority of these customers are liability customers (depositors) with varying size of deposits. The number of customers who are also borrowers (asset customers) is quite small, and the bank is interested in expanding this base rapidly to bring in more loan business and in the process earn more through the interest on loans. In particular, the management wants to explore ways of converting its liability customers to personal loan customers (while retaining them as depositors). A campaign that the bank ran last year for liability customers showed a healthy conversion rate of over 9% success. This has encouraged the retail marketing department to devise campaigns with better target marketing to increase the success ratio with minimal budget. The department wants to build a model that will help them identify the potential customers who have higher probability of purchasing the loan. This will increase the success ratio while at the same time reduce the cost of the campaign. The file Bank.xls contains data on 5000 customers. The data include customer demographic information (age, income, etc.), the customer's relationship with the bank (mortgage, securities account, etc.), and the customer response to the last personal loan campaign (Personal Loan). Among these 5000 customers, only 480 (= 9.6%) accepted the personal loan that was offered to them in the earlier campaign.

- Understanding the attributes - Find relationship between different attributes (Independent variables) and choose carefully which all attributes have to be a part of the analysis and why
- Exploratory Data Analysis
- Splitting data in Train and Test dataset
- Some Charts and Graphs to show case the relationship between Independent and Dependent Variables
- Model Development (Any one of the below techniques to be used)
  - Random Forest
  - CART
- Model Performance Measures
- Validation of Model
- Model Performance on Hold Out Sample

2. Exploratory Data Analysis

3.1 Environment Setup and Data import

3.1.1 Install necessary packages

The package needed to be installed is

- library(car)
- library(carData)

- library(lattice)
- library(ggplot2)
- library(caret)
- library(rpart)
- library(DataExplorer)
- library(ggplot2)
- library(ppcor)
- library(nFactors)
- library(psych)
- library(dplyr)
- library(tidyverse)
- library(purrr)
- library(grid)
- library(REdaS)
- library(foreign)
- library(PerformanceAnalytics)

3.1.2 Setup working directory

Set the working directory to the location where you have the dataset and code files using setwd() function. When a working directory is set, you don't have to mention the whole path of the file while importing a dataset which minimizes the time and probability of unwanted errors.

3.1.3 Import and read the dataset

The dataset under study is an xls file and so we can use read.xls() function to read the dataset and frame it as a data frame for our study.

3.2 Variable Identification

To understand the structure of the dataset, the following functions are being used,

FUNCTION	PURPOSE
dim(dataframe)	Number of columns and rows
str(dataframe)	Examine each column separately (the data type of each column and their sample values)
introduce(dataframe)	Displays number of rows, columns, discrete columns, continuous columns, missing columns, total missing values, complete rows, total observations and memory usage
summary(dataframe)	Data summary
mean(datacolumn)	Sample mean of the column
sd(datacolumn)	Standard deviation of column
table(datacolumn)	Frequency of datapoints for a particular column in a dataframe

anyNA(dataframe/attribute)	Returns a Boolean value indicating the presence.
plot(x)	Produces a scatterplot of the given attribute

3.2.1 Inferences

The given dataset contains 100 rows of 13 columns. All the values except Product ID are numeric values which represent the rating of the customer satisfaction.

```
 dim(mydata)
[1] 5000   14

 str(mydata)
'data.frame':  5000 obs. of  14 variables:
 $ ID                        : int  1 2 3 4 5 6 7 8 9 10 ...
 $ Age..in.years.            : int  25 45 39 35 35 37 53 50 35 34 ...
 $ Experience..in.years.: int  1 19 15 9 8 13 27 24 10 9 ...
 $ Income..in.K.month.   : int  49 34 11 100 45 29 72 22 81 180 ...
 $ ZIP.Code                  : int  91107 90089 94720 94112 91330 92121
91711 93943 90089 93023 ...
 $ Family.members            : int  4 3 1 1 4 4 2 1 3 1 ...
 $ CCAvg                     : num  1.6 1.5 1 2.7 1 0.4 1.5 0.3 0.6 8.9
...
 $ Education                 : int  1 1 1 2 2 2 2 3 2 3 ...
 $ Mortgage                  : int  0 0 0 0 0 155 0 0 104 0 ...
 $ Personal.Loan             : int  0 0 0 0 0 0 0 0 0 1 ...
 $ Securities.Account        : int  1 1 0 0 0 0 0 0 0 0 ...
 $ CD.Account                : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Online                    : int  0 0 0 0 0 1 1 0 1 0 ...
 $ CreditCard                : int  0 0 0 0 1 0 0 1 0 0 ...
```

