Mini Project – Bank_Personal_Loan

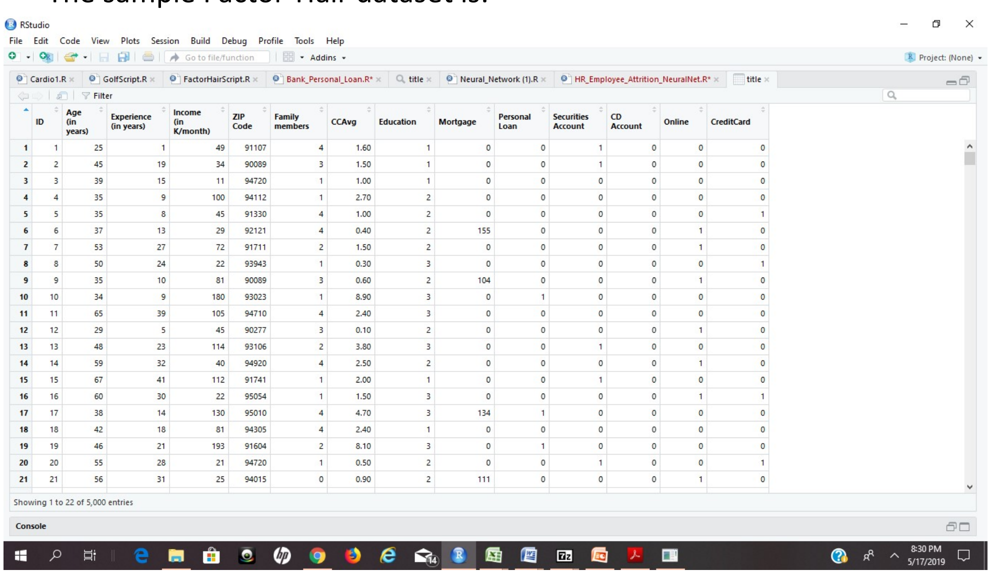
Table of Contents

1. Project Objective

The objective of the report is to explore the Bank_Personal_Loan_Modelling data set ("Bank_Personal_Loan_Modelling.xlsx") in R and prepare a Managerial Report by explaining the following points. This Managerial report consists of the following:

- Business Objective Statement
- Exploratory Data Analysis
- Hypothesis Statement
- Hypothesis Validation
- Splitting data in Train and Test dataset
- Feature Engineering / Derived Variable creation based on Hypothesis
- Some Charts and Graphs to show case the relationship between Independent and Dependent Variables
- Model Development (With the help of below techniques to be used)
    - CART
- Model Performance Measures
- Validation of Model
- Model Performance on Hold Out Sample
- Model Implementation / Deployment Strategy

The sample Factor-Hair dataset is:



	ID	Age (in years)	Experience (in years)	Income (in K/month)	ZIP Code	Family members	CCAvg	Education	Mortgage	Personal Loan	Securities Account	CD Account	Online	CreditCard
1	1	25	1	49	91107	4	1.60	1	0	0	1	0	0	0
2	2	45	19	34	90089	3	1.50	1	0	0	1	0	0	0
3	3	39	15	11	94720	1	1.00	1	0	0	0	0	0	0
4	4	35	9	100	94112	1	2.70	2	0	0	0	0	0	0
5	5	35	8	45	91330	4	1.00	2	0	0	0	0	0	1
6	6	37	13	29	92121	4	0.40	2	155	0	0	0	1	0
7	7	53	27	72	91711	2	1.50	2	0	0	0	0	1	0
8	8	50	24	22	93943	1	0.30	3	0	0	0	0	0	1
9	9	35	10	81	90089	3	0.60	2	104	0	0	0	1	0
10	10	34	9	180	93023	1	8.90	3	0	1	0	0	0	0
11	11	65	39	105	94710	4	2.40	3	0	0	0	0	0	0
12	12	29	5	45	90277	3	0.10	2	0	0	0	0	1	0
13	13	48	23	114	93106	2	3.80	3	0	0	1	0	0	0
14	14	59	32	40	94920	4	2.50	2	0	0	0	0	1	0
15	15	67	41	112	91741	1	2.00	1	0	0	1	0	0	0
16	16	60	30	22	95054	1	1.50	3	0	0	0	0	1	1
17	17	38	14	130	95010	4	4.70	3	134	1	0	0	0	0
18	18	42	18	81	94305	4	2.40	1	0	0	0	0	0	0
19	19	46	21	193	91604	2	8.10	3	0	1	0	0	0	0
20	20	55	28	21	94720	1	0.50	2	0	0	1	0	0	1
21	21	56	31	25	94015	0	0.90	2	111	0	0	0	1	0

Showing 1 to 22 of 5,000 entries

Console

Note: This is a sample data set of 22 rows. The actual provided data set has 5000 rows.

Business Objective: To build a model that helps bank to identify the potential customers who have higher probability of purchasing the loan.

2. Assumptions

Since, the data provided in the dataset is continuous and not appears on same scale, it is required to scale the data.

Please refer Appendix A for Source Code.

3. Step by step approach

We shall follow a step by step approach to arrive to the final conclusion as follows:

1. Environment set up and Data import
2. Identifying dependant and independent variables
3. Identify Correlation between independent and target variables
4. Divide the data into Train and Test Data sets.
5. Balance the Train Data Set
6. Build the CART Model
7. Predict Model performance using Train and Test data sets
8. Validate model
9. Conclusion

3.1. Environment Set up and Data Import

Please refer Appendix A for Source Code.

