USE CASE STUDY REPORT

Group No.: Group 03

Student Names: Mayur Darade and Hanoz Patel

Executive Summary

One of the major sources of revenue generation any Bank/Financial institution is to attract public's saving into the Bank. There is a huge competition among the Banks in increasing the consumer base. Due to improvement in media and technology, there is an increase in the amount of information at our figure tips. The marketing team of Banks want to successfully advertise their product most effectively with the highest consumers buying their products. With the advancement in data mining techniques and availability of the data, Banks are adopting a data-driven approach to effectively advertise their products. Using the data collected from the previous marketing campaign number of features about the consumer, marketing campaign and the market condition will be explored. The goal is to use supervised machine learning algorithm to build a classification model of the dataset to provide a suggestion for marketing campaign team to improve the marketing campaign. The goal is to predict whether a client will subscribe to a term deposit (variable y) with the help of a given set of dependent variables.

The dataset is a real-world data of Portuguese banking institution's marketing campaign. The data is related to direct marketing campaigns (phone calls) of a Portuguese banking institution. The dataset contains over 45000 rows and 17 attributes. The dataset was converted from text to CSV file. The dataset consists of categorical variables like Education categories, Job categories, months and numeric variables like bank balance, age. There are sixteen independent variables and one dependent variable called "y". The categorical variables were converted in factors and the numeric data was normalized. After preprocessing the data classification algorithms like Logistic Regression, SVM, K-NN, Decision tree, Random Forest and Naïve Bayes were used to build the models. Performance accuracy measures like ROC curve, confusion matrix and Accuracy were used to check the performance of the models.

Almost all attribute contributed to building binary classification models. The performance among all the approach we used, Logistic Regression gave us the best results with 89.99% accuracy. This model is powerful yet easy to implement. The duration of the call between the sales representative and the potential client is the important feature which we came to know through Decision Tree and Random Forest models. Our recommendation to the marketing team of the Bank is to focus on the duration of the call. The marketing team can focus on the group of consumers who are spending more than 375 seconds on the call as they are more likely to open a tern deposit. The marketing team can prepare a series of question that they can ask the customer so that the consumer will spend more time on the phone call. This will increase the success rate of the marketing campaign eventually increasing the Bank's revenue.

I. Background and Introduction

The problem:
Banks/financial institution invest heavily in marketing their products. The marketing team tries to approach a pool of potential consumers. But the number of potential consumers buying the product is very less. Thus, a strategic data-driven approach is needed to target potential consumer, so the marketing campaign is successful.

The goal of the study:
This case study will highlight the importance of marketing in the banks and thus the importance of the role of phone calls in this operation. The goal is to use data collected by the Bank's marketing team from previous marketing campaign and implement machine learning algorithms. Thus, helping the marketing team effectively advertise their products and setup a successful marketing campaign.

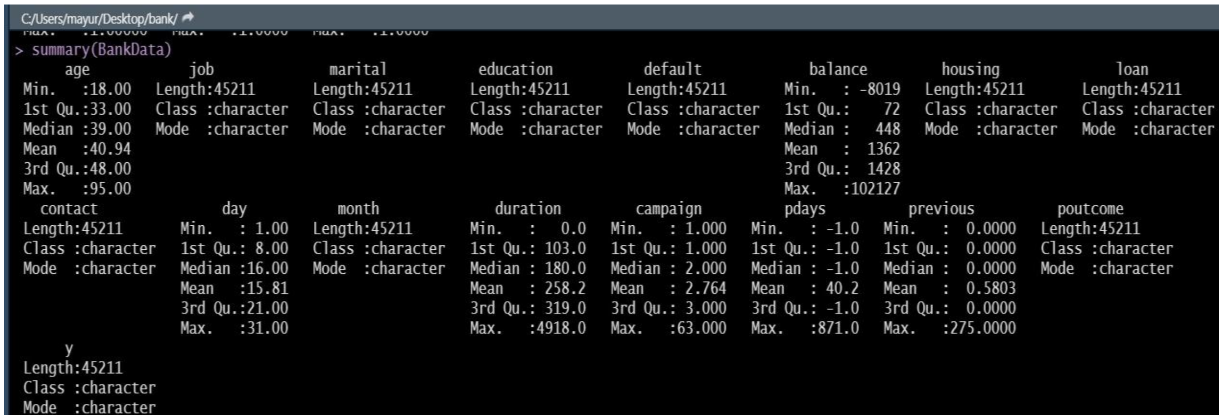
The possible solution:
There are various attributes in the dataset that can be utilized to determine the target customer. Our aim is to build a model for predicting whether a potential customer will buy a term deposit or not.

II. Data Exploration and Visualization

We checked the dimension and structure of the dataset. We observed that 10 columns are categorical and 7 are numeric.

```
#Checking the dimension of the dataset
> dim(BankData)
[1] 45211    17

#Checking Structure of the data
> str(BankData)

'data.frame':  45211 obs. of  17 variables:
 $ age      : int  58 44 33 47 33 35 28 42 58 43 ...
 $ job      : chr  "management" "technician" "entrepreneur" "blue-collar" ...
 $ marital  : chr  "married" "single" "married" "married" ...
 $ education: chr  "tertiary" "secondary" "secondary" "unknown" ...
 $ default  : chr  "no" "no" "no" "no" ...
 $ balance  : int  2143 29 2 1506 1 231 447 2 121 593 ...
 $ housing  : chr  "yes" "yes" "yes" "yes" ...
 $ loan     : chr  "no" "no" "yes" "no" ...
 $ contact  : chr  "unknown" "unknown" "unknown" "unknown" ...
 $ day      : int  5 5 5 5 5 5 5 5 5 5 ...
 $ month    : chr  "may" "may" "may" "may" ...
 $ duration : int  261 151 76 92 198 139 217 380 50 55 ...
 $ campaign : int  1 1 1 1 1 1 1 1 1 1 ...
 $ pdays    : int  -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 ...
 $ previous : int  0 0 0 0 0 0 0 0 0 0 ...
 $ poutcome : chr  "unknown" "unknown" "unknown" "unknown" ...
  $ y        : chr  "no" "no" "no" "no" ...
```

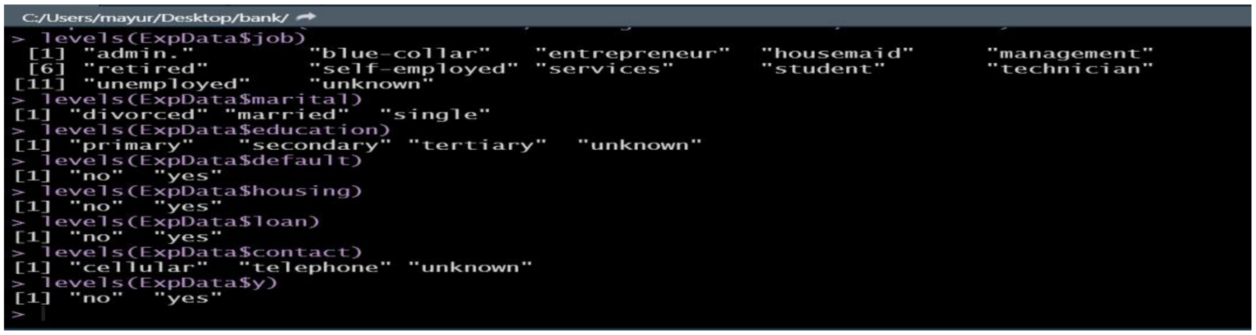
We ran the summary of the dataset to get more insights about the data. We came to know that the numeric columns are of different scale and magnitude and there is a need to normalize the data, so that no one feature will have more weightage over the classification.



C:/Users/mayur/Desktop/bank/

```
> summary(BankData)
      age            job              marital            education           default             balance          housing              loan          
 Min.   :18.00   Length:45211       Length:45211       Length:45211       Length:45211       Min.   : -8019   Length:45211       Length:45211      
 1st Qu.:33.00   Class :character   Class :character   Class :character   Class :character   1st Qu.:    72   Class :character   Class :character  
 Median :39.00   Mode  :character   Mode  :character   Mode  :character   Mode  :character   Median :   448   Mode  :character   Mode  :character  
 Mean   :40.94                                                                               Mean   :  1362                                        
 3rd Qu.:48.00                                                                               3rd Qu.:  1428                                        
 Max.   :95.00                                                                               Max.   :102127                                        
   contact              day           month              duration        campaign          pdays           previous           poutcome        
 Length:45211       Min.   : 1.00   Length:45211       Min.   :   0.0   Min.   : 1.000   Min.   : -1.0   Min.   :  0.0000   Length:45211      
 Class :character   1st Qu.: 8.00   Class :character   1st Qu.: 103.0   1st Qu.: 1.000   1st Qu.: -1.0   1st Qu.:  0.0000   Class :character  
 Mode  :character   Median :16.00   Mode  :character   Median : 180.0   Median : 2.000   Median : -1.0   Median :  0.0000   Mode  :character  
                    Mean   :15.81                      Mean   : 258.2   Mean   : 2.764   Mean   : 40.2   Mean   :  0.5803                     
                    3rd Qu.:21.00                      3rd Qu.: 319.0   3rd Qu.: 3.000   3rd Qu.: -1.0   3rd Qu.:  0.0000                     
                    Max.   :31.00                      Max.   :4918.0   Max.   :63.000   Max.   :871.0   Max.   :275.0000                     
      y            
 Length:45211      
 Class :character  
 Mode  :character  
```

We checked the levels of the categorical attributes and decided to convert these categories into factors.

