Data Science Assignment

Sumeet Shivgand

Introduction

Data is randomly selected from one of the bank.Kate, who is a manager at financial institution need a help in assessing the credit worthiness of future potential.There are total 793 observations of past customer cases and 14 feature for each cases.Kate has 10 new customers, which she want to know whether she should give them a loan or not.

Question A

Explotary Data Analysis(EDA)

```
#1. Reading the data from sheet "Credit_Risk6_final.xlsx" file.
sheet1<- read_excel("D:\\CIT\\Data Science and Analytics\\Assignment\\Credit_Risk6_final.xlsx",
 sheet="Training_Data")

#Generate the dataframe for above generated sheet.
df <- data.frame(sheet1)
#View(df)
#str(df)
head(df)
```

	ID <dbl>	Checking.Acct <chr>	Credit.History <chr>	Loan.Reason <chr>	Savings.Acct <chr>	Employm… <chr>	Personal.Sta <chr>
1	1	No Acct	All Paid	Car New	Low	Medium	Single
2	2	0Balance	Current	Car New	Low	Short	Divorced
3	3	0Balance	Current	Car New	No Acct	Long	Divorced
4	4	0Balance	Current	Furniture	No Acct	Long	NA
5	5	No Acct	All Paid	Small Appliance	No Acct	Long	Single
6	6	Low	Current	Car New	MedLow	Very Short	Divorced

6 rows | 1-8 of 15 columns

```
#Checking for missing values.
sum(is.na(df))
```

```
## [1] 44
```

```
#copying original data
df1<- df
summary(df1)
```

```
##        ID        Checking.Acct      Credit.History     Loan.Reason
##  Min.   :  1.0   Length:780         Length:780         Length:780
##  1st Qu.:195.8   Class :character   Class :character   Class :character
##  Median :390.5   Mode  :character   Mode  :character   Mode  :character
##  Mean   :390.5
##  3rd Qu.:585.2
##  Max.   :780.0
##  Savings.Acct        Employment        Personal.Status
##  Length:780         Length:780         Length:780
##  Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character
##
##
##
##    Housing            Job.Type         Foreign.National
##  Length:780         Length:780         Length:780
##  Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character
##
##
##
##  Months.since.Checking.Acct.opened Residence.Time..In.current.district.
##  Min.   :  5.0                     Min.   :-2.000
##  1st Qu.: 13.0                     1st Qu.: 2.000
##  Median : 19.0                     Median : 3.000
##  Mean   : 23.2                     Mean   : 2.868
##  3rd Qu.: 29.5                     3rd Qu.: 4.000
##  Max.   :120.0                     Max.   :10.000
##       Age        Credit.Standing
##  Min.   :18.00   Length:780
##  1st Qu.:26.00   Class :character
##  Median :32.00   Mode  :character
##  Mean   :34.75
##  3rd Qu.:41.00
##  Max.   :99.00
```

```
#Remove the NA values
df1<- na.omit(df1)


#Removed the 'ID' Column as it is not necessary
df1<- subset(df1, select = Checking.Acct:Credit.Standing)
#View(df1)
sum(is.na(df1))
```

```
## [1] 0
```

```
#Converted all character variables to Factors

cols1 <- c("Checking.Acct","Credit.History","Loan.Reason","Savings.Acct","Employment",
           "Personal.Status","Housing","Job.Type","Foreign.National","Credit.Standing" )
df1 %<>%
  mutate_each_(funs(factor(.)),cols1)
```