3.2. Variable Identification

In the given data set, first column is an ID column, which is not considered as a variable.

10^{th} Column "**Personal Loan**" is the dependant variable.

These are the Independent variables with their expansion in the given data set.

```
 $ Age (in years)        : num  25 45 39 35 35 37 53 50 35 34 ...
 $ Experience (in years): num  1 19 15 9 8 13 27 24 10 9 ...
 $ Income (in K/month)  : num  49 34 11 100 45 29 72 22 81 180 ...
 $ ZIP Code              : num  91107 90089 94720 94112 91330 ...
 $ Family members        : num  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             : num  1 1 1 2 2 2 2 3 2 3 ...
 $ Mortgage              : num  0 0 0 0 0 155 0 0 104 0 ...
 $ Personal Loan         : num  0 0 0 0 0 0 0 0 0 1 ...
 $ Securities Account    : num  1 1 0 0 0 0 0 0 0 0 ...
 $ CD Account            : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Online                : num  0 0 0 0 0 1 1 0 1 0 ...
 $ CreditCard            : num  0 0 0 0 1 0 0 1 0 0 ...
```

The Summary statistics of the data is shown below.

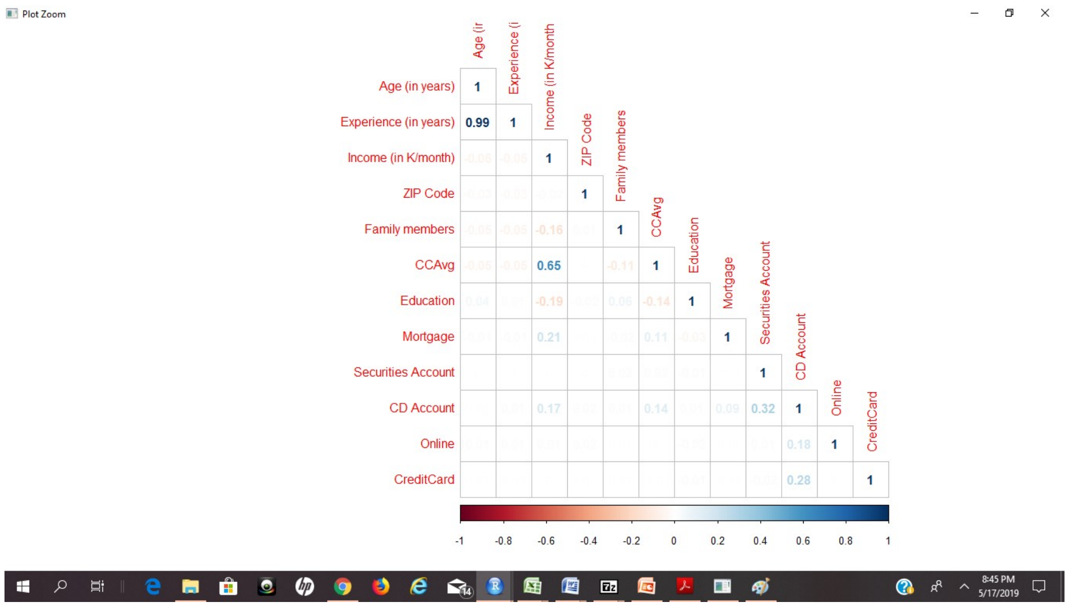
```
       ID       Age (in years)  Experience (in years) Income (in K/month)    ZIP Code    
 Min.   :   1   Min.   :23.00   Min.   :-3.0          Min.   :  8.00      Min.   : 9307  
 1st Qu.:1251   1st Qu.:35.00   1st Qu.:10.0          1st Qu.: 39.00      1st Qu.:91911  
 Median :2500   Median :45.00   Median :20.0          Median : 64.00      Median :93437  
 Mean   :2500   Mean   :45.34   Mean   :20.1          Mean   : 73.77      Mean   :93153  
 3rd Qu.:3750   3rd Qu.:55.00   3rd Qu.:30.0          3rd Qu.: 98.00      3rd Qu.:94608  
 Max.   :5000   Max.   :67.00   Max.   :43.0          Max.   :224.00      Max.   :96651  
 Family members      CCAvg          Education        Mortgage     Personal Loan   Securities Account
 Min.   :0.000   Min.   : 0.000   Min.   :1.000   Min.   :  0.0   Min.   :0.000   Min.   :0.0000    
 1st Qu.:1.000   1st Qu.: 0.700   1st Qu.:1.000   1st Qu.:  0.0   1st Qu.:0.000   1st Qu.:0.0000    
 Median :2.000   Median : 1.500   Median :2.000   Median :  0.0   Median :0.000   Median :0.0000    
 Mean   :2.389   Mean   : 1.938   Mean   :1.881   Mean   : 56.5   Mean   :0.096   Mean   :0.1044    
 3rd Qu.:3.000   3rd Qu.: 2.500   3rd Qu.:3.000   3rd Qu.:101.0   3rd Qu.:0.000   3rd Qu.:0.0000    
 Max.   :4.000   Max.   :10.000   Max.   :3.000   Max.   :635.0   Max.   :1.000   Max.   :1.0000    
   CD Account          Online         CreditCard   
 Min.   :0.0000   Min.   :0.0000   Min.   :0.000  
 1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.000  
 Median :0.0000   Median :1.0000   Median :0.000  
 Mean   :0.0604   Mean   :0.5968   Mean   :0.294  
 3rd Qu.:0.0000   3rd Qu.:1.0000   3rd Qu.:1.000  
 Max.   :1.0000   Max.   :1.0000   Max.   :1.000  
```

3.3. Identify Correlation between Independent variables

A Correlation matrix gives the correlation scores of each variable against each variable.Also, “corrplot” method of “corrplot” Package is used to obtain the correlation diagram with the correlation scores as shown below.