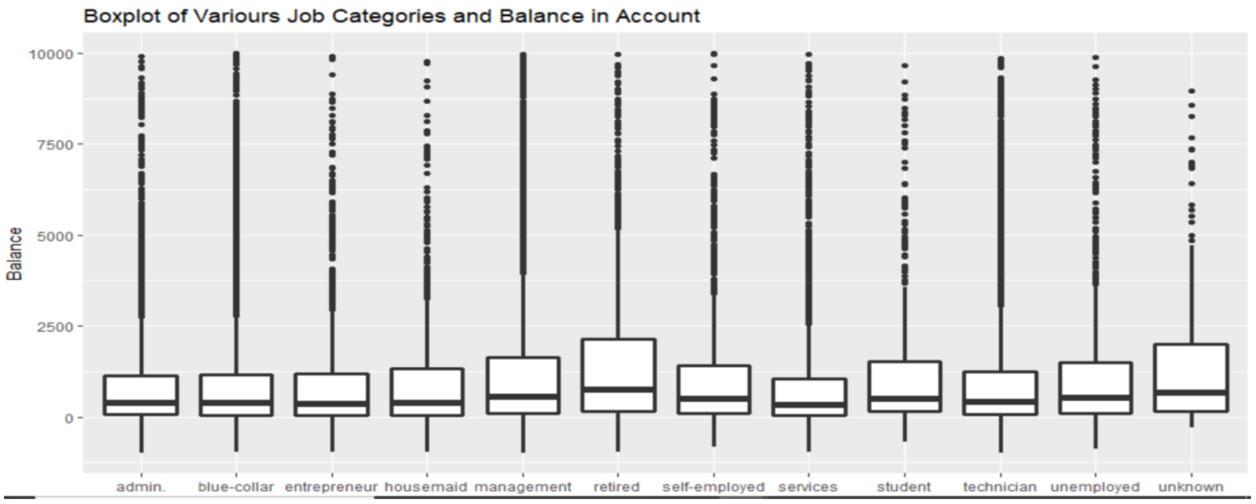


C:/Users/mayur/Desktop/bank/

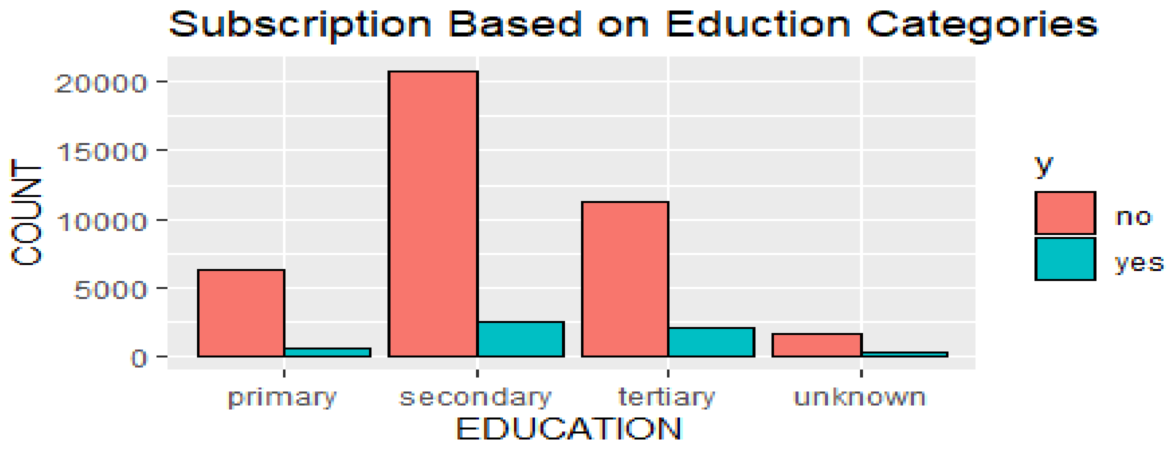
```
> levels(ExpData$job)
 [1] "admin."        "blue-collar"   "entrepreneur"  "housemaid"     "management"   
 [6] "retired"       "self-employed" "services"      "student"       "technician"   
[11] "unemployed"    "unknown"      
> levels(ExpData$marital)
[1] "divorced" "married"  "single"  
> levels(ExpData$education)
[1] "primary"   "secondary" "tertiary"  "unknown"  
> levels(ExpData$default)
[1] "no"  "yes"
> levels(ExpData$housing)
[1] "no"  "yes"
> levels(ExpData$loan)
[1] "no"  "yes"
> levels(ExpData$contact)
[1] "cellular"  "telephone" "unknown"  
> levels(ExpData$y)
[1] "no"  "yes"
> 
```

Data visualization:

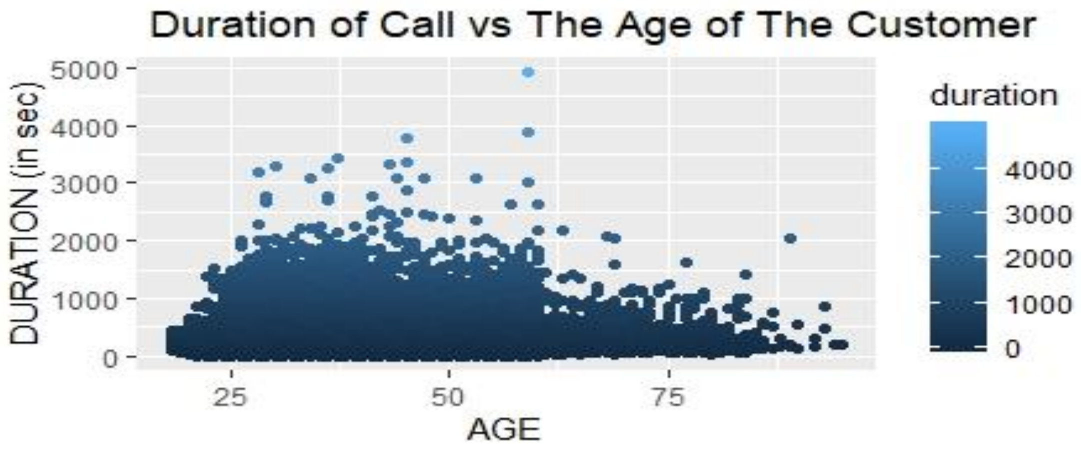
1) This plot shows the bank balance for various job categories.



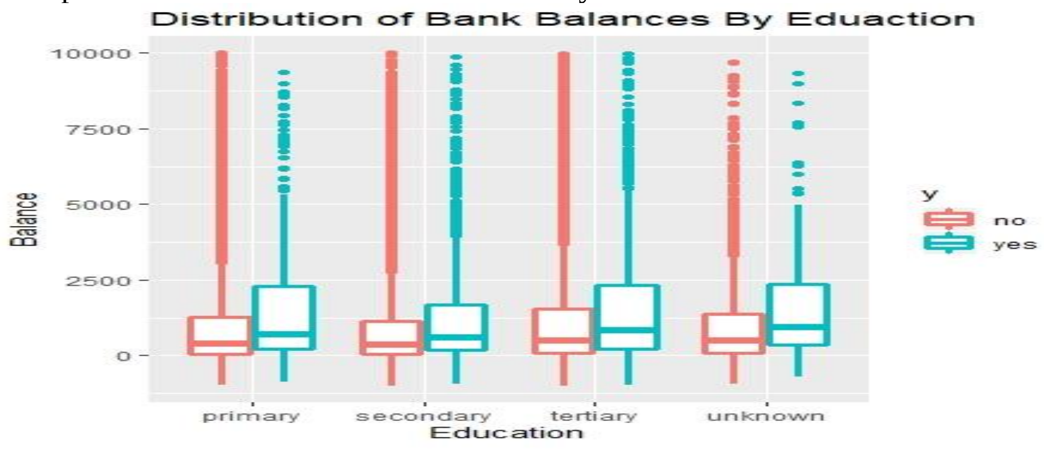
2) The Bar plot shows the number of consumers subscribing to term deposit by their education categories.



3) The scatter plot show duration of the call as per the age of the consumer.

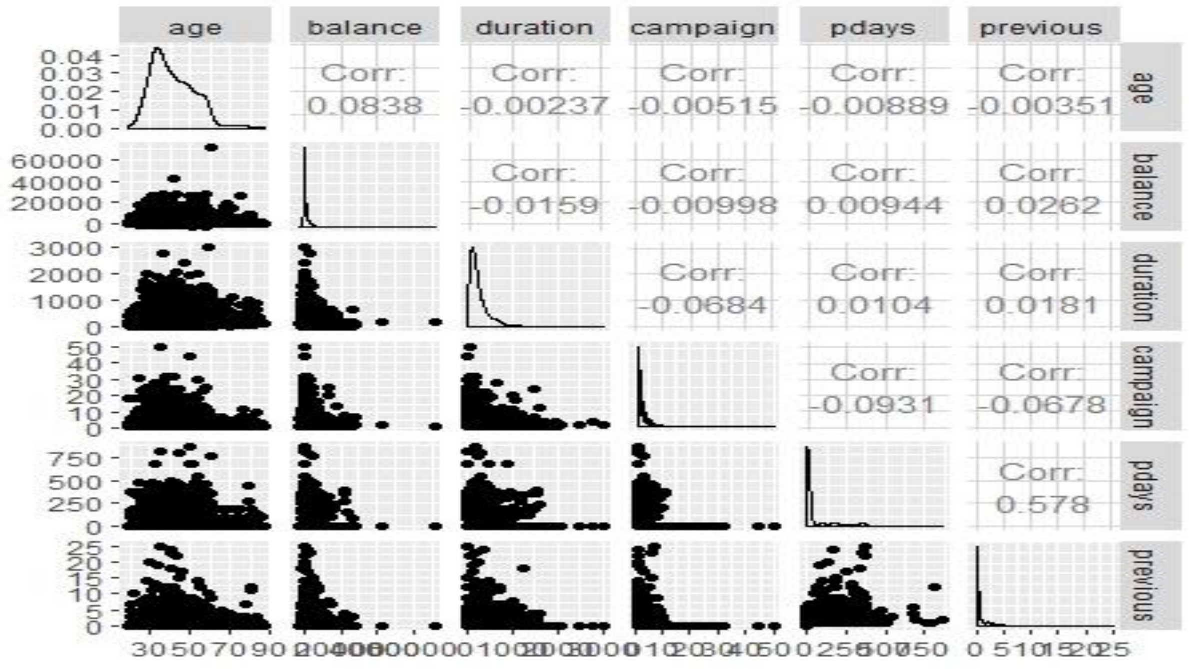


4) The plot shows the distribution of balances by Education.

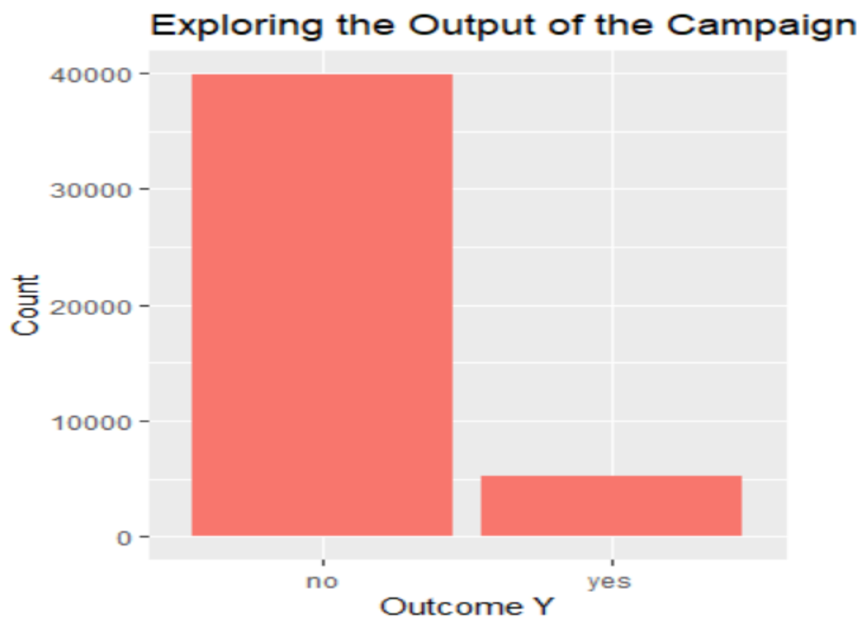


5) The plot shows the correlation of various numerical independent variables.

```
> #install.packages("GGally")
> library(GGally)
> ggpairs(ExpData[, c(1,6,12,13,14,15)])
```



6) The plot shows the total outcome of the marketing campaign.