Summary

```
summary(mydata)
       ID         Age..in.years.  Experience..in.years. Income..in.K.m
onth.
 Min.   :   1   Min.   :23.00   Min.   :-3.0           Min.   :  8.00
 1st Qu.:1251   1st Qu.:35.00   1st Qu.:10.0           1st Qu.: 39.00
 Median :2500   Median :45.00   Median :20.0           Median : 64.00
 Mean   :2500   Mean   :45.34   Mean   :20.1           Mean   : 73.77
 3rd Qu.:3750   3rd Qu.:55.00   3rd Qu.:30.0           3rd Qu.: 98.00
 Max.   :5000   Max.   :67.00   Max.   :43.0           Max.   :224.00

    ZIP.Code      Family.members      CCAvg           Education
 Min.   : 9307   Min.   :1.000   Min.   : 0.000   Min.   :1.000
 1st Qu.:91911   1st Qu.:1.000   1st Qu.: 0.700   1st Qu.:1.000
 Median :93437   Median :2.000   Median : 1.500   Median :2.000
 Mean   :93153   Mean   :2.397   Mean   : 1.938   Mean   :1.881
 3rd Qu.:94608   3rd Qu.:3.000   3rd Qu.: 2.500   3rd Qu.:3.000
 Max.   :96651   Max.   :4.000   Max.   :10.000   Max.   :3.000
                 NA's   :18
    Mortgage      Personal.Loan   Securities.Account   CD.Account
 Min.   :  0.0   Min.   :0.000   Min.   :0.0000     Min.   :0.0000
 1st Qu.:  0.0   1st Qu.:0.000   1st Qu.:0.0000     1st Qu.:0.0000
 Median :  0.0   Median :0.000   Median :0.0000     Median :0.0000
 Mean   : 56.5   Mean   :0.096   Mean   :0.1044     Mean   :0.0604
 3rd Qu.:101.0   3rd Qu.:0.000   3rd Qu.:0.0000     3rd Qu.:0.0000
 Max.   :635.0   Max.   :1.000   Max.   :1.0000     Max.   :1.0000

     Online         CreditCard
 Min.   :0.0000   Min.   :0.000
 1st Qu.:0.0000   1st Qu.:0.000
 Median :1.0000   Median :0.000
 Mean   :0.5968   Mean   :0.294
```

```
 3rd Qu.:1.0000   3rd Qu.:1.000  
 Max.   :1.0000   Max.   :1.000  
```

Missing Values:

```
                ID        Age..in.years. Experience..in.years. 
                 0                     0                     0 
Income..in.K.month.             ZIP.Code        Family.members 
                 0                     0                    18 
             CCAvg             Education              Mortgage 
                 0                     0                     0 
     Personal.Loan    Securities.Account            CD.Account 
                 0                     0                     0 
            Online            CreditCard 
                 0                     0 
```

It can be seen that there are missing values.

Missing Value Treatment:

newdata = na.omit(mydata)

newdata

dim(newdata)

mean(newdata$Family.members)

#Replace with 2(Round off value of mean)

mydata[is.na(mydata)] <- 2

colSums(is.na(mydata))

Replaced the missing values in family members with rounded off mean value

After which we check the dataset whether it contains any missing values,

```
                ID        Age..in.years. Experience..in.years. 
                 0                     0                     0 
Income..in.K.month.             ZIP.Code        Family.members 
                 0                     0                     0 
             CCAvg             Education              Mortgage 
                 0                     0                     0 
     Personal.Loan    Securities.Account            CD.Account 
                 0                     0                     0 
            Online            CreditCard 
                 0                     0 
```

3. Split data

Split the dataset into 70%-30% of training and test data with a seed of 123.

set.seed(123)

train1 = createDataPartition (mydata$Personal.Loan, p = .7, list = FALSE, times = 1)

head(train1)

trainingdata = mydata[train1,]

testData = mydata[-train1,]

dim(trainingdata)

dim(testData)

prop.table((table(trainingdata$Personal.Loan)))

prop.table((table(testData$Personal.Loan)))

```
dim(trainingdata)
[1] 3500   14
> dim(testData)
[1] 1500   14

It can be seen that the training data has 3500 rows of 14 columns and t
est data has 1500 rows of 14 columns.
```

The proportion of observations:

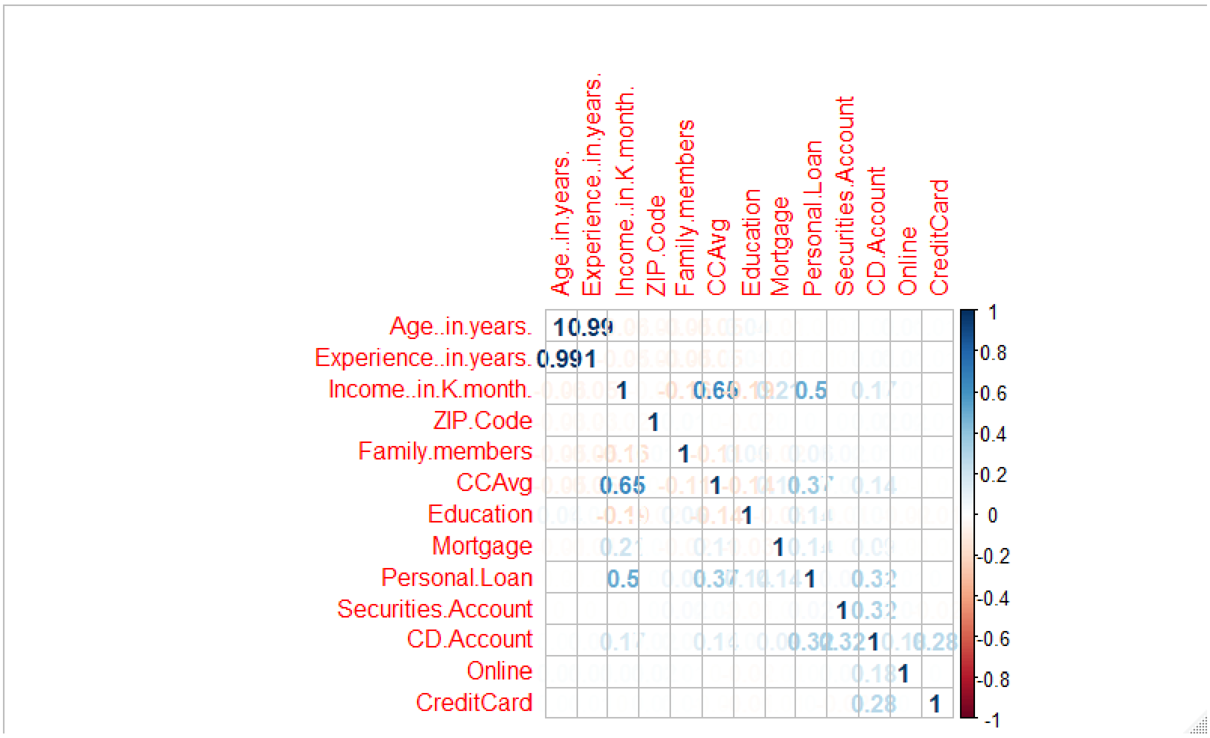
```
> prop.table((table(trainingdata$Personal.Loan)))

         0          1 
0.90542857 0.09457143 
> prop.table((table(testData$Personal.Loan)))

         0          1 
0.90066667 0.09933333 
     It can be seen that the proportions of who take the loan on adverti
sement or campaign is 9.4% in training data and 9.9% in test data and t
he proportion who doesn't take loan is 91% in training data and 90% in
test data.
```