The highlighted values in the correlation diagram shows the dependency between target variable and independent variables. As per the correlation diagram , Experience is highly correlated with Age, hence, we remove the Experience variable from the data set.

Please refer Appendix A for Source Code.



3.4. Divide data into Train and Test Data Sets

Please refer Appendix A for Source Code.

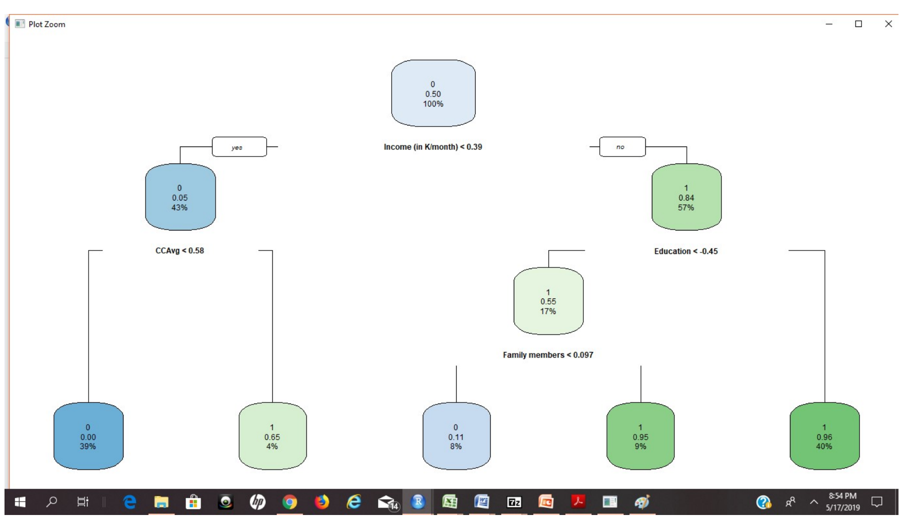
3.5. Balance the Train Data Set

Please refer Appendix A for Source Code.

3.6. Build CART Model

Please refer Appendix A for Source Code to build the CART model using Train Data set.

The CART model is



3.7. Predict Model performance using Train and Test Data Sets

The CP table for the CART model is shown below.

```
> DTModel$cptable
         CP nsplit rel error   xerror       xstd
1 0.7656250      0  1.000000 1.112500 0.03927753
2 0.0640625      1  0.234375 0.253125 0.02628503
3 0.0218750      3  0.106250 0.209375 0.02420331
4 0.0000000      4  0.084375 0.175000 0.02233883
```

The predicted model with Train and Test Data sets is shown below(with few rows)

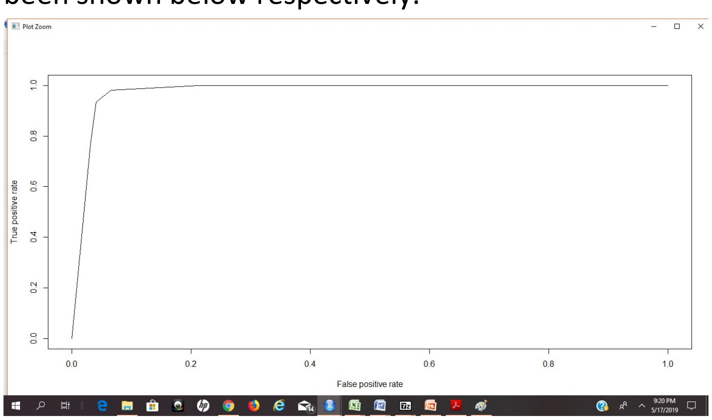
	0	1
3652	0.03921569	0.9607843
2062	0.03921569	0.9607843
2542	0.03921569	0.9607843
4147	1.00000000	0.0000000
1632	0.03921569	0.9607843
3129	1.00000000	0.0000000
1547	0.88679245	0.1132075
2197	0.03921569	0.9607843
3557	1.00000000	0.0000000
1667	0.03921569	0.9607843

	0	1
1	1.00000000	0.0000000
10	0.03921569	0.9607843
21	1.00000000	0.0000000
23	1.00000000	0.0000000
25	0.88679245	0.1132075
26	1.00000000	0.0000000
29	1.00000000	0.0000000
37	0.88679245	0.1132075
43	0.03921569	0.9607843
47	1.00000000	0.0000000

Please refer Appendix A for Source Code.