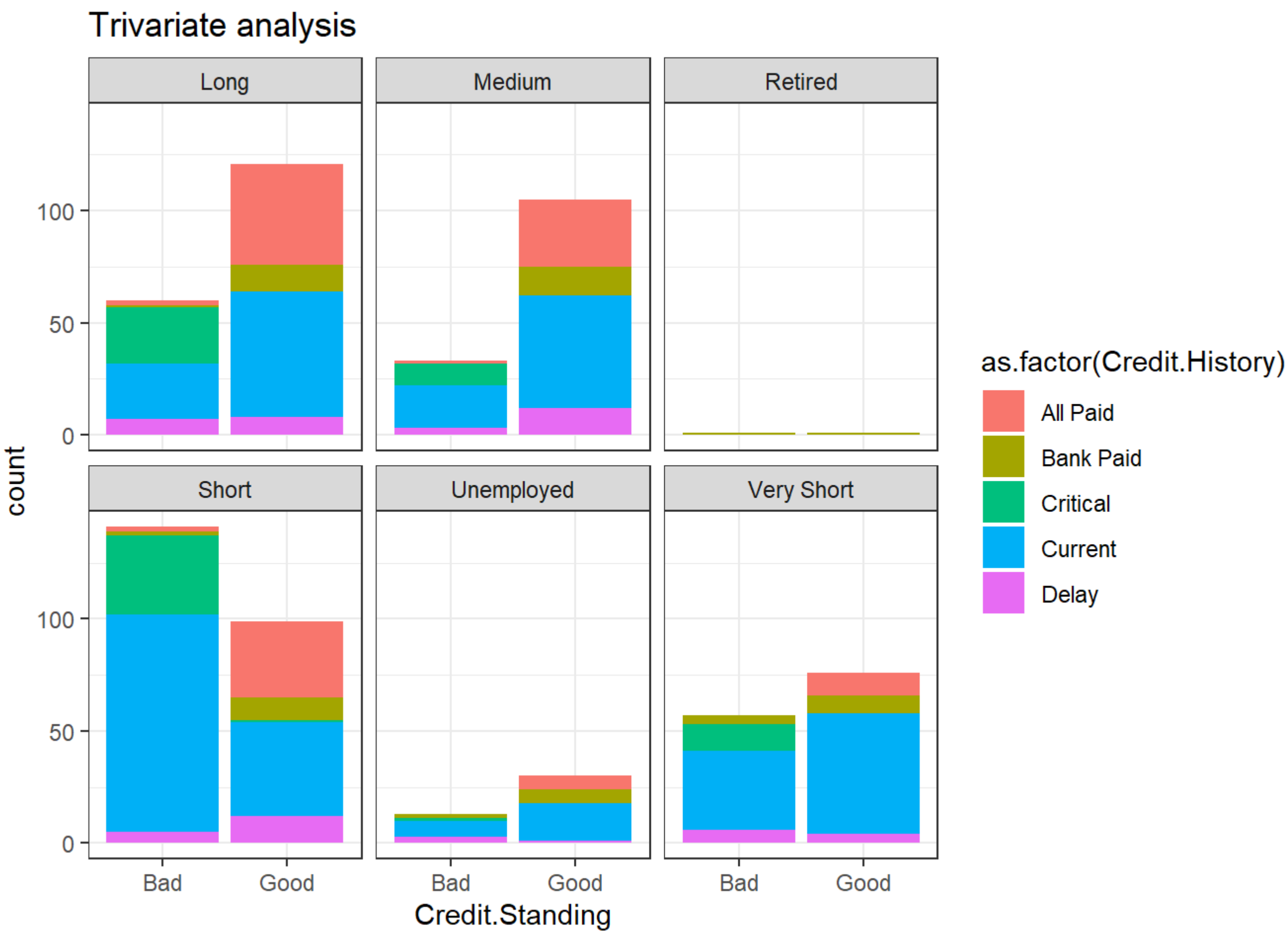
```
## Warning: mutate_each() is deprecated
## Please use mutate_if(), mutate_at(), or mutate_all() instead:
##
##   - To map `funs` over all variables, use mutate_all()
##   - To map `funs` over a selection of variables, use mutate_at()
## This warning is displayed once per session.
```

```
## Warning: funs() is soft deprecated as of dplyr 0.8.0
## Please use a list of either functions or lambdas:
##
##   # Simple named list:
##   list(mean = mean, median = median)
##
##   # Auto named with `tibble::lst()`:
##   tibble::lst(mean, median)
##
##   # Using lambdas
##   list(~ mean(., trim = .2), ~ median(., na.rm = TRUE))
## This warning is displayed once per session.
```

```
str(df1)
```

```
## 'data.frame':    737 obs. of  13 variables:
##  $ Checking.Acct                        : Factor w/ 4 levels "0Balance","High",..: 4 1 1 4 3 1
3 4 3 1 ...
##  $ Credit.History                       : Factor w/ 5 levels "All Paid","Bank Paid",..: 1 4 4
1 4 1 2 5 3 4 ...
##  $ Loan.Reason                          : Factor w/ 10 levels "Business","Car New",..: 2 2 2 1
0 2 2 4 3 5 5 ...
##  $ Savings.Acct                         : Factor w/ 5 levels "High","Low","MedHigh",..: 2 2 5
5 4 2 2 2 2 2 ...
##  $ Employment                           : Factor w/ 6 levels "Long","Medium",..: 2 4 1 1 6 1 2
4 6 4 ...
##  $ Personal.Status                      : Factor w/ 3 levels "Divorced","Married",..: 3 1 1 3
1 2 1 2 1 3 ...
##  $ Housing                              : Factor w/ 3 levels "Other","Own",..: 2 2 2 1 2 2 1 3
3 2 ...
##  $ Job.Type                             : Factor w/ 4 levels "Management","Skilled",..: 1 2 2
2 4 2 4 2 2 2 ...
##  $ Foreign.National                     : Factor w/ 2 levels "No","Yes": 1 1 1 2 1 2 1 2 2 1
...
##  $ Months.since.Checking.Acct.opened   : num  7 16 25 7 13 22 25 13 13 19 ...
##  $ Residence.Time..In.current.district.: num  3 2 2 4 2 3 4 4 3 3 ...
##  $ Age                                  : num  44 28 28 35 22 29 33 40 24 41 ...
##  $ Credit.Standing                      : Factor w/ 2 levels "Bad","Good": 2 1 1 2 2 2 2 2 1 1
...
```

```
#######Trivariate Analysis############
ggplot(data = df1)+ geom_bar(aes(x=Credit.Standing, fill=as.factor(Credit.History)))+
  ggtitle(label = "Trivariate analysis")+
  theme_bw()+facet_wrap(~Employment)
```