III. Data Preparation and Preprocessing

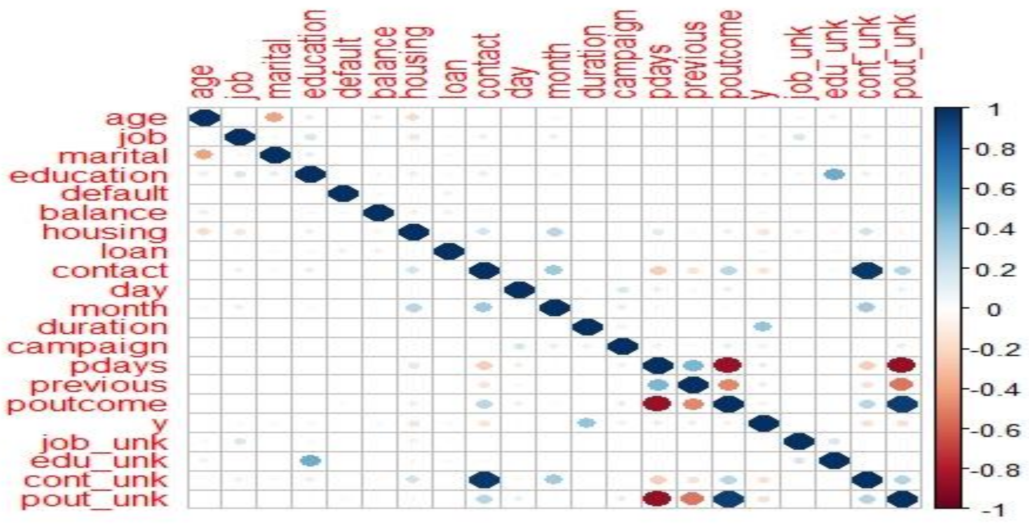
We checked for the missing values in the dataset.

```
> sum(is.na(BankData))
[1] 0
```

The character variables are converted into numeric type. The unknown responses of the variables are also taken in consideration. To account these responses new four variables are created.

```
> BankData$job_unk <- ifelse(BankData$job == "unknown", 1, 0)
> BankData$job <- as.numeric(as.factor(BankData$job))
> BankData$marital <- as.numeric(as.factor(BankData$marital))
> BankData$edu_unk <- ifelse(BankData$education == "unknown", 1, 0)
> BankData$education <- as.numeric(as.factor(BankData$education))
> BankData$default<- ifelse(BankData$default == "yes", 1, 0)
> BankData$housing <- ifelse(BankData$housing== "yes", 1, 0)
> BankData$loan<- ifelse(BankData$loan== "yes", 1, 0)
> BankData$month <- as.numeric(as.factor(BankData$month))
> BankData$cont_unk <- ifelse(BankData$contact == "unknown", 1, 0)
> BankData$contact <- as.numeric(as.factor(BankData$contact))
> BankData$pout_unk <- ifelse(BankData$poutcome == "unknown", 1, 0)
> BankData$poutcome <- as.numeric(as.factor(BankData$poutcome))
> BankData$y <- ifelse(BankData$y== "yes", 1, 0)
```

Checking the Collinearity of the variables



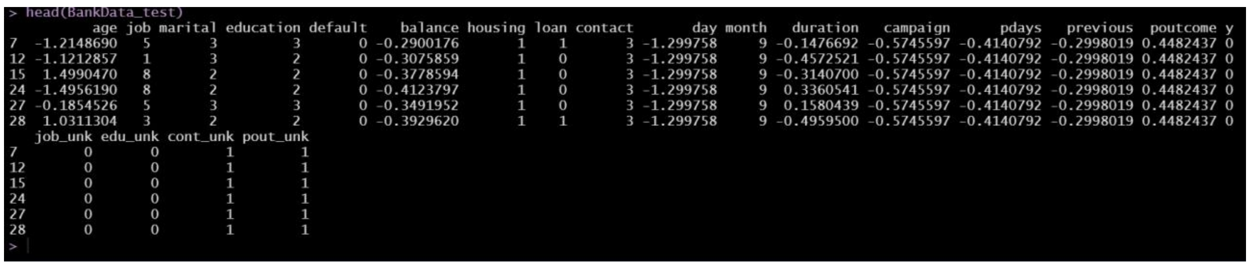
We divided   the dataset into training dataset(75%) and test dataset(25%) for all the models.

```
> #splitting the data into traing and test data
> library(caTools)
> set.seed(100)
> split=sample.split(BankData$y,SplitRatio = 0.75)
> BankData_training = subset(BankData, split==T)
> BankData_test = subset(BankData, split==F)
```

We normalized the data to prevent one feature having weightage over the classification models.

```
#feature scaling
> BankData_training[,c(1,6,10,12,13:16)] =
scale(BankData_training[,c(1,6,10,12,13:16)])
> BankData_test[,c(1,6,10,12,13:16)] =
scale(BankData_test[,c(1,6,10,12,13:16)] )
```

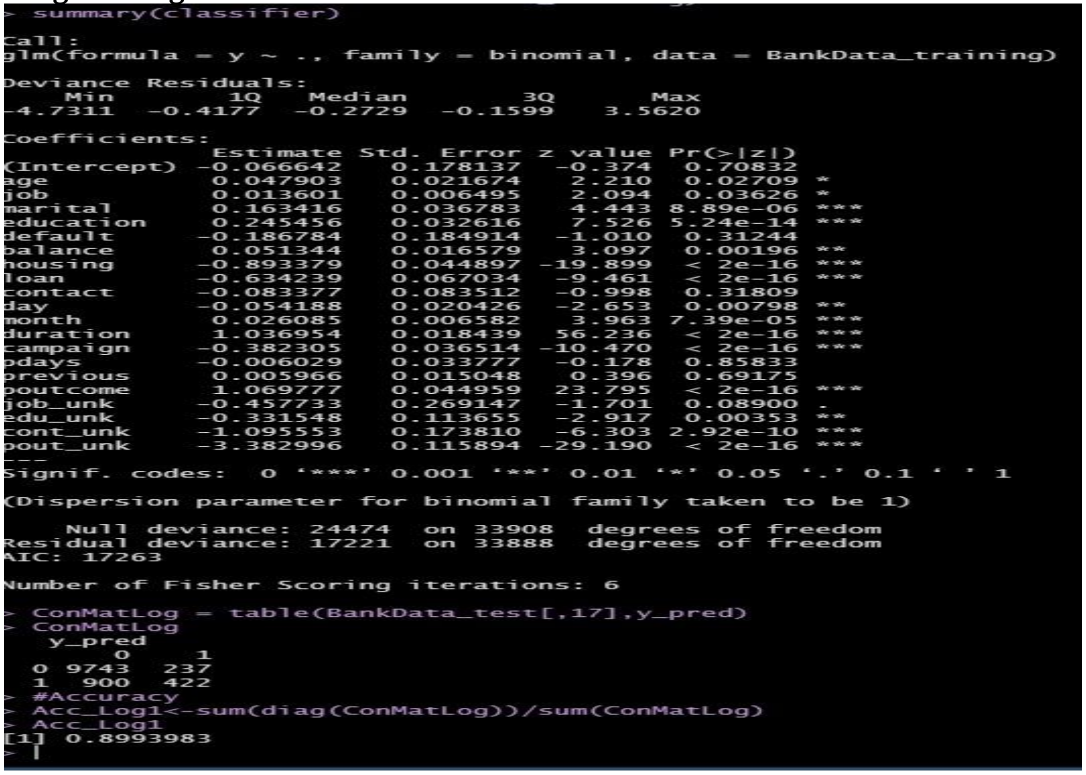
Summary of the processed data ready to use to build models.



```
> head(BankData_test)
          age job marital education default    balance housing loan contact       day month   duration   campaign      pdays   previous  poutcome y
7  -1.2148690   5       3         3       0 -0.2900176       1    1       3 -1.299758     9 -0.1476692 -0.5745597 -0.4140792 -0.2998019 0.4482437 0
12 -1.1212857   1       3         2       0 -0.3075859       1    0       3 -1.299758     9 -0.4572521 -0.5745597 -0.4140792 -0.2998019 0.4482437 0
15  1.4990470   8       2         2       0 -0.3778594       1    0       3 -1.299758     9 -0.3140700 -0.5745597 -0.4140792 -0.2998019 0.4482437 0
24 -1.4956190   8       2         2       0 -0.4123797       1    0       3 -1.299758     9  0.3360541 -0.5745597 -0.4140792 -0.2998019 0.4482437 0
27 -0.1854526   5       3         3       0 -0.3491952       1    0       3 -1.299758     9  0.1580439 -0.5745597 -0.4140792 -0.2998019 0.4482437 0
28  1.0311304   3       2         2       0 -0.3929620       1    1       3 -1.299758     9 -0.4959500 -0.5745597 -0.4140792 -0.2998019 0.4482437 0
   job_unk edu_unk cont_unk pout_unk
7        0       0        1        1
12       0       0        1        1
15       0       0        1        1
24       0       0        1        1
27       0       0        1        1
28       0       0        1        1
> 
```

IV. Data Mining Techniques and Implementation

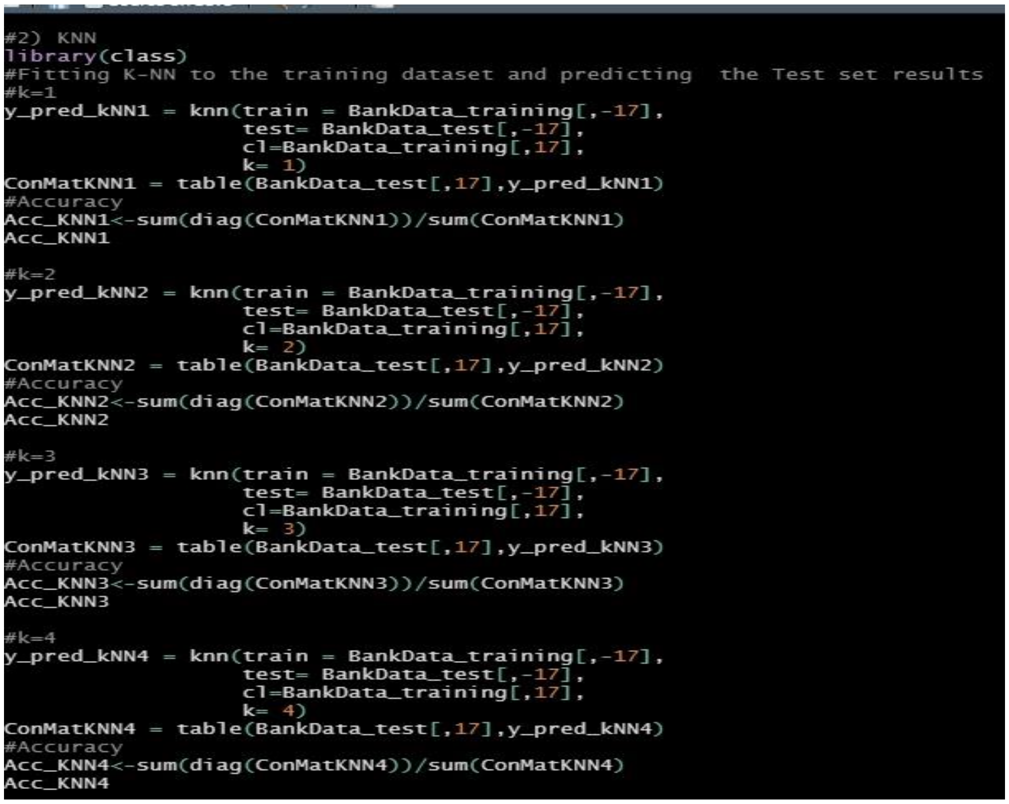
1) Logistic Regression:



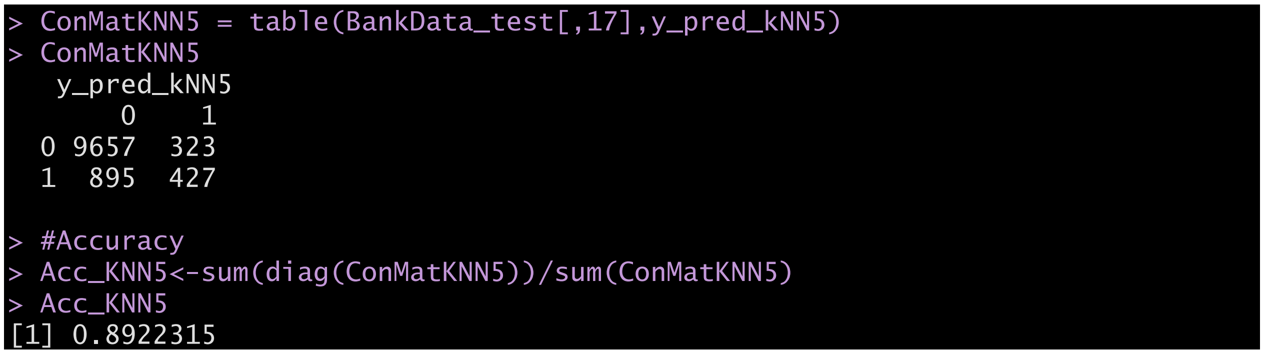
```
> summary(classifier)

Call:
glm(formula = y ~ ., family = binomial, data = BankData_training)

Deviance Residuals: 
    Min       1Q   Median       3Q      Max  
-4.7311  -0.4177  -0.2729  -0.1599   3.5620  

Coefficients:
             Estimate Std. Error z value Pr(>|z|)    
(Intercept) -0.066642   0.178137  -0.374  0.70832    
age          0.047903   0.021674   2.210  0.02709 *  
job          0.013601   0.006495   2.094  0.03626 *  
marital      0.163416   0.036783   4.443 8.89e-06 ***
education    0.245456   0.032616   7.526 5.24e-14 ***
default     -0.186784   0.184914  -1.010  0.31244    
balance      0.051344   0.016579   3.097  0.00196 ** 
housing     -0.893379   0.044897 -19.899  < 2e-16 ***
loan        -0.634239   0.067034  -9.461  < 2e-16 ***
contact     -0.083377   0.083512  -0.998  0.31809    
day         -0.054188   0.020426  -2.653  0.00798 ** 
month        0.026085   0.006582   3.963 7.39e-05 ***
duration     1.036954   0.018439  56.236  < 2e-16 ***
campaign    -0.382305   0.036514 -10.470  < 2e-16 ***
pdays       -0.006029   0.033777  -0.178  0.85833    
previous     0.005966   0.015048   0.396  0.69175    
poutcome     1.069777   0.044959  23.795  < 2e-16 ***
job_unk     -0.457733   0.269147  -1.701  0.08900 .  
edu_unk     -0.331548   0.113655  -2.917  0.00353 ** 
cont_unk    -1.095553   0.173810  -6.303 2.92e-10 ***
pout_unk    -3.382996   0.115894 -29.190  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 24474  on 33908  degrees of freedom
Residual deviance: 17221  on 33888  degrees of freedom
AIC: 17263

Number of Fisher Scoring iterations: 6

> ConMatLog = table(BankData_test[,17],y_pred)
> ConMatLog
   y_pred
       0    1
  0 9743  237
  1  900  422
> #Accuracy
> Acc_Log1<-sum(diag(ConMatLog))/sum(ConMatLog)
> Acc_Log1
[1] 0.8993983
> |
```

We created logistic regression model based on predictor variables in the model and we got 89.99% model accuracy we created second logistic regression model with variables having p-value less than 0.5. The accuracy of the second model was same as the initial model. So, we decide to consider all the attributes in the model.

2) K-NN:



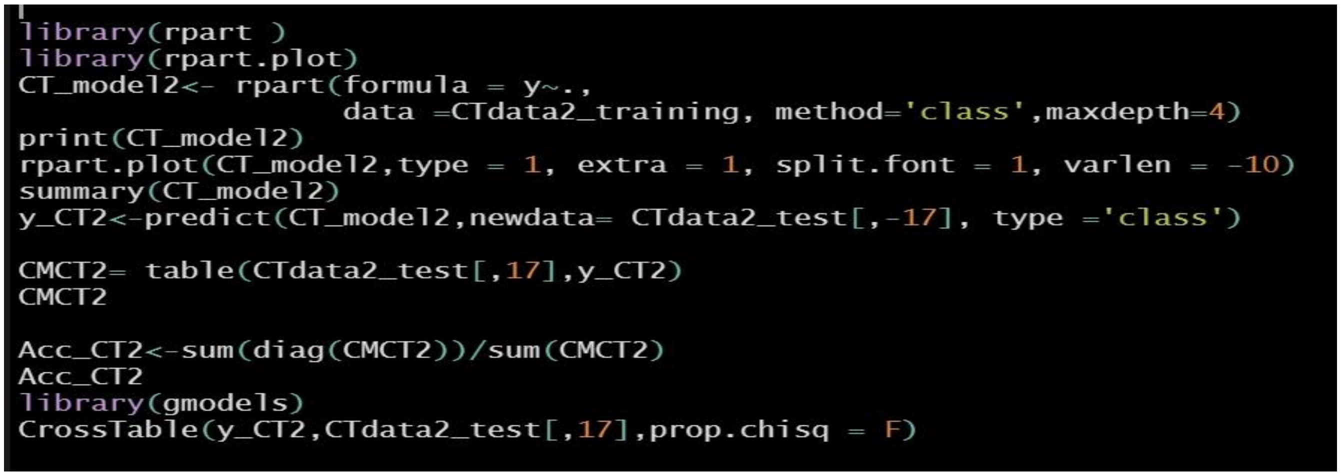
```
#2) KNN
library(class)
#Fitting K-NN to the training dataset and predicting  the Test set results
#k=1
y_pred_kNN1 = knn(train = BankData_training[,-17],
                  test= BankData_test[,-17],
                  cl=BankData_training[,17],
                  k= 1)
ConMatKNN1 = table(BankData_test[,17],y_pred_kNN1)
#Accuracy
Acc_KNN1<-sum(diag(ConMatKNN1))/sum(ConMatKNN1)
Acc_KNN1

#k=2
y_pred_kNN2 = knn(train = BankData_training[,-17],
                  test= BankData_test[,-17],
                  cl=BankData_training[,17],
                  k= 2)
ConMatKNN2 = table(BankData_test[,17],y_pred_kNN2)
#Accuracy
Acc_KNN2<-sum(diag(ConMatKNN2))/sum(ConMatKNN2)
Acc_KNN2

#k=3
y_pred_kNN3 = knn(train = BankData_training[,-17],
                  test= BankData_test[,-17],
                  cl=BankData_training[,17],
                  k= 3)
ConMatKNN3 = table(BankData_test[,17],y_pred_kNN3)
#Accuracy
Acc_KNN3<-sum(diag(ConMatKNN3))/sum(ConMatKNN3)
Acc_KNN3

#k=4
y_pred_kNN4 = knn(train = BankData_training[,-17],
                  test= BankData_test[,-17],
                  cl=BankData_training[,17],
                  k= 4)
ConMatKNN4 = table(BankData_test[,17],y_pred_kNN4)
#Accuracy
Acc_KNN4<-sum(diag(ConMatKNN4))/sum(ConMatKNN4)
Acc_KNN4
```



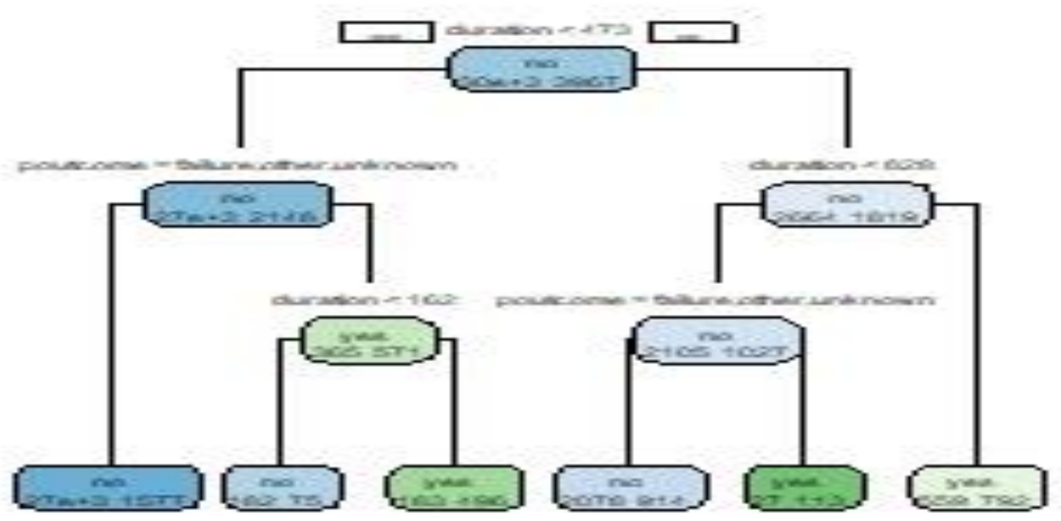
```
> ConMatKNN5 = table(BankData_test[,17],y_pred_kNN5)
> ConMatKNN5
   y_pred_kNN5
       0    1
  0 9657  323
  1  895  427

> #Accuracy
> Acc_KNN5<-sum(diag(ConMatKNN5))/sum(ConMatKNN5)
> Acc_KNN5
[1] 0.8922315
```

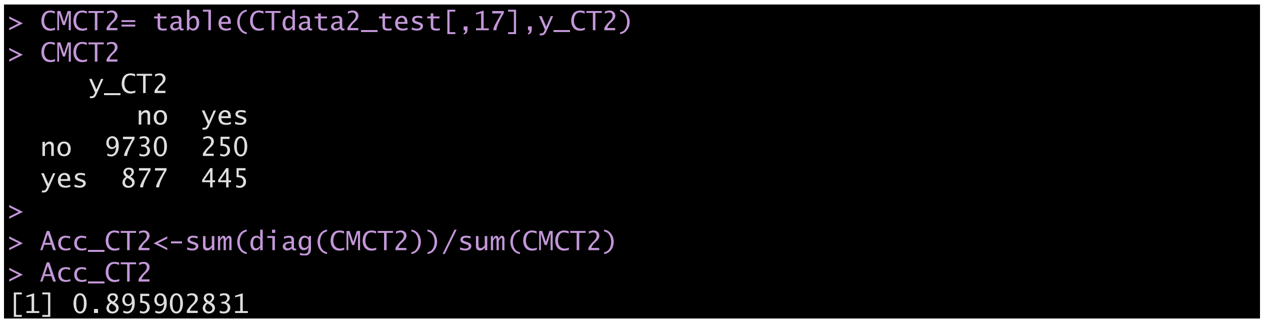
For k=5 we are getting the best accuracy of 89.22% accuracy. We can select k = 10 as a best k value.