4. Relationship between dependent and independent variables:

The correlation between the variables is shown here,

```
                  Age..in.years. Experience..in.years. Income..in.K.month. ZIP.Code
Age..in.years.              1.00                  0.99               -0.06    -0.0
Experience..in.years.       0.99                  1.00               -0.05    -0.0
Income..in.K.month.        -0.06                 -0.05                1.00    -0.0
ZIP.Code                   -0.03                 -0.03               -0.02     1.0
Family.members             -0.05                 -0.05               -0.16     0.0
CCAvg                      -0.05                 -0.05                0.65     0.0
Education                   0.04                  0.01               -0.19    -0.0
Mortgage                   -0.01                 -0.01                0.21     0.0
Personal.Loan              -0.01                 -0.01                0.50     0.0
Securities.Account          0.00                  0.00                0.00     0.0
CD.Account                  0.01                  0.01                0.17     0.0
Online                      0.01                  0.01                0.01     0.0
CreditCard                  0.01                  0.01                0.00     0.0
                      Family.members CCAvg Education Mortgage Personal.Loan Securities
Age..in.years.                 -0.05 -0.05      0.04    -0.01         -0.01
Experience..in.years.          -0.05 -0.05      0.01    -0.01         -0.01
```

```
Income..in.K.month.          -0.16  0.65    -0.19    0.21         0.50
ZIP.Code                      0.01  0.00    -0.02    0.01         0.00
Family.members                1.00 -0.11     0.06   -0.02         0.06
CCAvg                        -0.11  1.00    -0.14    0.11         0.37
Education                     0.06 -0.14     1.00   -0.03         0.14
Mortgage                     -0.02  0.11    -0.03    1.00         0.14
Personal.Loan                 0.06  0.37     0.14    0.14         1.00
Securities.Account            0.02  0.02    -0.01   -0.01         0.02
CD.Account                    0.01  0.14     0.01    0.09         0.32
Online                        0.01  0.00    -0.02   -0.01         0.01
CreditCard                    0.01 -0.01    -0.01   -0.01         0.00
                     CD.Account Online CreditCard
Age..in.years.             0.01   0.01       0.01
Experience..in.years.      0.01   0.01       0.01
Income..in.K.month.        0.17   0.01       0.00
ZIP.Code                   0.02   0.02       0.01
Family.members             0.01   0.01       0.01
CCAvg                      0.14   0.00      -0.01
Education                  0.01  -0.02      -0.01
Mortgage                   0.09  -0.01      -0.01
Personal.Loan              0.32   0.01       0.00
Securities.Account         0.32   0.01      -0.02
CD.Account                 1.00   0.18       0.28
Online                     0.18   1.00       0.00
CreditCard                 0.28   0.00       1.00
```



It can be seen from the plot that

- Age and Experience are correlated.
- CCAvg and Income are correlated.
- Personal Loan and income are correlated.
- Personal Loan and CCAvg are correlated.
- CD Account and Personal loan are correlated.

5. Random Forest Model:

```
Call:
 randomForest(formula = as.factor(trainingdata$Personal.Loan) ~      .,
data = trainingdata[, -1], ntree = 501, mtry = 3, nodesize = 50,      i
mportance = TRUE)
               Type of random forest: classification
                     Number of trees: 501
No. of variables tried at each split: 3

        OOB estimate of  error rate: 1.77%
Confusion matrix:
     0   1 class.error
0 3159  10  0.00315557
1   52 279  0.15709970
```