Comment:-

To build any machine learning model, we have to do some EDA(Exploratory Data Analysis) with dataset. EDA is one of the most crucial step in machine learning and it is necessary because it helps to gain familiarity with dataset as well as helps in identifying null or erroneous values and many others things. Majorly, EDA is executed by visualizing variables like doing Univariate visualization, Bivariate visualization, Multivariate Visualization and Dimensionality reduction. I am going to be working on dataset of loan customer cases i.e. Credit_Risk. Let's begin, I read the "Credit_Risk" excel file into a variable using 'read_excel()' function and then into a data frame. So, after analyzing the dataset we came to know that there are 780 observations and 13 features. There are total 44 rows which is having NA or Null values. So, there are two options to deal with NA values i.e. either remove NA values or Impute NA values using some method. I tried both methods and run the model but I find imputing NA values doesn't give better result so I remove the NA values from dataset. Also, I don't need 'ID' column so I removed that column from dataset. There are some categorical variables in data set which I have converted into factors. So, instead of converting one by one categorical variables to factor, I combines all categorical variables into factors at once using 'Magrittr' package. The main advantage of converting categorical variables into factor is that they can be used in statistical modelling. I have implemented trivariate(Multivariate) analysis using three variables such 'Credit History', 'Credit Standing' and 'Employment' to see if there is any unusual patterns with the data set. I found that Credit History having 'All paid' as well as 'Current' and Employment with 'Long' time has higher chances of getting the loan whereas Employment with 'Retired' has very less chances of getting the loan.

Question B

Build the Decision tree model and detailing its parameter

```
#Decision tree By 'Rpart'

#Copying the values of 'Credit Standing' variable of dataset to "Credit.Standing"
Credit.Standing <- df1$Credit.Standing

set.seed(850)
#Splitting the data into train and split
id<- sample(2,nrow(df1),prob= c(0.70,0.30),replace = TRUE)
df1_train<-df1[id==1,]
df1_test<-df1[id==2,]
#View(df1_train)

train1<-sample(1:nrow(df1_train))
#View(train1)
cr.test<- Credit.Standing[-train1]
#View(cr.test)

set.seed(850)
df1_model<-rpart(Credit.Standing~., data = df1_train)
#summary(df1_model)

#prp(df1_model, type = 3,extra = 4)

pred_df1<-predict(df1_model,newdata = df1_test,type = "class")
#pred_df1

table(pred_df1,df1_test$Credit.Standing)
```

```
##
## pred_df1 Bad Good
##     Bad   62   19
##     Good  39   97
```