3) Decision Tree:



```
library(rpart )
library(rpart.plot)
CT_model2<- rpart(formula = y~.,
                  data =CTdata2_training, method='class',maxdepth=4)
print(CT_model2)
rpart.plot(CT_model2,type = 1, extra = 1, split.font = 1, varlen = -10)
summary(CT_model2)
y_CT2<-predict(CT_model2,newdata= CTdata2_test[,-17], type ='class')

CMCT2= table(CTdata2_test[,17],y_CT2)
CMCT2

Acc_CT2<-sum(diag(CMCT2))/sum(CMCT2)
Acc_CT2
library(gmodels)
CrossTable(y_CT2,CTdata2_test[,17],prop.chisq = F)
```

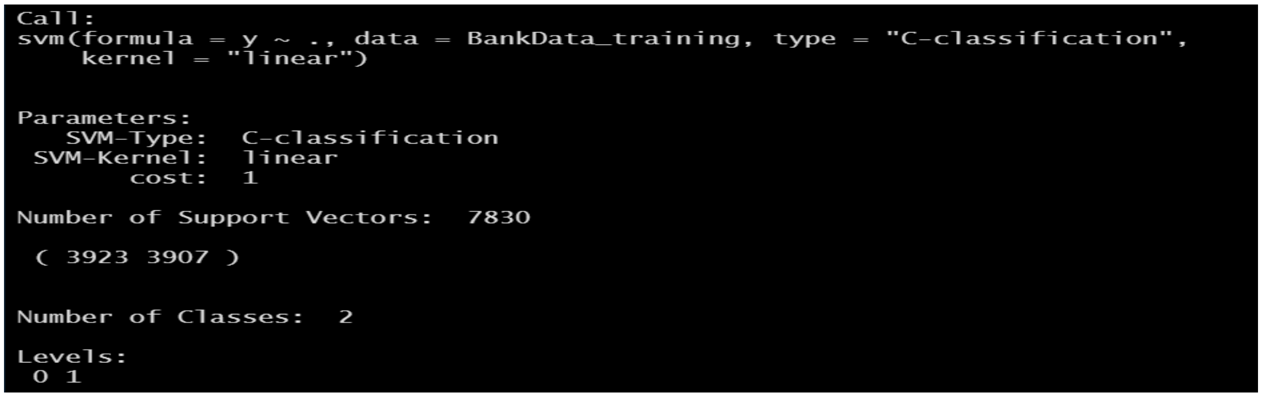




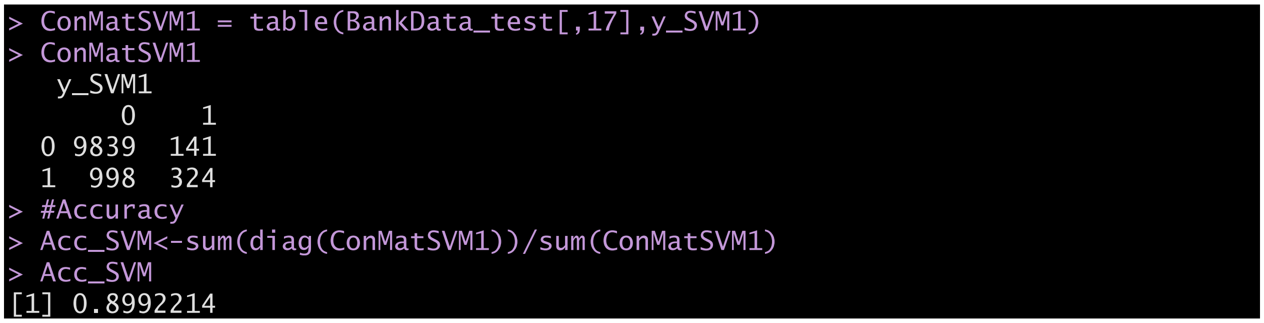
```
> CMCT2= table(CTdata2_test[,17],y_CT2)
> CMCT2
     y_CT2
        no  yes
  no  9730  250
  yes  877  445
>
> Acc_CT2<-sum(diag(CMCT2))/sum(CMCT2)
> Acc_CT2
[1] 0.895902831
```

Considering all the predictors we got accuracy of 89.59%, which is less than the logistic regression model

4) Support Vector Machine :



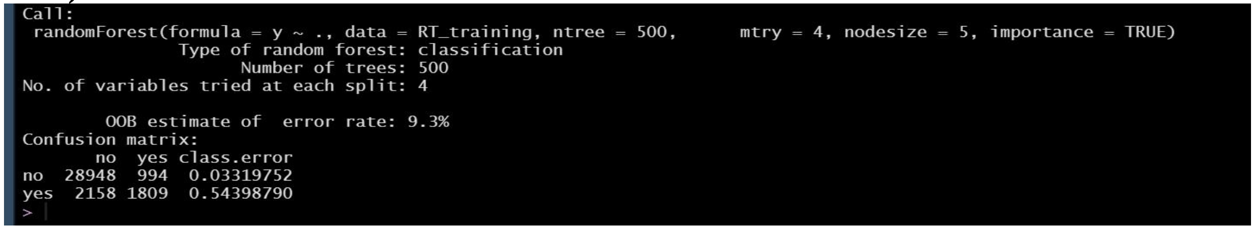
```
Call:
svm(formula = y ~ ., data = BankData_training, type = "C-classification",
    kernel = "linear")


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  1

Number of Support Vectors:  7830

 ( 3923 3907 )


Number of Classes:  2

Levels:
 0 1
```



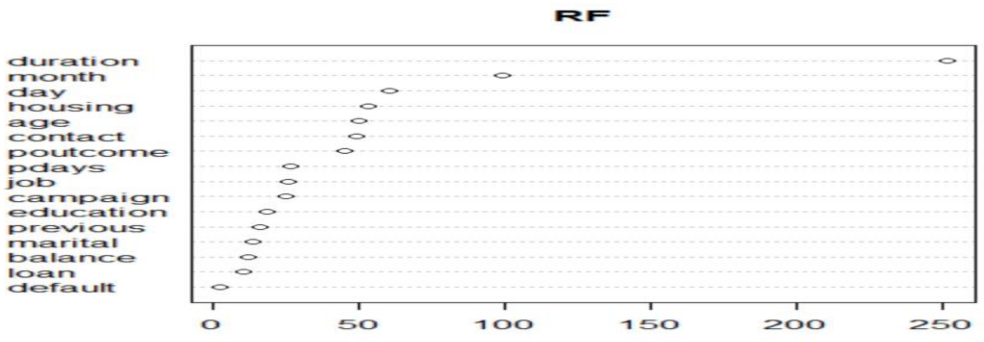
```
> ConMatSVM1 = table(BankData_test[,17],y_SVM1)
> ConMatSVM1
   y_SVM1
       0    1
  0 9839  141
  1  998  324
> #Accuracy
> Acc_SVM<-sum(diag(ConMatSVM1))/sum(ConMatSVM1)
> Acc_SVM
[1] 0.8992214
```

For support vector machine using linear kernel we are getting 89.99% accuracy, which is almost same as logistic regression.

5) Random Forest:



```
Call:
 randomForest(formula = y ~ ., data = RT_training, ntree = 500,      mtry = 4, nodesize = 5, importance = TRUE)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 4

        OOB estimate of  error rate: 9.3%
Confusion matrix:
       no  yes class.error
no   28948  994  0.03319752
yes   2158 1809  0.54398790
>
```

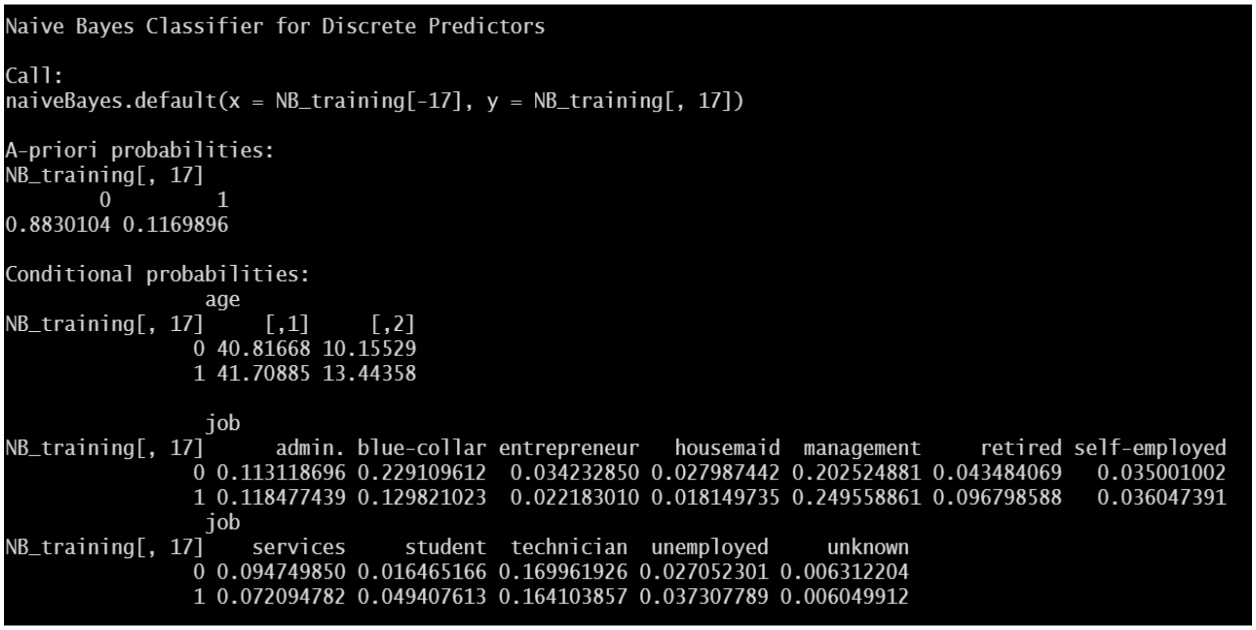


From the above chart we can see that the duration of the call between the sales representative and potential costumer is important feature.

```
> CMRT= table(RT_test[,17],rf.pred)
> CMRT
     rf.pred
        no  yes
  no  9630  350
  yes  701  621
>
> Acc_RT<-sum(diag(CMRT))/sum(CMRT)
> Acc_RT
[1] 0.9070076
```