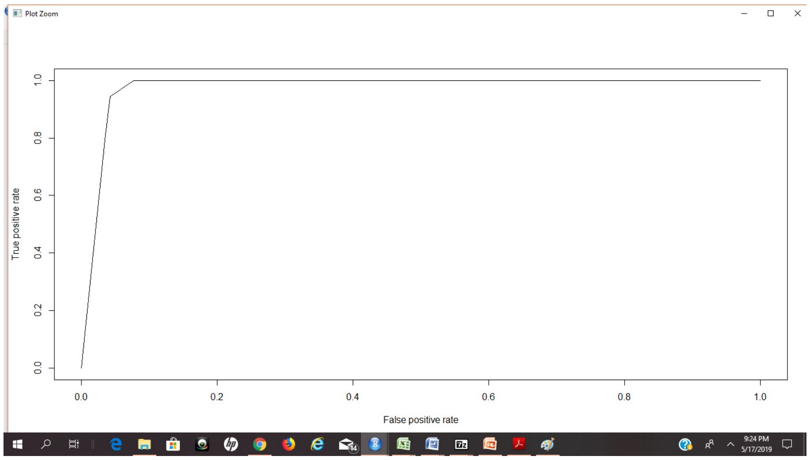
3.8. Validate Model

ROCR Confusion matrix is built using Train and Test Data sets and their plots have been shown below respectively.



The AUC calculated as per the above plot for Train Data set is: 0.976875

Which means, 97% times we can differentiate between True Positive Rate and False Positive Rate.



The AUC calculated as per the above plot for Test Data set is: 0.9773298
Which means, 97% times we can differentiate between True Positive Rate and False Positive Rate.

Both AUC values are almost equal which shows the model fits well.
Please refer Appendix A for Source Code.

KS values for Train and Test Data are shown below.

```
> #KS on Train
> KS = max(attr(perf1, 'y.values')[[1]]-attr(perf1, 'x.values')[[1]])
> KS
[1] 0.915625
> #KS on Test
> KS = max(attr(perf, 'y.values')[[1]]-attr(perf, 'x.values')[[1]])
> KS
[1] 0.9231343
```

As per the above values, the model seems to be over fitting.

Gini Values for Train and Test Data are shown below.

```
> #Gini For Train
> gini = ineq(predTrain[,2], type = "Gini")
> gini
[1] 0.476875
> #Gini for Test
> gini = ineq(predDT[,2], type="Gini")
> gini
[1] 0.8035083
```

Gini Value for Train data set seems to be a good fit, however, on test data set, it seems to be over fitting.

RandomForest model and Neural Network models are also built with the above Train and Test Data Sets and the model performance measures are shown below.

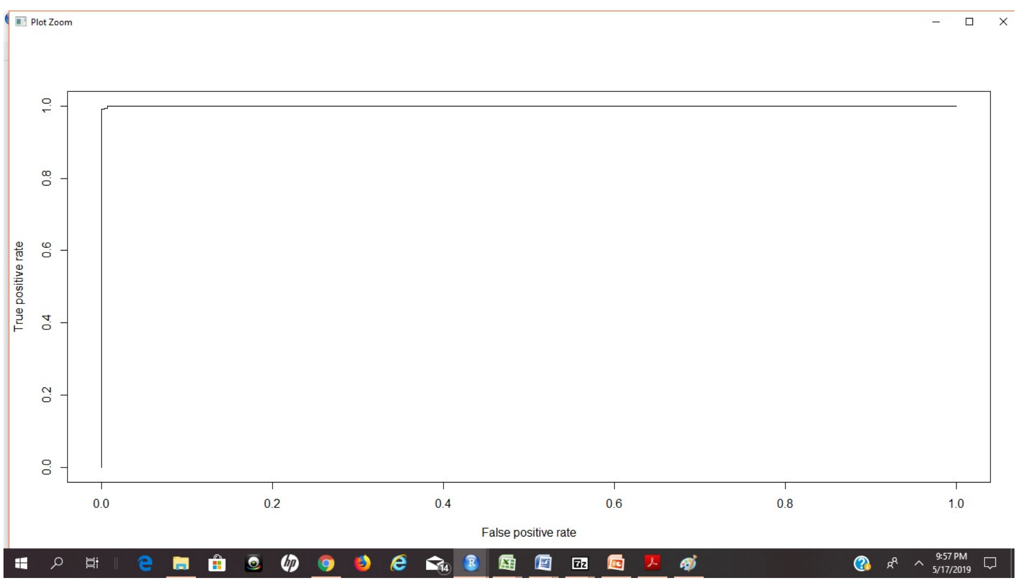
Random Forest model built with Train data set is:

```
> print(RFmodel)

Call:
 randomForest(formula = PersonalLoan ~ ., data = train.new[, -12],      mtry = 3, nodesize = 10, ntree = 1
00, importance = TRUE)
               Type of random forest: regression
                     Number of trees: 100
No. of variables tried at each split: 3

          Mean of squared residuals: 0.03552027
                    % Var explained: 85.79
>
```