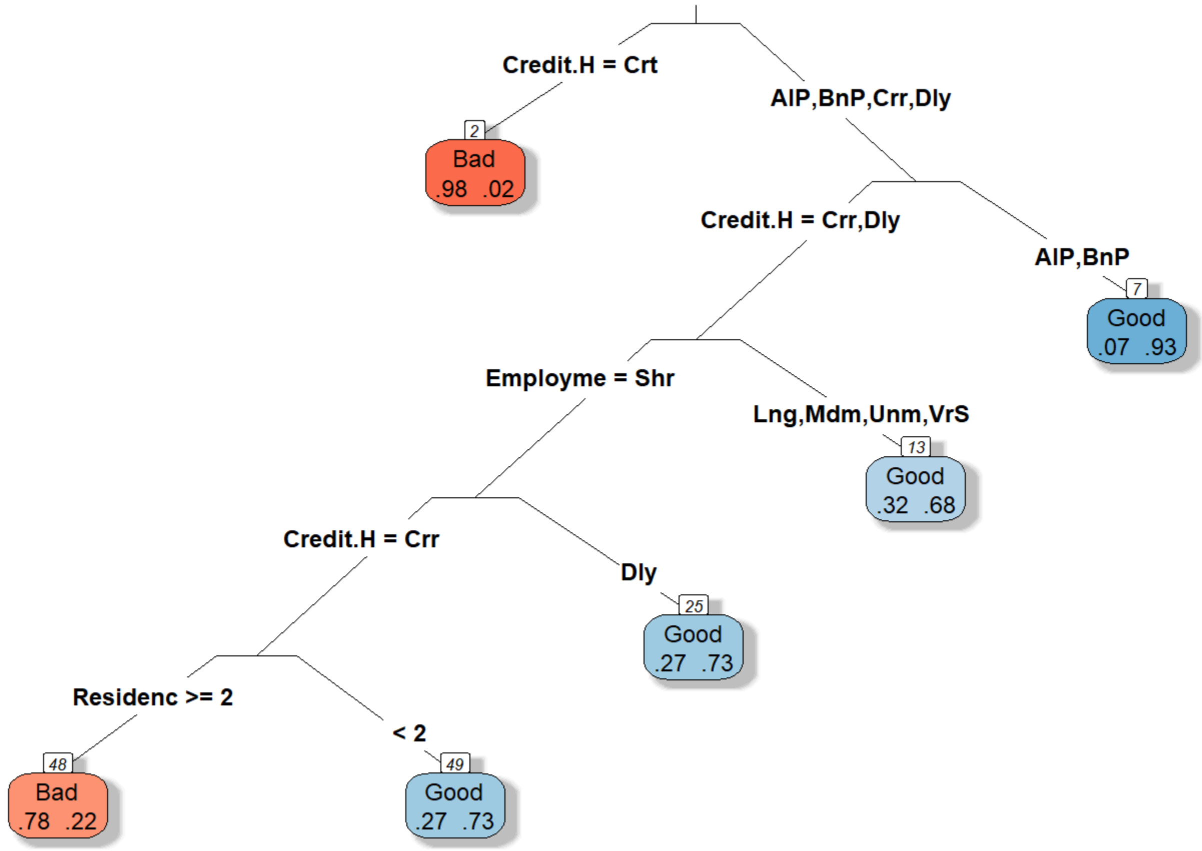
```
confusionMatrix(table(pred_df1,df1_test$Credit.Standing))
```

```
## Confusion Matrix and Statistics
##
##
## pred_df1 Bad Good
##     Bad   62   19
##     Good  39   97
##
##                Accuracy : 0.7327
##                  95% CI : (0.6686, 0.7904)
##     No Information Rate : 0.5346
##     P-Value [Acc > NIR] : 1.635e-09
##
##                   Kappa : 0.4559
##
##  Mcnemar's Test P-Value : 0.0126
##
##             Sensitivity : 0.6139
##             Specificity : 0.8362
##          Pos Pred Value : 0.7654
##          Neg Pred Value : 0.7132
##              Prevalence : 0.4654
##          Detection Rate : 0.2857
##    Detection Prevalence : 0.3733
##       Balanced Accuracy : 0.7250
##
##        'Positive' Class : Bad
##
```

```
##########Decision tree with cross validation#######

set.seed(850)
df1_modelCV<-rpart(Credit.Standing~., data = df1_train,control = rpart.control(cp = 0.02,xval =
10,maxdepth = 5))
#df1_modelCV
#summary(df1_modelCV)

prp(df1_modelCV,type = 3,extra = 4,box.palette="RdBu", shadow.col="gray", nn=TRUE)
```




```
pred_dfCV<-predict(df1_modelCV,newdata = df1_test,type = "class")

table(pred_dfCV,df1_test$Credit.Standing)
```

```
##
## pred_dfCV Bad Good
##      Bad   58   14
##      Good  43  102
```

```
confusionMatrix(table(pred_dfCV,df1_test$Credit.Standing))
```

```
## Confusion Matrix and Statistics
##
##
## pred_dfCV Bad Good
##      Bad   58   14
##      Good  43  102
##
##                Accuracy : 0.7373
##                  95% CI : (0.6735, 0.7946)
##     No Information Rate : 0.5346
##     P-Value [Acc > NIR] : 6.701e-10
##
##                   Kappa : 0.4621
##
##  Mcnemar's Test P-Value : 0.0002083
##
##             Sensitivity : 0.5743
##             Specificity : 0.8793
##          Pos Pred Value : 0.8056
##          Neg Pred Value : 0.7034
##              Prevalence : 0.4654
##          Detection Rate : 0.2673
##    Detection Prevalence : 0.3318
##       Balanced Accuracy : 0.7268
##
##        'Positive' Class : Bad
##
```

Comment:-

Decision tree is the one of the most popular technique in machine algorithm. It is a non-parametric supervised learning algorithm used for solving classification and regression problems. It works for both categorical and continuous input and output variables. Further, I go through the two concepts like entropy and information gain. Entropy is used to measure the randomness or uncertainty of a sample. Entropy is zero if sample is completely homogenous and it is one if sample is equally divided. Information gain is nothing but decrease in entropy. Decision tree is build on the basis of attribute that gives the highest information gain. I had implemented the decision tree using ‘Rpart’. I tried building two the decision tree model, one using cross validation and another without cross validation. I got little a bit improvement in my accuracy using cross validation(cp value = 0.02 and xval = 10). Using cross validation, I got 73.73 accuracy and 73.27 accuracy without using cross validation. So, I am going to consider the decision tree model with higher accuracy.