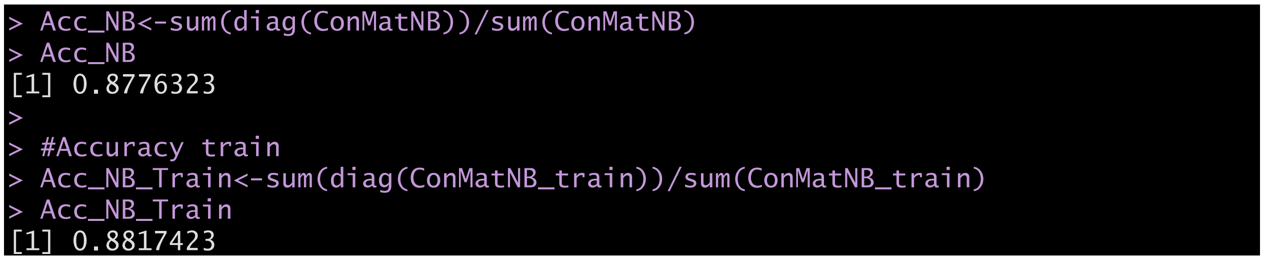
For Random Forest we got the best accuracy of 90.7%

6) Naïve Bayes:



```
Naive Bayes Classifier for Discrete Predictors

Call:
naiveBayes.default(x = NB_training[-17], y = NB_training[, 17])

A-priori probabilities:
NB_training[, 17]
        0         1
0.8830104 0.1169896

Conditional probabilities:
                 age
NB_training[, 17]     [,1]     [,2]
                0 40.81668 10.15529
                1 41.70885 13.44358

                 job
NB_training[, 17]      admin. blue-collar entrepreneur   housemaid  management     retired self-employed
                0 0.113118696 0.229109612  0.034232850 0.027987442 0.202524881 0.043484069   0.035001002
                1 0.118477439 0.129821023  0.022183010 0.018149735 0.249558861 0.096798588   0.036047391
                 job
NB_training[, 17]    services     student  technician  unemployed     unknown
                0 0.094749850 0.016465166 0.169961926 0.027052301 0.006312204
                1 0.072094782 0.049407613 0.164103857 0.037307789 0.006049912
```

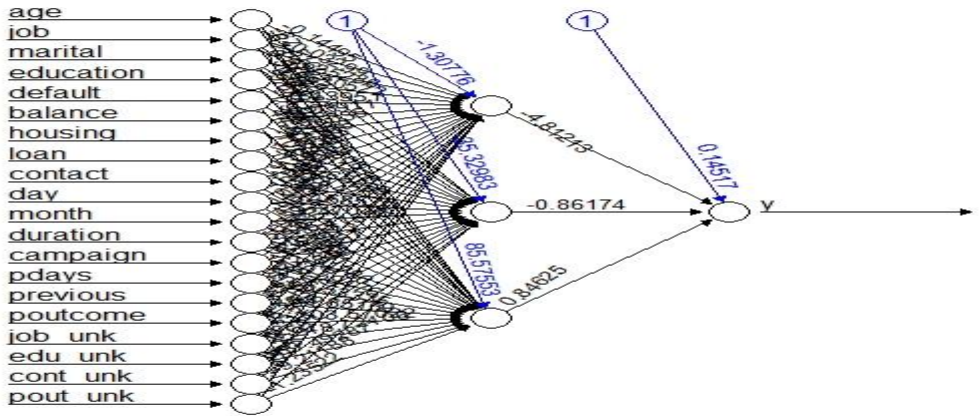
```
#Creating Confusion  the  training matrix
> ConMatNB_train = table(NB_training[,17],NB_Pred_Training)
> ConMatNB_train
   NB_Pred_Training
        0     1
  0 27740  2202
  1  1808  2159
>
> #Creating Confusion  the  test matrix
> ConMatNB = table(NB_test[,17],NB_Pred)
> ConMatNB
   NB_Pred
       0    1
  0 9225  755
  1  628  694
>
> #Accuracy test
```

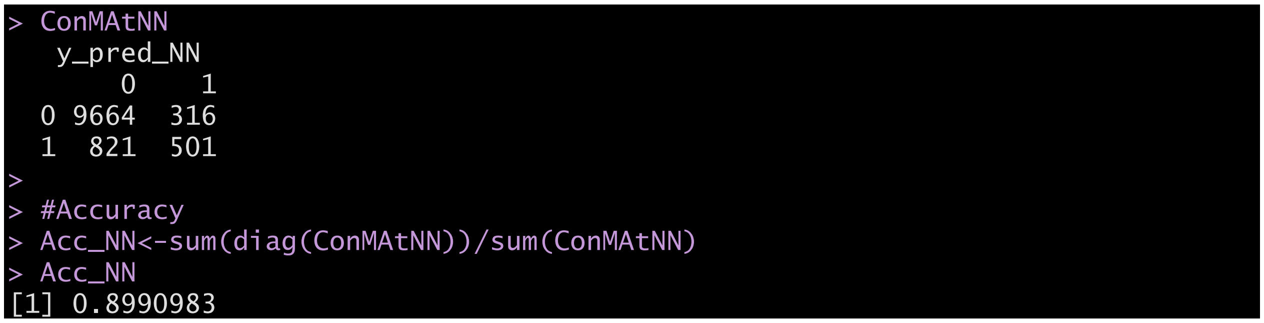


```
> Acc_NB<-sum(diag(ConMatNB))/sum(ConMatNB)
> Acc_NB
[1] 0.8776323
>
> #Accuracy train
> Acc_NB_Train<-sum(diag(ConMatNB_train))/sum(ConMatNB_train)
> Acc_NB_Train
[1] 0.8817423
```

For Naïve Bayes model we got training accuracy of 88.17% and model accuracy of 87.7% which is the least accurate model among all the models

7) **Neural Nets :**



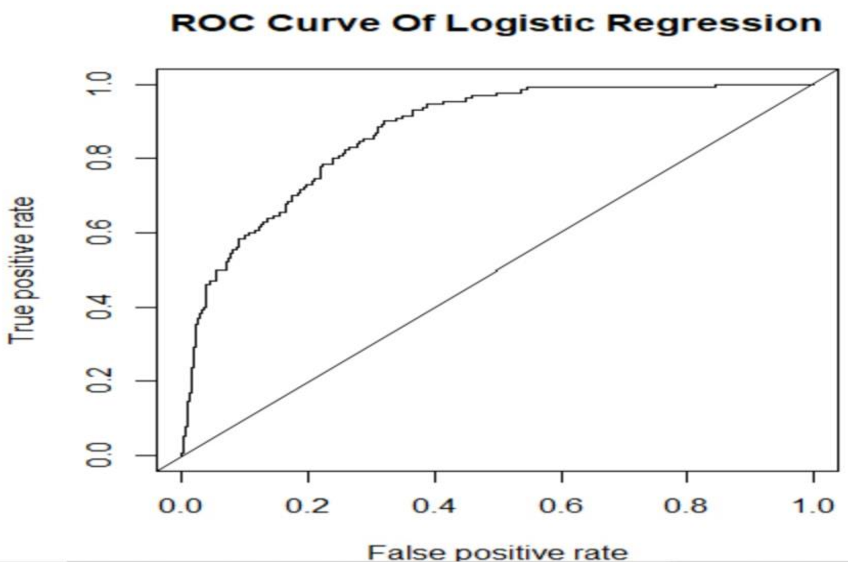


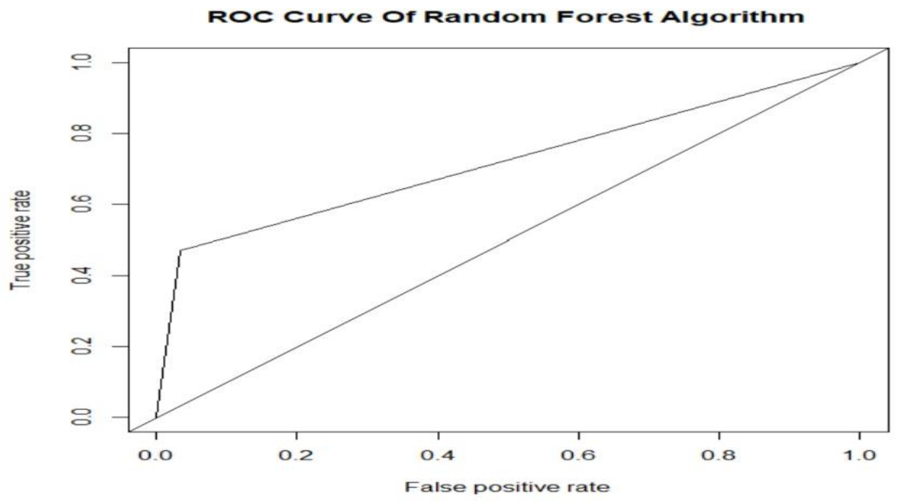
```
> ConMAtNN
   y_pred_NN
       0    1
  0 9664  316
  1  821  501
>
> #Accuracy
> Acc_NN<-sum(diag(ConMAtNN))/sum(ConMAtNN)
> Acc_NN
[1] 0.8990983
```