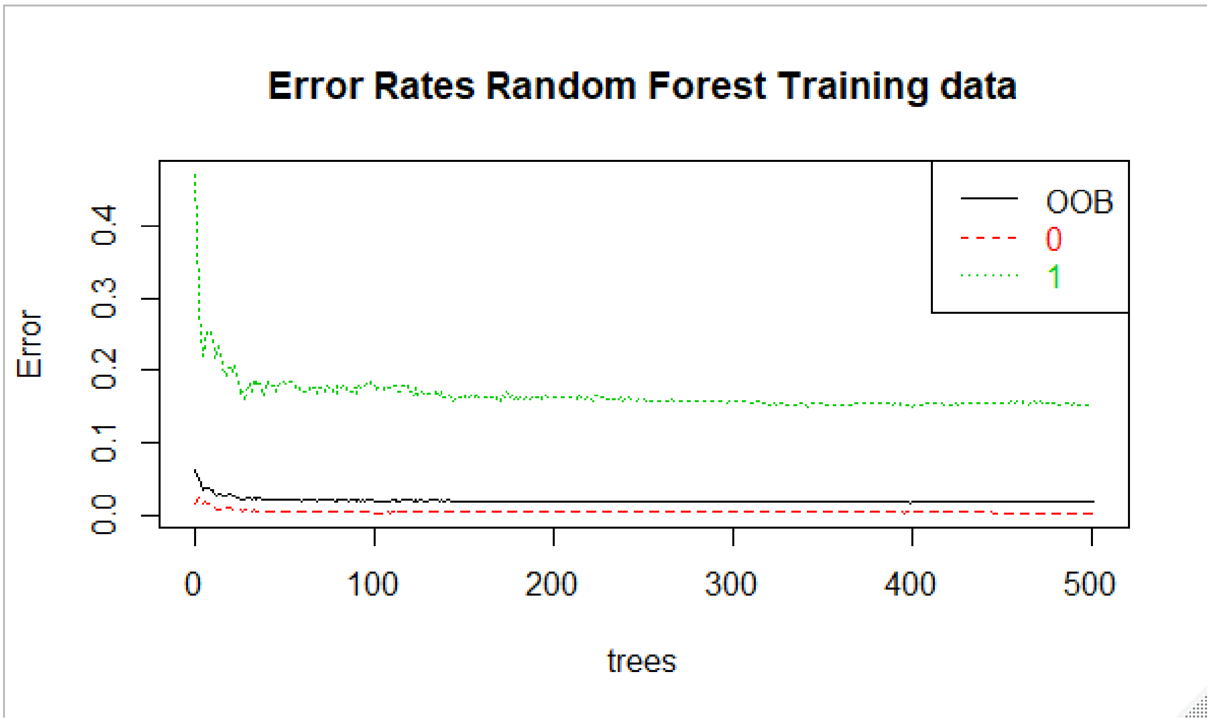
It is seen that the error rate for nodesize of 50 is 1.77% and the classification error rate for 0 is 0.3% and 1is 15%

Retrying with a smaller node size,(40)

```
Call:
 randomForest(formula = as.factor(trainingdata$Personal.Loan) ~      ., da
ta = trainingdata[, -1], ntree = 501, mtry = 3, nodesize = 40,      import
ance = TRUE)
               Type of random forest: classification
                     Number of trees: 501
No. of variables tried at each split: 3

        OOB estimate of  error rate: 1.8%
Confusion matrix:
     0   1 class.error
0 3157  12 0.003786683
1   51 280 0.154078550


Retrying with a larger node size,(55)

Call:
 randomForest(formula = as.factor(trainingdata$Personal.Loan) ~      ., da
ta = trainingdata[, -1], ntree = 501, mtry = 3, nodesize = 55,      import
ance = TRUE)
               Type of random forest: classification
                     Number of trees: 501
No. of variables tried at each split: 3

        OOB estimate of  error rate: 1.83%
Confusion matrix:
     0   1 class.error
0 3159  10  0.00315557
1   54 277  0.16314199
Call:
 randomForest(formula = as.factor(trainingdata$Personal.Loan) ~      ., da
ta = trainingdata[, -1], ntree = 501, mtry = 3, nodesize = 45,      import
ance = TRUE)
               Type of random forest: classification
                     Number of trees: 501
No. of variables tried at each split: 3

        OOB estimate of  error rate: 1.74%
Confusion matrix:
     0   1 class.error
```

```
0 3158   11 0.003471127
1   50 281 0.151057402
```

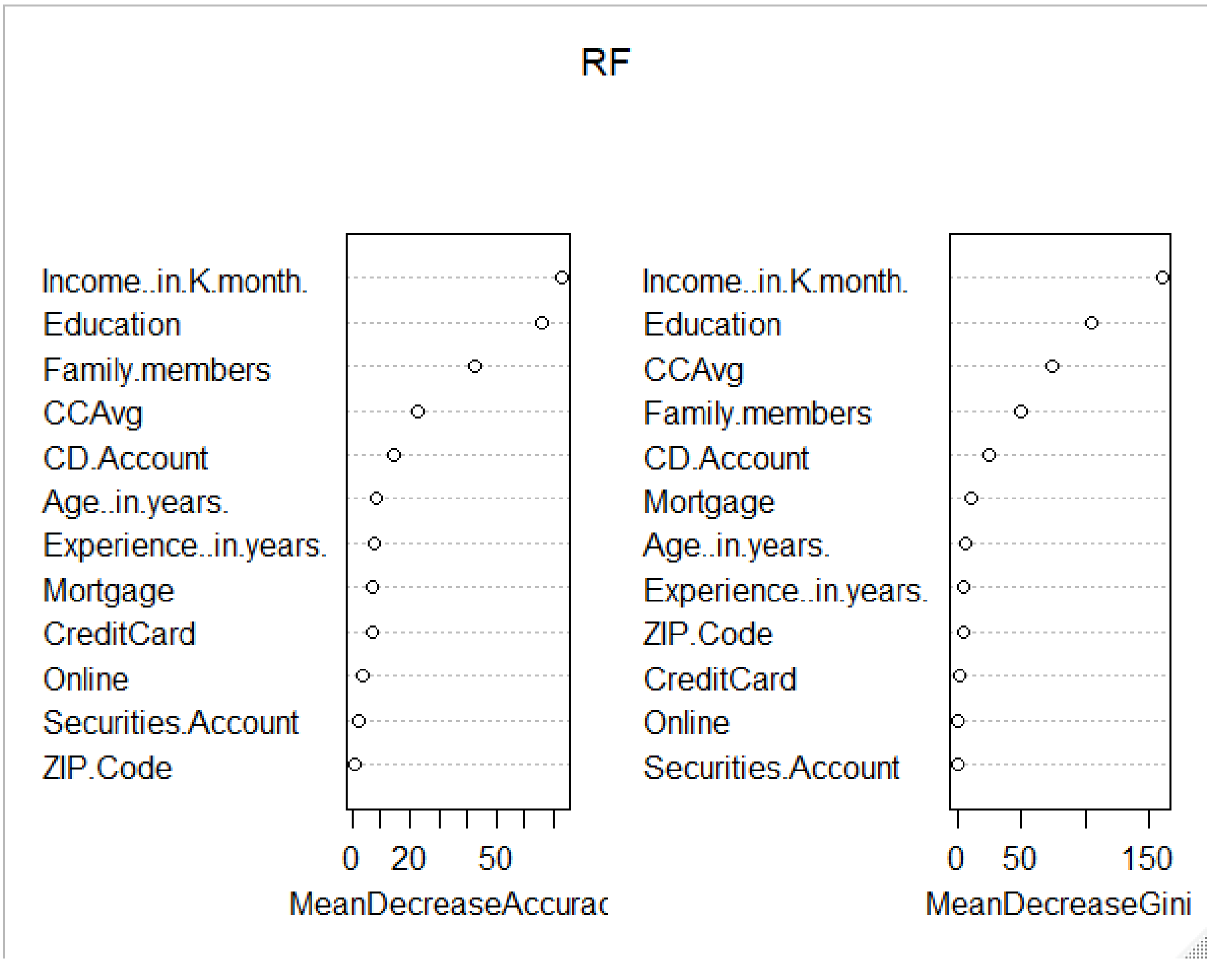
The error rate has been reduced, hence we take the model with node size = 45.