Question C

Predicting results for the scoring set using decision tree. Choose 5 different potential loan clients. Explaining Kate in plain english how decision tree works and how accuracy/probabilities calculated by decision tree for good/bad loan.

```
sheet2<- read_excel("D:\\CIT\\Data Science and Analytics\\Assignment\\Credit_Risk6_final.xlsx",
 sheet="Scoring_Data")

#Generate the dataframe for above generated sheet.
Scoring_data <- data.frame(sheet2)
#View(Scoring_data)

# Variables is in mixed data type like character and numeric
str(Scoring_data)
```

```
## 'data.frame':    13 obs. of  13 variables:
##  $ ID                           : num  781 782 783 784 785 786 787 788 789 790 ...
##  $ Checking.Acct                : chr  "No Acct" "Low" "No Acct" "High" ...
##  $ Credit.History               : chr  "All Paid" "Current" "Current" "Current" ...
##  $ Loan.Reason                  : chr  "Car New" "Small Appliance" "Small Appliance" "Bus
iness" ...
##  $ Savings.Acct                 : chr  "MedHigh" "Low" "Low" "Low" ...
##  $ Employment                   : chr  "Short" "Medium" "Very Short" "Medium" ...
##  $ Personal.Status              : chr  "Single" "Single" "Divorced" "Single" ...
##  $ Housing                      : chr  "Rent" "Rent" "Own" "Rent" ...
##  $ Job.Type                     : chr  "Unskilled" "Skilled" "Skilled" "Skilled" ...
##  $ Foreign.National             : chr  "No" "No" "No" "Yes" ...
##  $ Months.since.Checking.Acct.opened: num  11 37 13 16 9 49 37 12 19 16 ...
##  $ Residence.Time               : num  2 4 2 4 2 4 2 1 4 4 ...
##  $ Age                          : num  39 23 28 25 43 39 30 19 38 32 ...
```

```
#Rename the column name
setnames(Scoring_data, "Residence.Time", "Residence.Time..In.current.district.")

#Adding extra columns to match the arguments in train data

Scoring_test<- subset(Scoring_data, select = Checking.Acct:Age)

cols <- c("Checking.Acct","Credit.History","Loan.Reason","Savings.Acct","Employment",
        "Personal.Status","Housing","Job.Type","Foreign.National")
Scoring_test %<>%
  mutate_each_(funs(factor(.)),cols)

#Variables converted into factors and numeric
str(Scoring_test)
```

```
## 'data.frame':    13 obs. of  12 variables:
##  $ Checking.Acct                      : Factor w/ 4 levels "0Balance","High",..: 4 3 4 2 3 3 
2 3 4 4 ...
##  $ Credit.History                     : Factor w/ 3 levels "All Paid","Critical",..: 1 3 3 3 
3 3 3 1 3 3 ...
##  $ Loan.Reason                        : Factor w/ 6 levels "Business","Car New",..: 2 6 6 1 
6 5 6 4 6 4 ...
##  $ Savings.Acct                       : Factor w/ 4 levels "High","Low","MedHigh",..: 3 2 2 
2 2 4 2 1 2 4 ...
##  $ Employment                         : Factor w/ 5 levels "Long","Medium",..: 3 2 5 2 2 4 1 
5 5 2 ...
##  $ Personal.Status                    : Factor w/ 3 levels "Divorced","Married",..: 3 3 1 3 
1 3 3 1 2 1 ...
##  $ Housing                            : Factor w/ 3 levels "Other","Own",..: 3 3 2 3 2 1 2 2 
2 3 ...
##  $ Job.Type                           : Factor w/ 4 levels "Management","Skilled",..: 4 2 2 
2 4 3 2 2 2 4 ...
##  $ Foreign.National                   : Factor w/ 2 levels "No","Yes": 1 1 1 2 2 2 2 2 2 2 
...
##  $ Months.since.Checking.Acct.opened  : num  11 37 13 16 9 49 37 12 19 16 ...
##  $ Residence.Time..In.current.district.: num  2 4 2 4 2 4 2 1 4 4 ...
##  $ Age                                : num  39 23 28 25 43 39 30 19 38 32 ...
```

```
#Copying original data into another variable df_training
df_training <- df1

set.seed(850)
#colnames(Scoring_test)

df1_pmodel<-rpart(Credit.Standing~., data = df_training)
#df1_pmodel
#summary(df1_pmodel)

pred_scoring<-predict(df1_pmodel,newdata = Scoring_test,type = "class")
#pred_scoring

#Copying Scoring data into another variable Scoring_test
Scoring_test1<- Scoring_test

#Cbind is used to combine to dataframe with the same number of rows
Scoring_test1<-cbind(Scoring_test1,pred_scoring)

setnames(Scoring_test1, "pred_scoring", "Credit.Standing")
#View(Scoring_test1)
```

Comment:-

Here, I am going to predict the results for scoring set using decision tree. I am using credit dataset as training data and scoring set as test data. Decision tree works on several steps like splitting, pruning and tree selection. In splitting, dataset is divided into subsets and then at significant variable splits are forms. Then, pruning helps in reducing the size of the tree by turning some of the branch nodes into leaf nodes and then removing the leaf node from the original branch. Lastly, in the tree selection, we need to find the smallest tree that fits the data. This is the tree that yields the lowest cross-validated error. Confusion matrix is used to calculate the accuracy or probabilities of decision tree. The confusion matrix is nothing but a table that is often used to describe the performance of a classification model on a set of test data for which the true values are known. The number of correct and incorrect predictions are summarized with count values and broken down by each class. Confusion matrix is not only gives the insight of the errors which is being made by classifier but also gives the types of errors that are being made.After predicting the result on scoring data we get an output like out off 13 clients there are 9 clients with 'Good' and 4 clients with 'Bad'. So, Kate can choose any of 5 potential loan clients of her choice and decide whether to give loan or not.