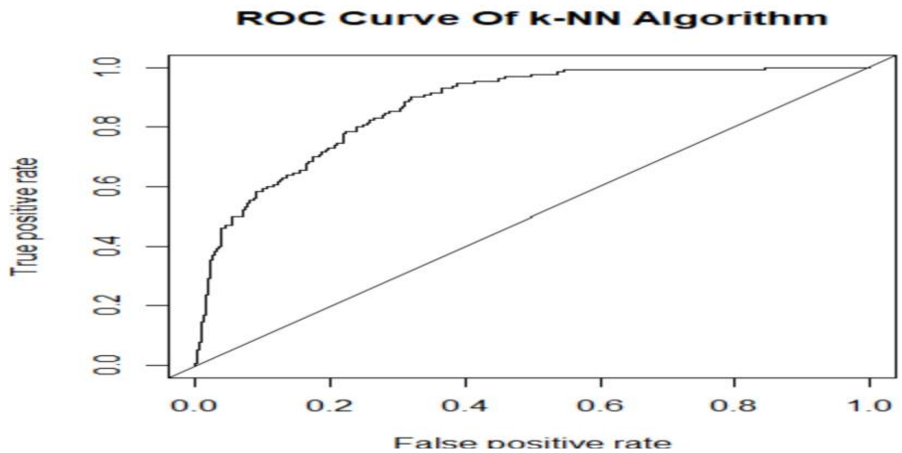
We got 89.9% accuracy for Neural Nets

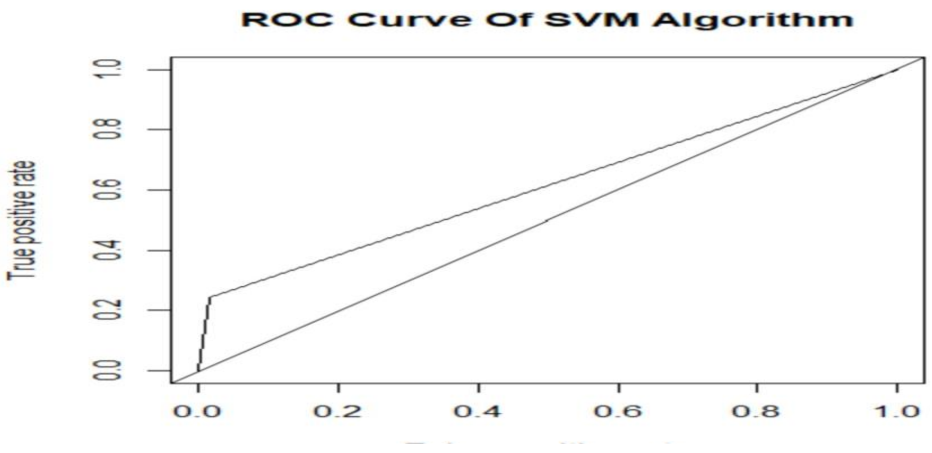
V. Performance Evaluation

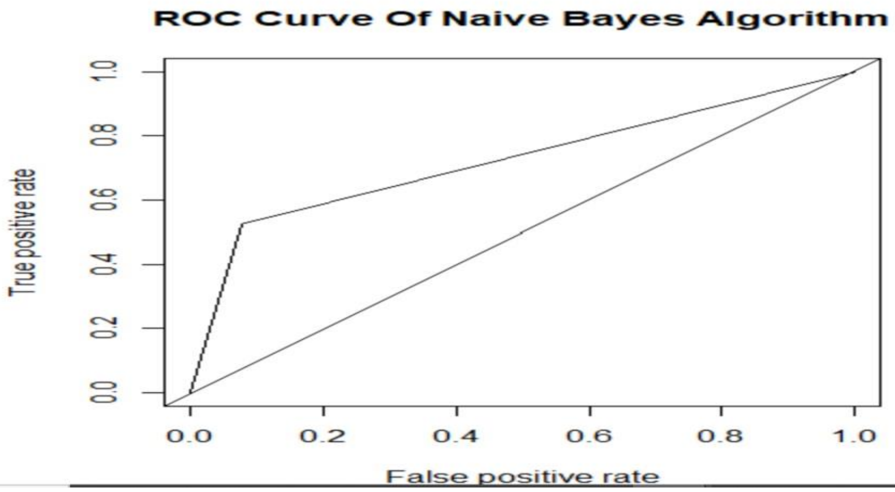
As we can see in results above, efficiency of different models are as follows:
1) For logistic regression, we got 89.93% accuracy
2) For K-NN, accuracy is 89.22% for k=5.
3) For Decision Tree we got 89.59% accuracy
4) For Random Forest, we got 90.7% accuracy y
5) For Support Vector Machine, we got 89.92% accuracy
6) For Naïve Bayes, we got 88.17% accuracy
7) For Neural Nets, we got 89.9% accuracy

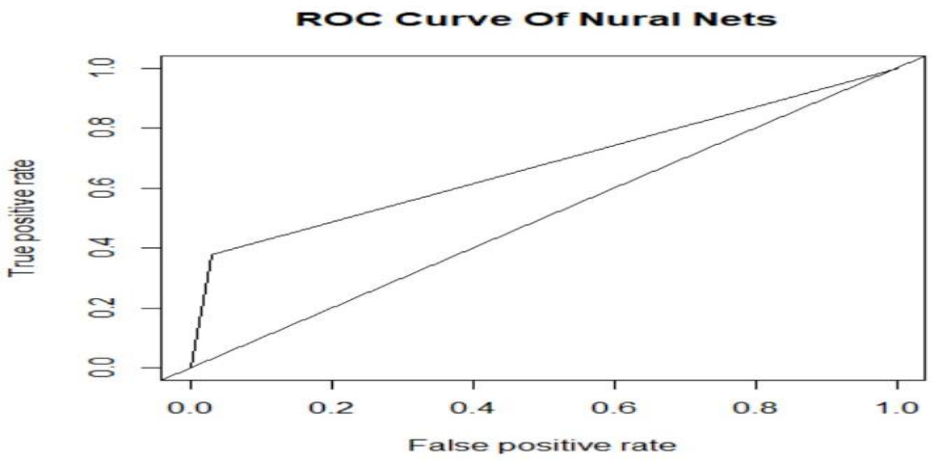

ROC Curve Of Logistic Regression
True positive rate
False positive rate
0.0
0.2
0.4
0.6
0.8
1.0


ROC Curve Of Random Forest Algorithm
True positive rate
False positive rate
0.0
0.2
0.4
0.6
0.8
1.0


ROC Curve Of k-NN Algorithm
True positive rate
False positive rate
0.0
0.2
0.4
0.6
0.8
1.0


ROC Curve Of SVM Algorithm
True positive rate
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0


ROC Curve Of Naive Bayes Algorithm
True positive rate
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0
False positive rate


ROC Curve Of Nural Nets
True positive rate
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0
False positive rate

VI. Discussion and Recommendation

From the case study conducted, the results are very good in terms of using machine learning in the banking sector to improve the marketing campaign. Almost all the attributes contributed towards the model, but duration of the call was the key feature. Among the all the models created Logistic regression and Random Forest were among the best models for the study. We recommend using Logistic regression model as its accuracy is almost same as Random forest and the Roc curve is much better than Random Forest.

Banks can improve their marketing campaign by focusing on target population and increase the rate of subscription to their product. The marketing team can identify potential client with high possibility of subscription using the models mentioned above. This will help reducing the investment of the banks in advertainment by avoiding the costumers who are unlikely to buy their products. Thus, produce better results from the marketing campaign.

VII. Summary

We followed the steps mentioned below:

1. Data Selection – Found the data set on UCI Machine learning repository
2. Data Exploration – Primary analysis of data
3. Data Cleaning – Removing missing values, Normalizing.
4. Data Visualization – Box plots, Correlation, Bar charts, Scatter plot
5. Data Splitting – Train 75%, Test 25%
6. Building Models – Logistic Regression, CART, K-NN, Naïve Bayes, SVM.
7. Performance evaluations – Confusion matrix and ROC Curves
8. Conclusion – Selection of best predicting model based on Accuracy and ROC curve – Logistic Regression