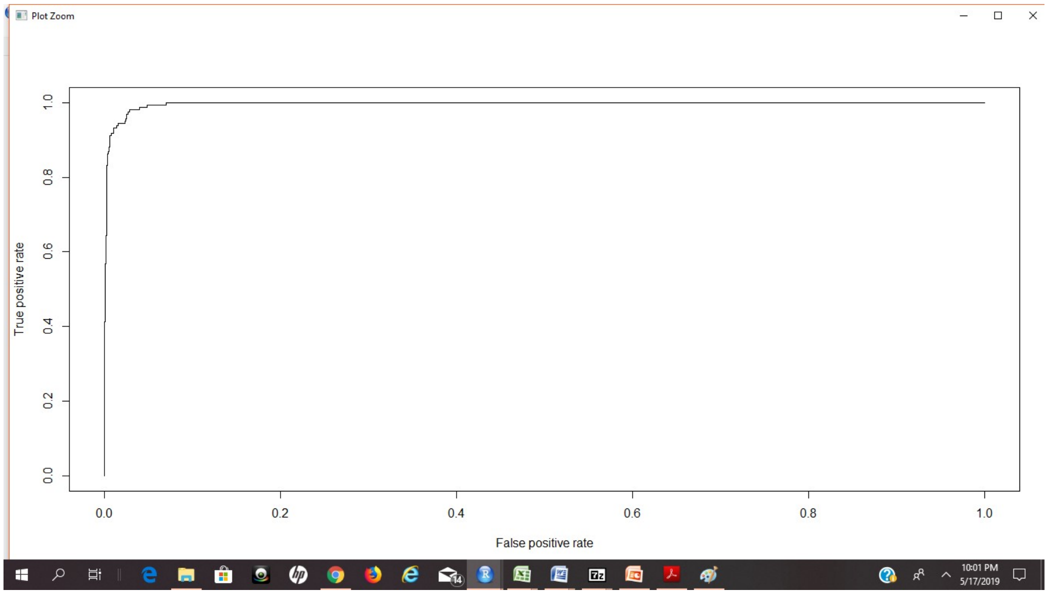
Random Forest model built with Test data set is:

```
> print(RFmodel1)

Call:
 randomForest(formula = PersonalLoan ~ ., data = test_bank_data,      mtry = 3, nodesize = 10, ntree = 100
, importance = TRUE)
               Type of random forest: regression
                     Number of trees: 100
No. of variables tried at each split: 3

          Mean of squared residuals: 0.01875294
                    % Var explained: 80.32
```

Confusion matrix ROCR plots for Random Forest models of Train and Test data are shown below.



The AUC calculated as per the above plot for Test Data set is: 0. 9999512



The AUC calculated as per the above plot for Test Data set is: `0.9966931`

Both AUC values are almost equal which shows the model fits well.
Please refer Appendix A for Source Code.

KS values for Train and Test Data are shown below.

```
#KS on Train
> KS = max(attr(perf1, 'y.values')[[1]]-attr(perf1, 'x.values')[[1]])
> KS
 0.99375

#KS on Test
> KS = max(attr(perf, 'y.values')[[1]]-attr(perf, 'x.values')[[1]])
> KS
 0.9528918
```

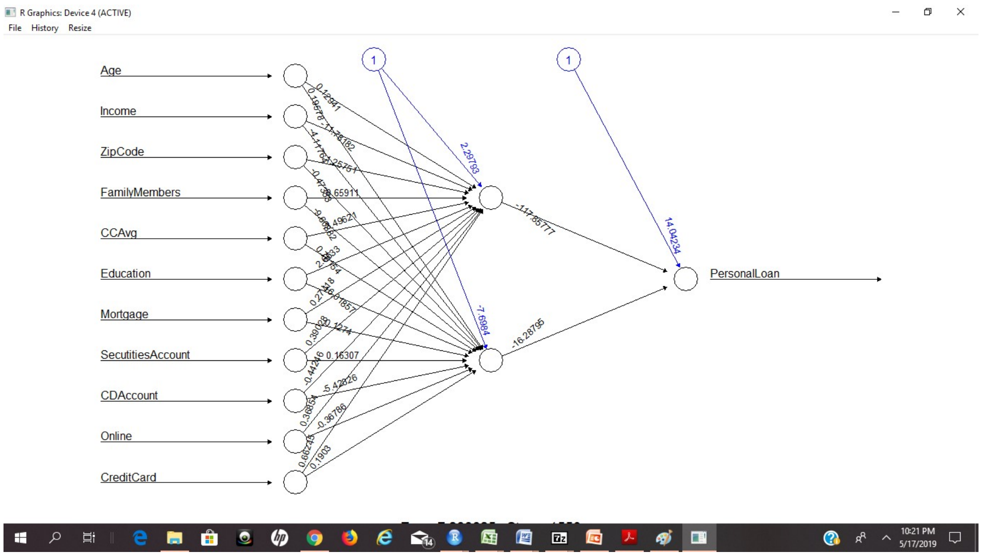
As per the above values, the model seems to be over fitting.

Gini Values for Train and Test Data sets are shown below:

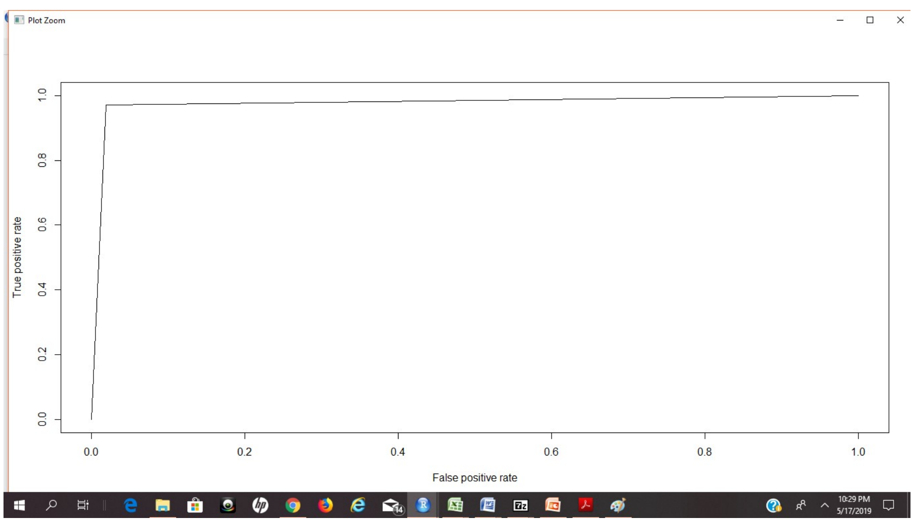
```
#Gini For Train
> gini = ineq(predRF1, type = "Gini")
> gini
 0.4767935

> #Gini For Test
> gini = ineq(predRF, type = "Gini")
> gini
0.7061141
```

Gini Value for Train data set seems to be a good fit, however, on test data set, it seems to be over fitting.

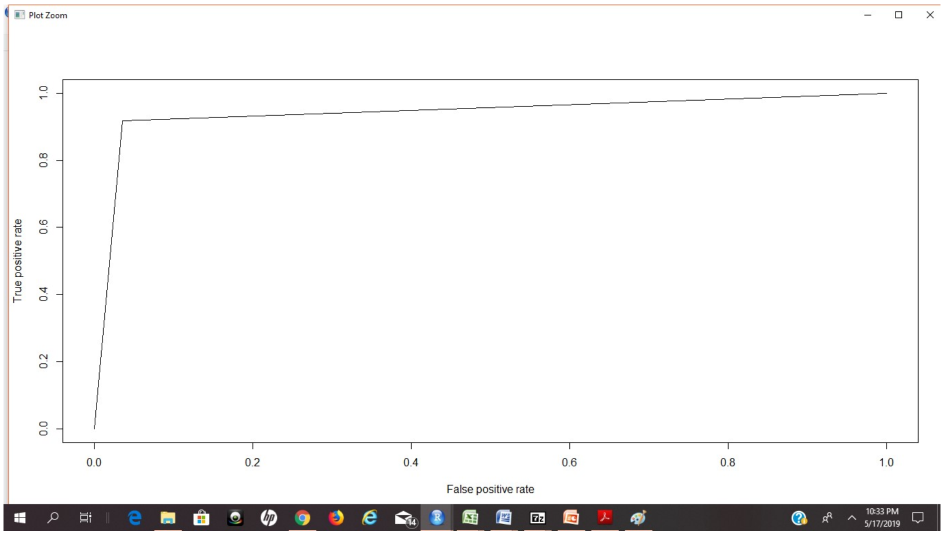
The Neural network built with Train Data is



The Neural Network built with Test data is



The AUC calculated as per the above plot for Train Data set is: 0.9765625



The AUC calculated as per the above plot for Test Data set is: 0.9765625

Both AUC values are almost equal which shows the model fits well.
Please refer Appendix A for Source Code.

KS values of Train Data set of NN model is

```
## deciling Train data
> train.new$deciles <- decile(predNN.round[,1])
> library(ineq)
> KS <- max(attr(perf, 'y.values')[[1]]-attr(perf, 'x.values')[[1]])
> KS
 0.953125
```

KS values of Test Data set of NN Model is

```
test_bank_data$deciles <- decile(predNN1.round[,1])
>
> library(ineq)
> KS <- max(attr(perf1, 'y.values')[[1]]-attr(perf1, 'x.values')[[1]])
> KS
 0.8836754
```

As per the above values, the model seems to be over fitting.

Gini Values for Train and Test Data sets are shown below:

```
#Gini for Train data
> gini = ineq(predNN.round[,1], type="Gini")
> gini
 0.5046875
> #Gini for Test data
> gini = ineq(predNN1.round[,1], type="Gini")
> gini
 0.8706667
```

Gini Value for Train data set seems to be a good fit, however, on test data set, it seems to be over fitting.

3.9. Conclusion

The model performance measures of the various models built are :

CART Model:

Model Performance measures	Train	Test
AUC	0.976875	0.9773298
KS	0.915625	0.9231343
Gini	0.476875	0.8035083

Random Forest Model:

Model Performance measures	Train	Test
AUC	0.9999512	0.9966931
KS	0.99375	0.9528918
Gini	0.4767935	0.7061141

Neural Network Model:

Model Performance measures	Train	Test
AUC	0.9765625	0.9765625
KS	0.953125	0.8836754
Gini	0.5046875	0.8706667

Above results show that the models being built are over fitting and hence there is need to improve the model performance in order meet the business objective.