Question D

Improve your model using 2 other approaches, e.g. ensemble technique, boosting or a different model. Comment on your results and analyse why your model is giving better/worse results.

```
##Random Forest##

set.seed(850)
rf<- randomForest(Credit.Standing~., data = df1_train,ntree = 540,mtry = 3,importance = TRUE,pro
ximity=TRUE)

print(rf) #75.1(540)
```

```
##
## Call:
##  randomForest(formula = Credit.Standing ~ ., data = df1_train,      ntree = 540, mtry = 3, im
portance = TRUE, proximity = TRUE)
##                Type of random forest: classification
##                      Number of trees: 540
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 24.81%
## Confusion matrix:
##      Bad Good class.error
## Bad  133   71   0.3480392
## Good  58  258   0.1835443
```

```
#attributes(rf)

#Prediction and confusion matrix for train data
p1<-predict(rf,df1_train)

confusionMatrix(p1,df1_train$Credit.Standing)
```

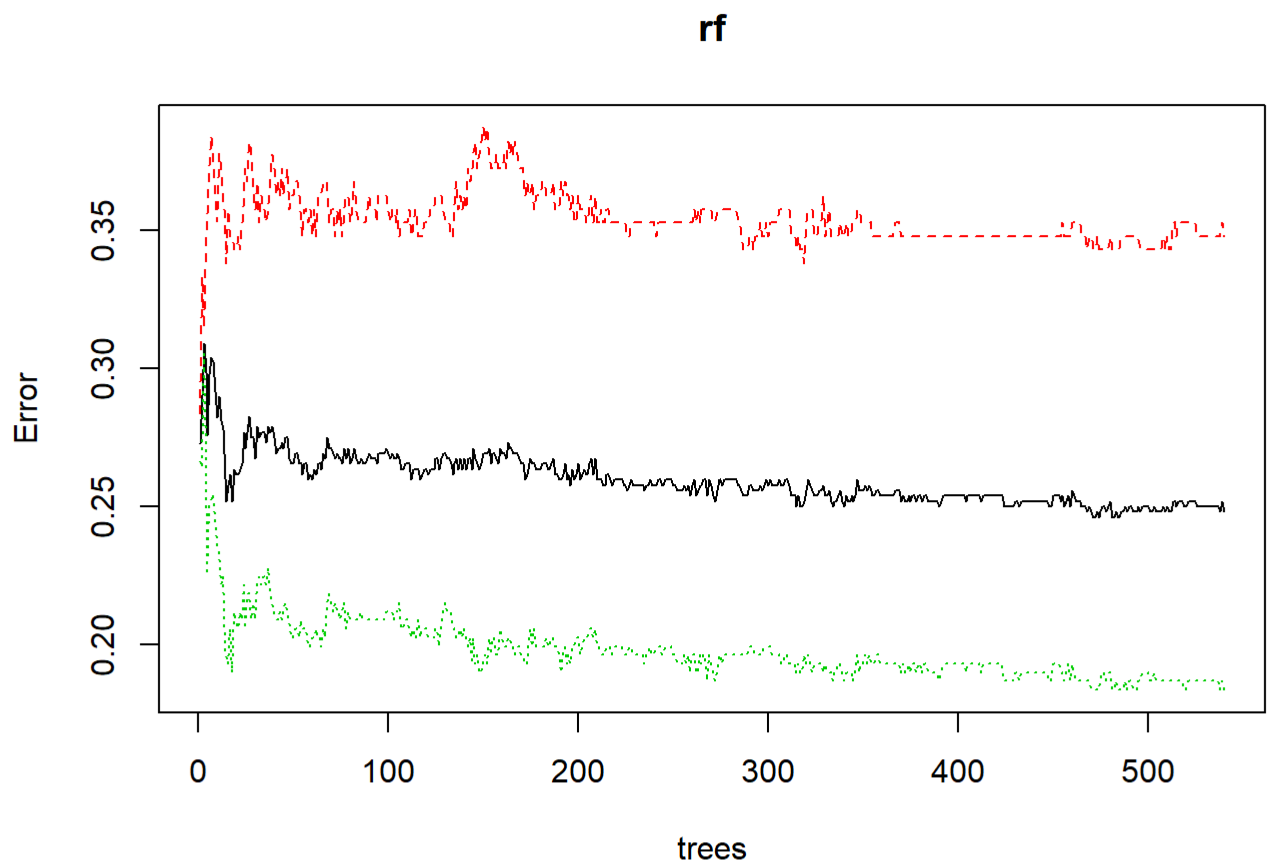
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Bad Good
##       Bad  195   16
##       Good   9  300
##
##                Accuracy : 0.9519
##                  95% CI : (0.9298, 0.9686)
##     No Information Rate : 0.6077
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8998
##
##  Mcnemar's Test P-Value : 0.2301
##
##             Sensitivity : 0.9559
##             Specificity : 0.9494
##          Pos Pred Value : 0.9242
##          Neg Pred Value : 0.9709
##              Prevalence : 0.3923
##          Detection Rate : 0.3750
##    Detection Prevalence : 0.4058
##       Balanced Accuracy : 0.9526
##
##        'Positive' Class : Bad
##
```