Important variables:

```
> impVar[order(impVar[,1], decreasing=TRUE),]
                           0     1 MeanDecreaseAccuracy MeanDecreaseGini
Income..in.K.month.    67.46 56.89                72.44           160.49
Education              66.17 46.74                65.51           104.86
Family.members         43.15 27.06                42.90            48.72
CCAvg                  19.41 17.88                22.51            73.71
CD.Account             11.60  9.92                14.46            25.26
Mortgage                8.82 -4.29                 6.81            10.11
Age..in.years.          7.88  1.84                 8.44             5.49
Experience..in.years.   7.56  0.26                 7.68             5.40
CreditCard              5.40  2.58                 6.80             1.66
Online                  3.45  0.11                 3.53             0.73
Securities.Account      1.42  1.80                 2.21             0.59
ZIP.Code               -0.03  1.62                 1.14             4.54
```

Variable Importance Plot:



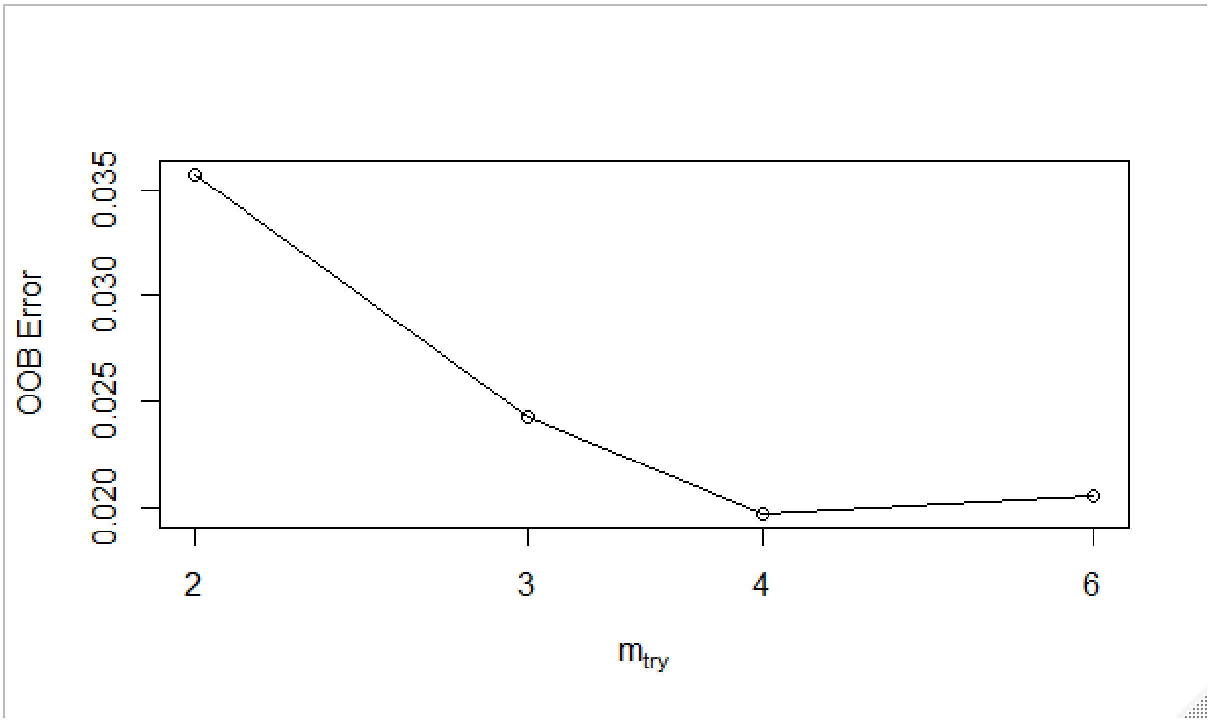
Building model with important variables,

```
Call:
 randomForest(formula = as.factor(trainingdata$Personal.Loan) ~      train
ingdata$Income..in.K.month. + trainingdata$Education +          trainingda
ta$Family.members + trainingdata$CCAvg + trainingdata$CD.Account +
trainingdata$Mortgage + trainingdata$Age..in.years. +         trainingdat
a$Experience..in.years., data = trainingdata[,      c(1, 10)], ntree = 501
, mtry = 3, nodesize = 45, importance = TRUE)
               Type of random forest: classification
                     Number of trees: 501
No. of variables tried at each split: 3

        OOB estimate of  error rate: 1.69%
```

Confusion matrix:

0	1	class.error
0	3156	00410224
1	46	13897281

It can be seen that the Out of Bag Error rate has been reduced while considering the important variables alone.