4. Appendix A – Source Code

```
#==============================================================================
# Data Analysis - Bank Personal Loan
#==============================================================================
#Environment Set up and Data Import
#Set up working Directory
setwd("C:/Users/Radhika/Desktop/R Programming/Project_DataMining")
getwd()
#
#Import the required packages and install them
#install.packages("readxl")
#install.packages("tidyverse")
#install.packages("dplyr")
#install.packages("ROCR")
#install.packages("caret")
#install.packages("ModelMetrics")
#install.packages("corrplot")

library(readxl)
library(tidyverse)
library(dplyr)
library(ROCR)
library(caret)
library(ModelMetrics)
library(corrplot)

#Read the input file
bank_data <- read_excel("Bank_Personal_Loan_Modelling.xlsx", sheet = "Bank_Personal_Loan_Modelling")
```

```
bank_data <- read_excel("Bank_Personal_Loan_Modelling.xlsx", sheet = "Bank_Personal_Loan_Modelling")
attach(bank_data)
view(bank_data)
sum(is.na(bank_data))
#Find the internal structure of the data
str(bank_data)
#Find the descriptive statistics of the data
summary(bank_data)
#Remove target variable before scaling the data
temp_data <-  bank_data %>% select (-`Personal Loan`)
str(temp_data)
new_bank_data = scale(temp_data[,-1])
summary(new_bank_data)
new_bank_data = as.data.frame(new_bank_data)
str(new_bank_data)

cor.mat <- cor(new_bank_data)
corrplot(cor.mat, type = "lower", method = "number")

### Experience is highly correlated with Age
## For now we decide to remove Experience
new_bank_data <-  new_bank_data %>% select (-`Experience (in years)`)
dim(new_bank_data)
names(new_bank_data)
view(new_bank_data)
#Add the Target variable that was removed while scaling the data
new_bank_data = cbind(new_bank_data, `Personal Loan` = bank_data$`Personal Loan`)
str(new_bank_data)
### Check the proportion of data in bank_data

nrow(subset(new_bank_data, `Personal Loan` == 1))/nrow(new_bank_data)

set.seed(1000)
train_indx <- sample(c(1:nrow(new_bank_data)), round(nrow(new_bank_data) * 0.7,0), replace = FALSE)
```

```
train_indx <- sample(c(1:nrow(new_bank_data)), round(nrow(new_bank_data) * 0.7,0), replace = FALSE)
train_bank_data <- new_bank_data[train_indx, ]
test_bank_data <- new_bank_data[-train_indx, ]
sum(is.na(train_bank_data))
sum(is.na(test_bank_data))
dim(train_bank_data)
dim(test_bank_data)

train.pos <- subset(train_bank_data, `Personal Loan` == 1)
nrow(train.pos)

train.neg <- subset(train_bank_data, `Personal Loan` == 0)
nrow(train.neg)

dim(train.pos)
dim(train.neg)

set.seed(200)    ## Set the seed

## Take the sample subset from the major class (here negative)
train.neg.sub_idx <- sample(c(1:nrow(train.neg)), nrow(train.pos), replace = FALSE)
train.new <- train.neg[train.neg.sub_idx,]
dim(train.new)
sum(is.na(train.new))
train.new <- rbind(train.new, train.pos)  ## Merge the negative and positive cases
dim(train.new)

train.new <- train.new[sample(1:nrow(train.new)),]

### Now check the proportion of Attrition in the sample
## in train_data
nrow(subset(train_bank_data, `Personal Loan` == 1))/nrow(train_bank_data)
## in train.new
nrow(subset(train.new, `Personal Loan` == 1))/nrow(train.new)
nrow(subset(train.new, `Personal Loan` == 1))/nrow(train.new)

###CART Model
library(rpart)
install.packages("rpart.plot")
library(rpart.plot)
r.ctrl = rpart.control(minsplit = 100, minbucket = 10, cp=0, xval=10)
DTModel = rpart(`Personal Loan`~.,data=train.new, method="class", control = r.ctrl)
rpart.plot(DTModel)
attributes(DTModel)
DTModel$cptable
ptree = prune(DTModel, 0.0218750, "CP")
ptree$variable.importance
#CART Validation on Train Data
predTrain = predict(ptree, newdata = train.new)
view(predTrain)
predDT = predict(ptree, newdata = test_bank_data)
view(predDT)
#
library(ROCR)
#Validation on Train data
DTpredROC1 = ROCR::prediction(predTrain[,2], train.new$`Personal Loan`)
perf1 = performance(DTpredROC1, "tpr", "fpr")
plot(perf1)
as.numeric(performance(DTpredROC1, "auc")@y.values)
#
#Validation on Test Data
DTpredROC = ROCR::prediction(predDT[,2], test_bank_data$`Personal Loan`)
perf = performance(DTpredROC, "tpr", "fpr")
plot(perf)
as.numeric(performance(DTpredROC, "auc")@y.values)
```

```r
#Decile code
decile <- function(x){
  deciles <- vector(length=10)
  for (i in seq(0.1,1,.1)){
    deciles[i*10] <- 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, 10
                                                            ))))))))))
}

train.new$deciles = decile(predTrain[,2])
predTrain[,2]
test_bank_data$deciles = decile(predDT[,2])
predDT[,2]
#KS on Train
install.packages("ineq")
library(ineq)
#KS on Train