```
# there is mismatch between OOB[Out Of Bag](24.81) and Accuracy(95.19)

#Prediction and confusion matrix for test data
p2<-predict(rf,df1_test)
confusionMatrix(p2,df1_test$Credit.Standing)  #75.58
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction Bad Good
##       Bad   73   25
##       Good  28   91
##                                           
##                Accuracy : 0.7558          
##                  95% CI : (0.693, 0.8114)
##     No Information Rate : 0.5346          
##     P-Value [Acc > NIR] : 1.494e-11       
##                                           
##                   Kappa : 0.5082          
##                                           
##  Mcnemar's Test P-Value : 0.7835          
##                                           
##             Sensitivity : 0.7228          
##             Specificity : 0.7845          
##          Pos Pred Value : 0.7449          
##          Neg Pred Value : 0.7647          
##              Prevalence : 0.4654          
##          Detection Rate : 0.3364          
##    Detection Prevalence : 0.4516          
##       Balanced Accuracy : 0.7536          
##                                           
##        'Positive' Class : Bad             
## 
```

```
#Error rate of random Forest
plot(rf)
```

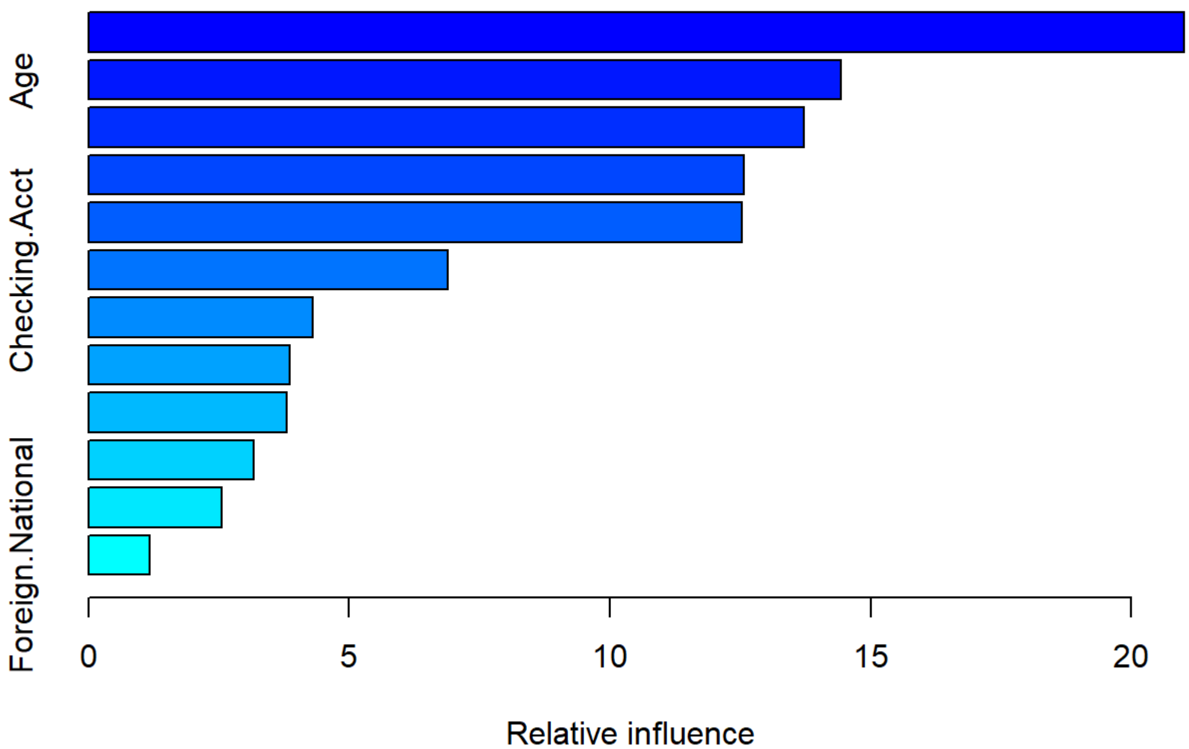