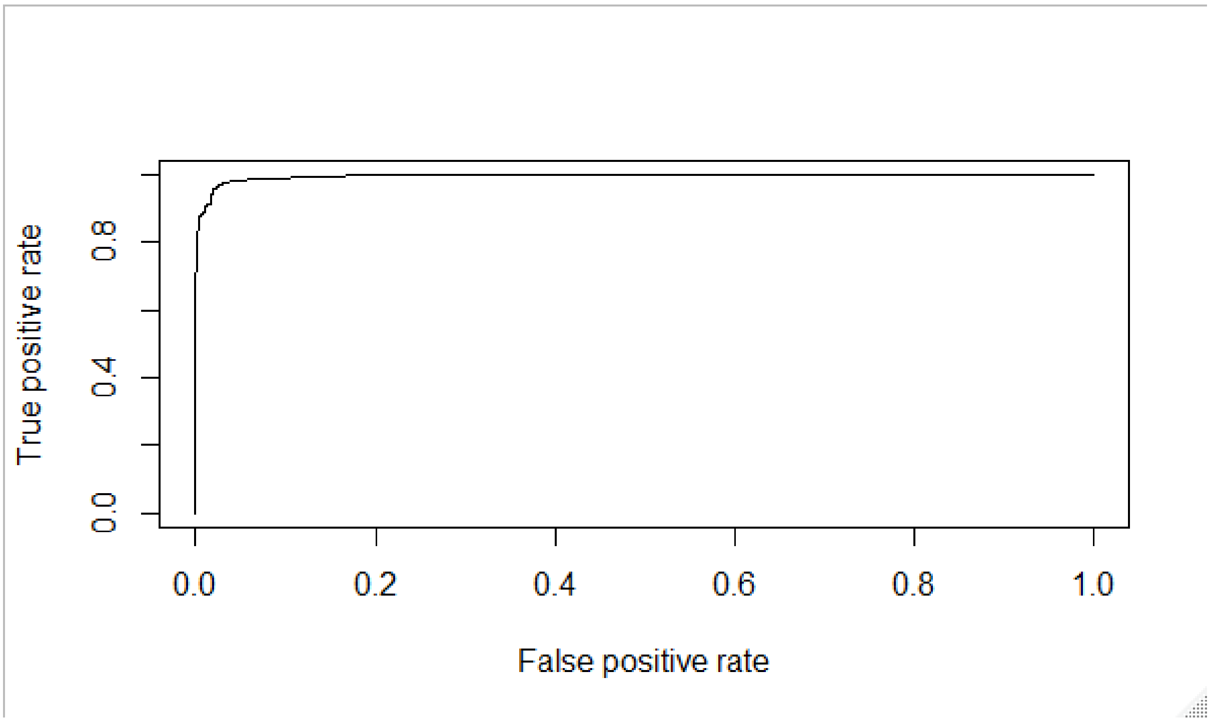
By tuning the random forest, we get OOB error as minimum at mtry=4

6. Model Performance Measures

	deciles	cnt	cnt_resp	cnt_non_resp	rrate	cum_resp	cum_non_resp	cum_rel_resp	cum_rel_non_resp	ks
1	14	3500	331	50	9.00%	331	50	25.0%	2.0%	0.23
2	12	3500	331	339	9.00%	662	389	50.0%	12.0%	0.38
3	11	3500	331	447	9.00%	993	836	75.0%	26.0%	0.49
4	9	3500	331	2333	9.00%	1324	3169	100.0%	100.0%	0.00

Hence it can be clearly seen that the bucket 9 has 2333 non responders to the loan campaign which should be clearly dealt with.

It can be seen that 9.5% will take personal loan after the campaign.



From the above graph, it can be seen that the model has a performance measure of 95%.

The area under the curve (AUC) seems to be 99.62%.(The plot of true positive rate (specificity) against the false positive rate(sensitivity).

Gini is 88.62% which indicates that the model is good.

The classification error is

	0	**1**
0	**3157**	**12**
1	**46**	**285**

It can be seen that 12 are predicted to be 1(will respond to campaign) instead of 0

And 46 that should be predicted as a respondent to the campaign is predicted as 0.

The success rate of the (respondent's) classification will be

```
> 285/(285+46)
[1] 0.8610272
```