KS = max(attr(perf1, 'y.values')[[1]]-attr(perf1, 'x.values')[[1]])
KS
#KS on Test
KS = max(attr(perf, 'y.values')[[1]]-attr(perf, 'x.values')[[1]])
KS
#Gini For Train
gini = ineq(predTrain[,2], type = "Gini")
gini
#Gini for Test
gini = ineq(predDT[,2], type="Gini")
gini
#RandomForest
install.packages("randomForest")
library(randomForest)
names(train.new)
#sum(is.na(train.new$`Family members`))
train.new =  train.new %>% select (-`deciles`)
test_bank_data =  test_bank_data %>% select (-`deciles`)
colnames(train.new) = c("Age", "Income", "ZipCode", "FamilyMembers", "CCAvg", "Education", "Mortgage", "SecutitiesAccount", "CDAccount", "Online", "CreditCard
colnames(test_bank_data) = c("Age", "Income", "ZipCode", "FamilyMembers", "CCAvg", "Education", "Mortgage", "SecutitiesAccount", "CDAccount", "Online", "Credi
RFmodel = randomForest(`PersonalLoan`~ ., data=train.new[,-12], mtry = 3, nodesize = 10, ntree = 100, importance = TRUE)
print(RFmodel)
names(test_bank_data)
view(test_bank_data)
RFmodel1 = randomForest(PersonalLoan ~ ., data=test_bank_data[,-12], mtry = 3, nodesize = 10, ntree = 100, importance = TRUE)
print(RFmodel1)

#Validate RF Model on Train Data
predRF1 = predict(RFmodel, newdata = train.new)
predRF1.round = round(predRF1, 0)
predRF1.round
table(train.new$PersonalLoan, predRF1.round)
mean(train.new$PersonalLoan==predRF1.round)

RFPredROC1 = ROCR::prediction(predRF1, train.new$PersonalLoan)
perf1 = performance(RFPredROC1,"tpr","fpr")
plot(perf1)
as.numeric(performance(RFPredROC1, "auc")@y.values)
train.new$deciles = decile(predRF)

#Validate RF Model on Test Data
predRF = predict(RFmodel, newdata = test_bank_data)
predRF.round = round(predRF, 0)
```

```r
predRF.round = round(predRF, 0)
predRF.round
table(test_bank_data$PersonalLoan, predRF.round)
mean(test_bank_data$PersonalLoan==predRF.round)

RFPredROC = ROCR::prediction(predRF, test_bank_data$PersonalLoan)
perf = performance(RFPredROC,"tpr","fpr")
plot(perf)
as.numeric(performance(RFPredROC, "auc")@y.values)
test_bank_data$deciles = decile(predRF)

#KS on Train
install.packages("ineq")
library(ineq)

#KS on Train
KS = max(attr(perf1, 'y.values')[[1]]-attr(perf1, 'x.values')[[1]])
KS
#KS on Test
KS = max(attr(perf, 'y.values')[[1]]-attr(perf, 'x.values')[[1]])
KS
#Gini For Train
gini = ineq(predRF1, type = "Gini")
gini
#Gini For Test
gini = ineq(predRF, type = "Gini")
gini
#Neural Network model
install.packages("neuralnet")
library(neuralnet)

names(train.new)
attach(train.new)
attach(test_bank_data)
#NN Built with Train data
NNmodel = neuralnet(formula = PersonalLoan ~ Age+Income+ZipCode+FamilyMembers+CCAvg+Education+Mortgage+SecutitiesAccount+CDAccount+Online+CreditCard, data =
                    linear.output = FALSE,
                    lifesign = "full",
                    lifesign.step = 10,
                    threshold = 0.01,
                    stepmax = 2000)
plot(NNmodel)

#NN built with Test data
NNmodel1 = neuralnet(formula = PersonalLoan ~ Age+Income+ZipCode+FamilyMembers+CCAvg+Education+Mortgage+SecutitiesAccount+CDAccount+Online+CreditCard, data =
                    linear.output = FALSE,
                    lifesign = "full",
                    lifesign.step = 10,
                    threshold = 0.01,
                    stepmax = 2000)
plot(NNmodel1)
str(train.new)
names(test_bank_data)
#For Train data
predNN = neuralnet::compute(NNmodel, train.new)
predNN
predNN.round = round(predNN$net.result, 0)
predNN.round

table(train.new$PersonalLoan, predNN.round[,1])
mean(train.new$PersonalLoan==predNN.round)
library(ROCR)
NNpredROC = ROCR::prediction(predNN.round[,1], train.new$PersonalLoan)
perf = performance(NNpredROC, "tpr", "fpr")
plot(perf)
as.numeric(performance(NNpredROC, "auc")@y.values)
#For Test data
predNN1 = neuralnet::compute(NNmodel, test_bank_data)
predNN1
```

```r
predNN1.round = round(predNN1$net.result, 0)
predNN1.round

table(test_bank_data$PersonalLoan, predNN1.round[,1])
mean(test_bank_data$PersonalLoan==predNN1.round)
library(ROCR)
NNpredROC1 = ROCR::prediction(predNN1.round[,1], test_bank_data$PersonalLoan)
perf1 = performance(NNpredROC1, "tpr", "fpr")
plot(perf1)
as.numeric(performance(NNpredROC, "auc")@y.values)
## deciling Train data
train.new$deciles <- decile(predNN.round[,1])

library(ineq)
KS <- max(attr(perf, 'y.values')[[1]]-attr(perf, 'x.values')[[1]])
KS

## deciling Test data
test_bank_data$deciles <- decile(predNN1.round[,1])

library(ineq)
KS <- max(attr(perf1, 'y.values')[[1]]-attr(perf1, 'x.values')[[1]])
KS
#Gini for Train data
gini = ineq(predNN.round[,1], type="Gini")
gini
#Gini for Test data
gini = ineq(predNN1.round[,1], type="Gini")
gini
```