Appendix: R Code for use case study

```
#Loading Data
BankData<- read.csv('bank-full.csv', stringsAsFactors = F, header = T)
ExpData<- read.csv('bank-full.csv', stringsAsFactors = T, header = T)

#Checking the dimension
dim(BankData)

#Checking Structure of the data
str(BankData)
str(ExpData)

#summary of the dataset
summary(BankData)

#Checking the Categories
levels(ExpData$job)
levels(ExpData$marital)
levels(ExpData$education)
levels(ExpData$default)
levels(ExpData$housing)
levels(ExpData$loan)
levels(ExpData$contact)
levels(ExpData$y)

#Visualizing the data
#1)Exploring the output the Marketing Campaign
library(ggplot2)
ggplot(data=ExpData, aes(x=y, fill="Navi blue"))+
  geom_bar()+
  xlab("Outcome Y")+
  ylab("Count")+
  ggtitle("Exploring the Output of the Campaign")

#2)Boxplot of Jobs categories and Bank balance
ggplot(ExpData, aes(x=job,y=balance)) +
  geom_boxplot(size=1.2)+
  ylim(-1000,10000)+
  xlab("Job")+
  ylab("Balance")+
  ggtitle("Boxplot of Variours Job Categories and Balance in Account")

#3)Subscription based on Education categories
ggplot(ExpData, aes(x=education,fill=y)) +
  geom_bar(position = "dodge", colour="black")+
  xlab("EDUCATION")+
  ylab("COUNT")+ggtitle("Subscription Based on Eduction Categories")

#4)Duration of Call vs The Age of Teh Customer
#age vs Duration
ggplot(data=ExpData, aes(x=age,y=duration,colour=duration))+
  geom_point()+
  xlab("AGE")+
  ylab("DURATION (in sec)")+ggtitle(" Duration of Call vs The Age of The Customer")



#5)Distribustion of balances by education
ggplot(ExpData, aes(x=education,y=balance, colour=y)) +
  geom_boxplot(size=1.2)+
  ylim(-1000,10000)+
  xlab("Education")+
  ylab("Balance")+
  ggtitle("Distribution of Bank Balances By Eduaction")

#6) Scatter plt matrix
#install.packages("GGally")
library(GGally)
ggpairs(ExpData[, c(1,6,12,13,14,15)])


##### Data Preparation and Preprocessing

#Checking for missing values
sum(is.na(BankData))

#To check the collinearity among predictor variables
corrmat<-cor(ExpData)
#install.packages("corrplot")
library(corrplot)
corrplot(corrmat)
corrplot

# This code of chunk make additional columns for unknownn values
#and tranformsthe character data into numeic format

BankData$job_unk <- ifelse(BankData$job == "unknown", 1, 0)
BankData$job <- as.numeric(as.factor(BankData$job))
BankData$marital <- as.numeric(as.factor(BankData$marital))
BankData$edu_unk <- ifelse(BankData$education == "unknown", 1, 0)
BankData$education <- as.numeric(as.factor(BankData$education))
```

```
BankData$default<- ifelse(BankData$default == "yes", 1, 0)
BankData$housing <- ifelse(BankData$housing== "yes", 1, 0)
BankData$loan<- ifelse(BankData$loan== "yes", 1, 0)
BankData$month <- as.numeric(as.factor(BankData$month))
BankData$cont_unk <- ifelse(BankData$contact == "unknown", 1, 0)
BankData$contact <- as.numeric(as.factor(BankData$contact))
BankData$pout_unk <- ifelse(BankData$poutcome == "unknown", 1, 0)
BankData$poutcome <- as.numeric(as.factor(BankData$poutcome))
BankData$y <- ifelse(BankData$y== "yes", 1, 0)

str(BankData)

#splitting the data into traing and test data
library(caTools)
set.seed(100)
split=sample.split(BankData$y,SplitRatio = 0.75)
BankData_training = subset(BankData, split==T)
BankData_test = subset(BankData, split==F)


#feature scaling
BankData_training[,c(1,6,10,12,13:16)] = scale(BankData_training[,c(1,6,10,12,13:16)])
BankData_test[,c(1,6,10,12,13:16)] = scale(BankData_test[,c(1,6,10,12,13:16)] )
head(BankData_test)

#Data Mining Techniques and Implementation
#1)Logistic Regression

classifier = glm(formula = y~.,
           family =binomial,
           data= BankData_training)
summary(classifier)
#predicting the Test set results
Prob_pred = predict(classifier, type= 'response',
              newdata= BankData_test[-17])
#head(Prob_pred)
y_pred= ifelse(Prob_pred>0.5,1,0)

#Creating Confusion  the  matrix
ConMatLog = table(BankData_test[,17],y_pred)
ConMatLog

#Accuracy
Acc_Log1<-sum(diag(ConMatLog))/sum(ConMatLog)
Acc_Log1
```

```
library(gmodels)
CrossTable(y_pred,BankData_test[,17],prop.chisq = F)

library(ROCR)
ROCRpredLogi <- prediction(Prob_pred,BankData_test$y)
ROCRperfLogi <- performance(ROCRpred, 'tpr','fpr')
plot(ROCRperf, colorize = F, text.adj = c(-0.2,1.7))
abline(a=0,b=1)
title(main="ROC Curve Of Logistic Regression")

#2) KNN
library(class)
#Fitting K-NN to the training dataset and predicting  the Test set results
#k=1
y_pred_kNN1 = knn(train = BankData_training[,-17],
                test= BankData_test[,-17],
                cl=BankData_training[,17],
                k= 1)
ConMatKNN1 = table(BankData_test[,17],y_pred_kNN1)
#Accuracy
Acc_KNN1<-sum(diag(ConMatKNN1))/sum(ConMatKNN1)
Acc_KNN1

#k=2
y_pred_kNN2 = knn(train = BankData_training[,-17],
                test= BankData_test[,-17],
                cl=BankData_training[,17],
                k= 2)
ConMatKNN2 = table(BankData_test[,17],y_pred_kNN2)
#Accuracy
Acc_KNN2<-sum(diag(ConMatKNN2))/sum(ConMatKNN2)
Acc_KNN2

#k=3
y_pred_kNN3 = knn(train = BankData_training[,-17],
                test= BankData_test[,-17],
                cl=BankData_training[,17],
                k= 3)
ConMatKNN3 = table(BankData_test[,17],y_pred_kNN3)
#Accuracy
Acc_KNN3<-sum(diag(ConMatKNN3))/sum(ConMatKNN3)
Acc_KNN3

#k=4
y_pred_kNN4 = knn(train = BankData_training[,-17],
                test= BankData_test[,-17],
                cl=BankData_training[,17],
                k= 4)
ConMatKNN4 = table(BankData_test[,17],y_pred_kNN4)
ConMatKNN4
#Accuracy
Acc_KNN4<-sum(diag(ConMatKNN4))/sum(ConMatKNN4)
Acc_KNN4

#k=5
y_pred_kNN5 = knn(train = BankData_training[,-17],
                test= BankData_test[,-17],
                cl=BankData_training[,17],
                k= 5)
ConMatKNN5 = table(BankData_test[,17],y_pred_kNN5)
ConMatKNN5
#Accuracy
#Accuracy
Acc_KNN5<-sum(diag(ConMatKNN5))/sum(ConMatKNN5)
Acc_KNN5

#k=6
y_pred_kNN6 = knn(train = BankData_training[,-17],
                test= BankData_test[,-17],
                cl=BankData_training[,17],
                k= 6)
ConMatKNN6 = table(BankData_test[,17],y_pred_kNN6)
#Accuracy
Acc_KNN6<-sum(diag(ConMatKNN6))/sum(ConMatKNN6)
Acc_KNN6

#k=7
y_pred_kNN7 = knn(train = BankData_training[,-17],
                test= BankData_test[,-17],
                cl=BankData_training[,17],
                k= 7)
ConMatKNN7 = table(BankData_test[,17],y_pred_kNN7)
#Accuracy
Acc_KNN7<-sum(diag(ConMatKNN7))/sum(ConMatKNN7)
Acc_KNN7

#k=8
y_pred_kNN8 = knn(train = BankData_training[,-17],
                test= BankData_test[,-17],
                cl=BankData_training[,17],
                k= 8)
ConMatKNN8 = table(BankData_test[,17],y_pred_kNN8)
```

```
#Accuracy
Acc_KNN8<-sum(diag(ConMatKNN8))/sum(ConMatKNN8)
Acc_KNN8

#k=9
y_pred_kNN9 = knn(train = BankData_training[,-17],
                test= BankData_test[,-17],
                cl=BankData_training[,17],
                k= 9)
ConMatKNN9 = table(BankData_test[,17],y_pred_kNN9)
#Accuracy
Acc_KNN9<-sum(diag(ConMatKNN9))/sum(ConMatKNN9)
Acc_KNN9

#k=10
y_pred_kNN10 = knn(train = BankData_training[,-17],
                test= BankData_test[,-17],
                cl=BankData_training[,17],
                k= 10)
ConMatKNN10 = table(BankData_test[,17],y_pred_kNN10)
#Accuracy
Acc_KNN10<-sum(diag(ConMatKNN10))/sum(ConMatKNN10)
Acc_KNN10

ConMatKNN5
CrossTable(y_pred_kNN5,BankData_test[,17],prop.chisq = F)

#ROC knn
ROCRpredknn <- prediction(as.numeric(y_pred_kNN10),BankData_test$y)
ROCRperfknn <- performance(ROCRpredknn, 'tpr','fpr')
plot(ROCRperf, colorize = F, text.adj = c(-0.2,1.7))
abline(a=0,b=1)
title(main="ROC Curve Of k-NN Algorithm")

#3)Decision tree####

library(caTools)
set.seed(100)
split=sample.split(ExpData$y,SplitRatio = 0.75)
CTdata2_training = subset(ExpData, split==T)
CTdata2_test = subset(ExpData, split==F)

library(rpart )
library(rpart.plot)
CT_model2<- rpart(formula = y~.,
                data =CTdata2_training, method='class',maxdepth=5)
```

```
print(CT_model2)
rpart.plot(CT_model2,type = 1, extra = 1, split.font = 1, varlen = -10)
summary(CT_model2)
y_CT2<-predict(CT_model2,newdata= CTdata2_test[,-17], type ='class')

CMCT2= table(CTdata2_test[,17],y_CT2)
CMCT2

Acc_CT2<-sum(diag(CMCT2))/sum(CMCT2)
Acc_CT2
library(gmodels)
CrossTable(y_CT2,CTdata2_test[,17],prop.chisq = F)

#ROC--CT
ROCRpredCT1 <- prediction(as.numeric(y_CT2),BankData_test$y)
ROCRperfCT1<- performance(ROCRpred, 'tpr','fpr')
plot(ROCRperfCT1, colorize = F, text.adj = c(-0.2,1.7))
abline(a=0,b=1)
title(main="ROC Curve Of Decision tree Algorithm")

#4)SVM####################################
library(e1071)
SVM_Model1 = svm(formula = y~.,
            data= BankData_training,
            type='C-classification',
            kernel='linear')
summary(SVM_Model1)
#predicting the Test set results
y_SVM1 = predict(SVM_Model1,newdata= BankData_test[-17])

#making the matrix
ConMatSVM1 = table(BankData_test[,17],y_SVM1)
ConMatSVM1
#Accuracy
Acc_SVM<-sum(diag(ConMatSVM1))/sum(ConMatSVM1)
Acc_SVM

#ROC SVM
ROCRpredSVM <- prediction(as.numeric(y_SVM1),BankData_test$y)
ROCRperfSVM <- performance(ROCRpredSVM, 'tpr','fpr')
plot(ROCRperfSVM, colorize = F, text.adj = c(-0.2,1.7))
abline(a=0,b=1)
title(main="ROC Curve Of SVM Algorithm")

#5)RandomForest
#install.packages("randomForest")
```

```
library(caTools)
set.seed(100)
split=sample.split(ExpData$y,SplitRatio = 0.75)
RT_training = subset(ExpData, split==T)
RT_test = subset(ExpData, split==F)

library(randomForest)
RF <- randomForest(y ~ ., data = RT_training, ntree = 500,
                   mtry = 4, nodesize = 5, importance = TRUE)
RF

rf.pred <- predict(RF, RT_test[,-17])

#confusion Matrix
CMRT= table(RT_test[,17],rf.pred)
CMRT

Acc_RT<-sum(diag(CMRT))/sum(CMRT)
Acc_RT

ROCRpredRF <- prediction(as.numeric(rf.pred),RT_test[,17] )
ROCRperfRF <- performance(ROCRpredRF, 'tpr','fpr')
plot(ROCRperfRF, colorize = F, text.adj = c(-0.2,1.7))
abline(a=0,b=1)
title(main="ROC Curve Of Random Forest Algorithm")


#Naive Bayes

library(e1071)
#importing data
NB_data<-read.csv("bank-full.csv")

#encoding the target feature
NB_data$y<- factor(BankData$y, levels=c(0,1))

#Spliting data
library(caTools)
set.seed(100)
split=sample.split(NB_data$y,SplitRatio = 0.75)
NB_training = subset(NB_data, split==T)
NB_test = subset(NB_data, split==F)

NB1 =naiveBayes(x=NB_training[-17],
            y= NB_training[,17])
```

```
NB1

#summary(NB1)
#predicting the Test set results
NB_Pred_Training= predict(NB1,newdata= NB_training[,-17])
NB_Pred_Training

#predicting the Test set results
NB_Pred = predict(NB1,newdata= NB_test[,-17])
NB_Pred

#Creating Confusion  the  training matrix
ConMatNB_train = table(NB_training[,17],NB_Pred_Training)
ConMatNB_train

#Creating Confusion  the  test matrix
ConMatNB = table(NB_test[,17],NB_Pred)
ConMatNB

#Accuracy test
Acc_NB<-sum(diag(ConMatNB))/sum(ConMatNB)
Acc_NB

#Accuracy train
Acc_NB_Train<-sum(diag(ConMatNB_train))/sum(ConMatNB_train)
Acc_NB_Train

#ROC for NB
ROCRpredNB <- prediction(as.numeric(NB_Pred),NB_test[,17] )
ROCRperfNB <- performance(ROCRpredNB, 'tpr','fpr')
plot(ROCRperfNB, colorize = F, text.adj = c(-0.2,1.7))
abline(a=0,b=1)
title(main="ROC Curve Of Naive Bayes Algorithm")

#Nural nets
install.packages("neuralnet")
library(neuralnet)
nn<- neuralnet(y~.,
          data = BankData_training,
          linear.output = F,
          hidden = 3)
nn
plot(nn, rep="best")
predictnn <- neuralnet::compute(nn,BankData_test[,-17])
pre_nn <- predict(nn,newdata= BankData_test[,-17])
```

```
y_pred_NN= ifelse(pre_nn>0.5,1,0)
#confusion Matrix
ConMAtNN = table(BankData_test[,17],y_pred_NN)
ConMAtNN

#Accuracy
Acc_NN<-sum(diag(ConMAtNN))/sum(ConMAtNN)
Acc_NN
#ROC NN

ROCRpredNN <- prediction(as.numeric(y_pred_NN),BankData_test[,17] )
ROCRperfNN <- performance(ROCRpredNN, 'tpr','fpr')
plot(ROCRperfNN, colorize = F, text.adj = c(-0.2,1.7))
abline(a=0,b=1)
title(main="ROC Curve Of Nural Nets ")
```