The success rate of the non-respondents classification will be

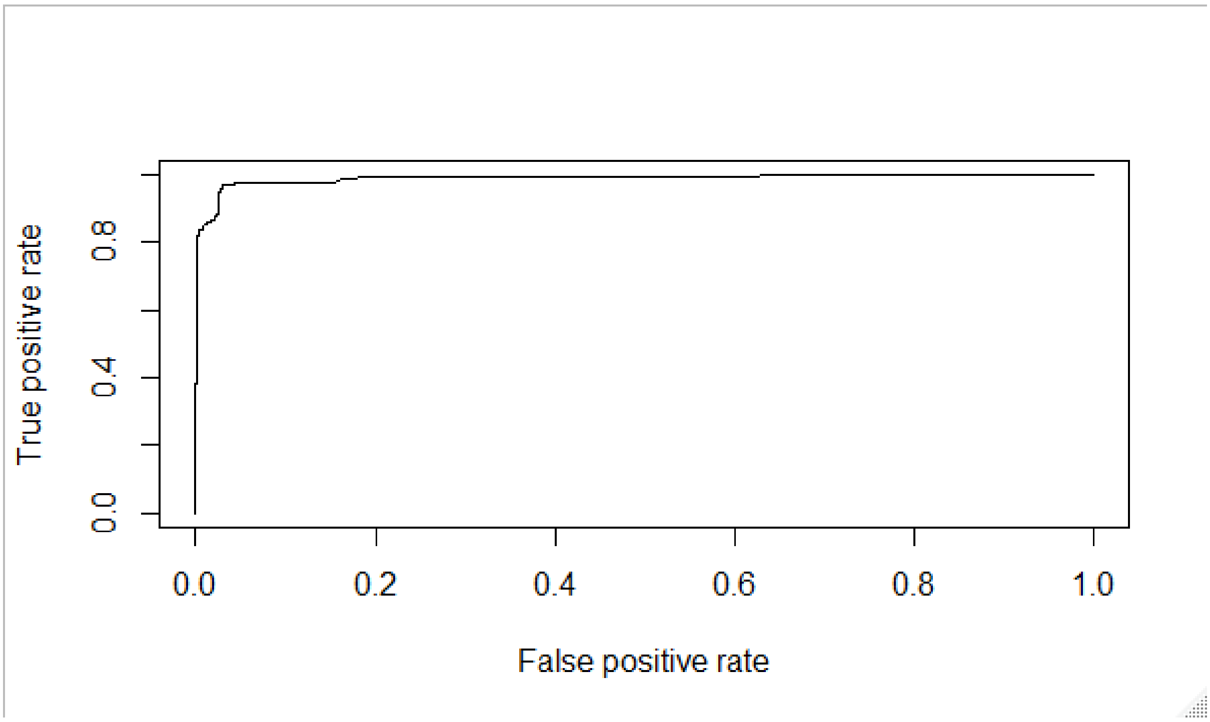
```
> 3157/(3157+12)
[1] 0.9962133
```

7. Test data

The bucket classification of test data is

	deciles	cnt	cnt_resp	cnt_non_resp	rrate	cum_resp	cum_non_resp	cum_rel_resp	cum_rel_non_resp	ks
1	14	1500	149	1351	10.0%	149	1351	25.0%	25.0%	0
2	12	1500	149	1351	10.0%	298	2702	50.0%	50.0%	0
3	11	1500	149	1351	10.0%	447	4053	75.0%	75.0%	0
4	9	1500	149	1351	10.0%	596	5404	100.0%	100.0%	0

It can be seen that 9.9% will take personal loan after the campaign.



From the above graph, it can be seen that the model has a performance measure of 94.3%.

The area under the curve (AUC) seems to be 98.97%.(The plot of true positive rate (specificity) against the false positive rate(sensitivity).

Gini is 87.87% which indicates that the model is good.

The classification error is

	0	**1**
0	**1346**	**5**
1	**25**	**124**

It can be seen that only 5 are predicted to be 1(will respond to campaign) instead of 0.

And 25 that should be predicted as a respondent to the campaign is predicted as 0.

The success rate of the (respondent's) classification will be

124/149 i.e 83.2%

The success rate of the non-respondents classification will be

1346/1351 i.e 99.62%

8. Appendix A – Source Code

```
setwd("C:/Users/HP/Desktop/Mini Project4")
getwd()
mydata = read.csv("Bank Personal Loan Dataset.csv", header = TRUE)
install.packages("PerformanceAnalytics")

library(car)
library(carData)
library(lattice)
library(ggplot2)
library(caret)
library(rpart)
library(DataExplorer)
library(ggplot2)
library(ppcor)
library(nFactors)
library(psych)
library(dplyr)
library(tidyverse)
library(purrr)
library(grid)
library(REdaS)
```

```
library(foreign)
library(PerformanceAnalytics)
attach(mydata)
names(mydata)
dim(mydata)
summary(mydata)
table(Education)
str(mydata)
colSums(is.na(mydata))
plot_missing(mydata)



describe(mydata)
hist(data = mydata)




#Missing value treatment
newdata = na.omit(mydata)
newdata
dim(newdata)
mean(newdata$Family.members)
#Replace with 2(Round off value of mean)
mydata[is.na(mydata)] <- 2
colSums(is.na(mydata))

#Split data
set.seed(123)
train1 = createDataPartition (mydata$Personal.Loan, p = .7, list = FALSE, times = 1)
head(train1)
```

```
trainingdata = mydata[train1,]
testData = mydata[-train1, ]
dim(trainingdata)
dim(testData)
prop.table((table(trainingdata$Personal.Loan)))
prop.table((table(testData$Personal.Loan)))
pairs(mydata)

install.packages("randomForest")

library(randomForest)



RF = randomForest(as.factor(trainingdata$Personal.Loan) ~ ., data = trainingdata[,-1],
          ntree = 501, mtry = 3, nodesize = 50,importance = TRUE )
print(RF)
plot(RF, main = "")

legend("topright", c("OOB", "0", "1"), text.col=1:6, lty=1:3, col=1:3)
title(main="Error Rates Random Forest Training data")
RF$err.rate

# List the importance of the variables.
impVar <- round(randomForest::importance(RF), 2)
# impVar[order(impVar[,3], decreasing=TRUE),]
# impVar[order(impVar[,4], decreasing=TRUE),]
impVar[order(impVar[,1], decreasing=TRUE),]

chart.Correlation(mydata, histogram = TRUE, pch = 19)
corr_mydata1 = cor(mydata)
```

```
library(corrplot)
round(corrplot(corr_mydata1, method = "number"), 2)

## Tuning Random Forest
tRF <- tuneRF(x = mydata[,-c(1,10)],
        y=as.factor(mydata$Personal.Loan),
        mtryStart = 2,
        ntreeTry=100,
        stepFactor = 1.5,
        improve = 0.0001,
        trace=TRUE,
        plot = TRUE,
        doBest = TRUE,
        nodesize = 100,
        importance=TRUE
)

tRF$importance

trainingdata$predict.class <- predict(tRF, trainingdata, type="class")
trainingdata$predict.class

trainingdata$predict.score <- predict(tRF, trainingdata, type="prob")
trainingdata$predict.score
head(trainingdata)
class(trainingdata$predict.score)




## deciling code
```

```
decile <- function(x){
  deciles <- vector(length=14)
  for (i in seq(0.1,1,.1)){
    deciles[i*14] <- quantile(x, i, na.rm=T)
  }
  return (
    ifelse(x<deciles[1], 1,
    ifelse(x<deciles[2], 2,
    ifelse(x<deciles[3], 3,
    ifelse(x<deciles[4], 4,
    ifelse(x<deciles[5], 5,
    ifelse(x<deciles[6], 6,
    ifelse(x<deciles[7], 7,
    ifelse(x<deciles[8], 8,
    ifelse(x<deciles[9], 9,
    ifelse(x<deciles[10], 10,
    ifelse(x<deciles[11], 11,
    ifelse(x<deciles[12], 12,
    ifelse(x<deciles[13], 13,14
    )))))))))))))
}

trainingdata$deciles <- decile(trainingdata$predict.score[,2])
trainingdata$deciles

library(data.table)
tmp_DT = data.table(trainingdata)
rank <- tmp_DT[, list(
  cnt = length(trainingdata$Personal.Loan),
  cnt_resp = sum(trainingdata$Personal.Loan),
```

```
  cnt_non_resp = sum(Personal.Loan == 0)) ,
  by=deciles][order(-deciles)]
rank$rrate <- round (rank$cnt_resp / rank$cnt,2);
rank$cum_resp <- cumsum(rank$cnt_resp)
rank$cum_non_resp <- cumsum(rank$cnt_non_resp)
rank$cum_rel_resp <- round(rank$cum_resp / sum(rank$cnt_resp),2);
rank$cum_rel_non_resp <- round(rank$cum_non_resp / sum(rank$cnt_non_resp),2);
rank$ks <- abs(rank$cum_rel_resp - rank$cum_rel_non_resp);


library(scales)
rank$rrate <- percent(rank$rrate)
rank$cum_rel_resp <- percent(rank$cum_rel_resp)
rank$cum_rel_non_resp <- percent(rank$cum_rel_non_resp)

View(rank)

sum(trainingdata$Personal.Loan) / nrow(trainingdata)


library(ROCR)
pred <- prediction(trainingdata$predict.score[,2], trainingdata$Personal.Loan)
perf <- performance(pred, "tpr", "fpr")
plot(perf)
KS <- max(attr(perf, 'y.values')[[1]]-attr(perf, 'x.values')[[1]])
KS

## Area Under Curve
auc <- performance(pred,"auc");
auc <- as.numeric(auc@y.values)
```

```
auc

## Gini Coefficient
library(ineq)
gini = ineq(trainingdata$predict.score[,2], type="Gini")
gini

## Classification Error
with(trainingdata, table(trainingdata$Personal.Loan, predict.class))



## Scoring syntax
testData$predict.class <- predict(tRF, testData, type="class")
testData$predict.score <- predict(tRF, testData, type="prob")

testData$deciles <- decile(testData$predict.score[,2])

tmp_DT = data.table(testData)
h_rank <- tmp_DT[, list(
  cnt = length(testData$Personal.Loan),
  cnt_resp = sum(testData$Personal.Loan),
  cnt_non_resp = sum(testData$Personal.Loan == 0)) ,
  by=deciles][order(-deciles)]
h_rank$rrate <- round (h_rank$cnt_resp / h_rank$cnt,2);
h_rank$cum_resp <- cumsum(h_rank$cnt_resp)
h_rank$cum_non_resp <- cumsum(h_rank$cnt_non_resp)
h_rank$cum_rel_resp <- round(h_rank$cum_resp / sum(h_rank$cnt_resp),2);
h_rank$cum_rel_non_resp <- round(h_rank$cum_non_resp / sum(h_rank$cnt_non_resp),2);
h_rank$ks <- abs(h_rank$cum_rel_resp - h_rank$cum_rel_non_resp);
```

```r
library(scales)
h_rank$rrate <- percent(h_rank$rrate)
h_rank$cum_rel_resp <- percent(h_rank$cum_rel_resp)
h_rank$cum_rel_non_resp <- percent(h_rank$cum_rel_non_resp)

View(h_rank)

sum(testData$Personal.Loan) / nrow(testData)

pred <- prediction(testData$predict.score[,2], testData$Personal.Loan)
perf <- performance(pred, "tpr", "fpr")
plot(perf)
KS <- max(attr(perf, 'y.values')[[1]]-attr(perf, 'x.values')[[1]])
KS

## Area Under Curve
auc <- performance(pred,"auc");
auc <- as.numeric(auc@y.values)
auc

## Gini Coefficient
library(ineq)
gini = ineq(testData$predict.score[,2], type="Gini")
gini

#Classification error on test data
with(testData, table(testData$Personal.Loan, testData$predict.class))
```