```
#Gives the importance variables
#varImpPlot(rf,sort = TRUE,n.var = 10,main = "Top 10 Important Variables")
#importance(rf)

##Gradient Boosting-GBM##

df1$Credit.Standing1 <- as.numeric(df1$Credit.Standing)
#df1$Credit.Standing1
#df1$Credit.Standing
df1$Credit.Standing1 <- df1$Credit.Standing1-1
#df1$Credit.Standing1
# 1 is for Yes and 0 is for No Credit standing

set.seed(850)
gbm.boost.df1=gbm(Credit.Standing1~.-Credit.Standing,data=df1[train1,],distribution="bernoulli",
n.trees=5000, interaction.depth=4)
summary(gbm.boost.df1)
```



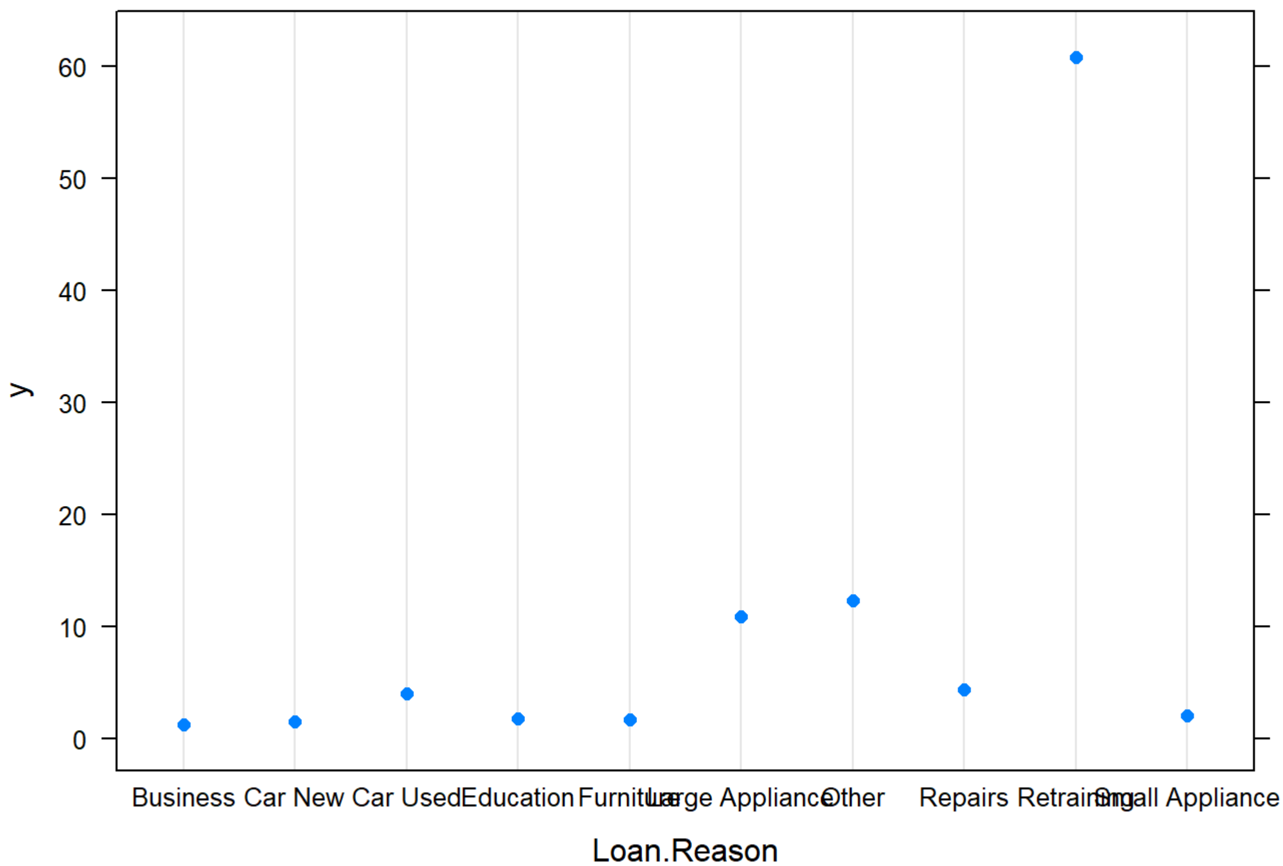
	var	re
	<fctr>	<c
Loan.Reason	Loan.Reason	21.008
Age	Age	14.426
Months.since.Checking.Acct.opened	Months.since.Checking.Acct.opened	13.712
Credit.History	Credit.History	12.562
Employment	Employment	12.538
Checking.Acct	Checking.Acct	6.895
Residence.Time..In.current.district.	Residence.Time..In.current.district.	4.296
Savings.Acct	Savings.Acct	3.851
Job.Type	Job.Type	3.800
Personal.Status	Personal.Status	3.167

1-10 of 12 rows Previous **1** 2 Next

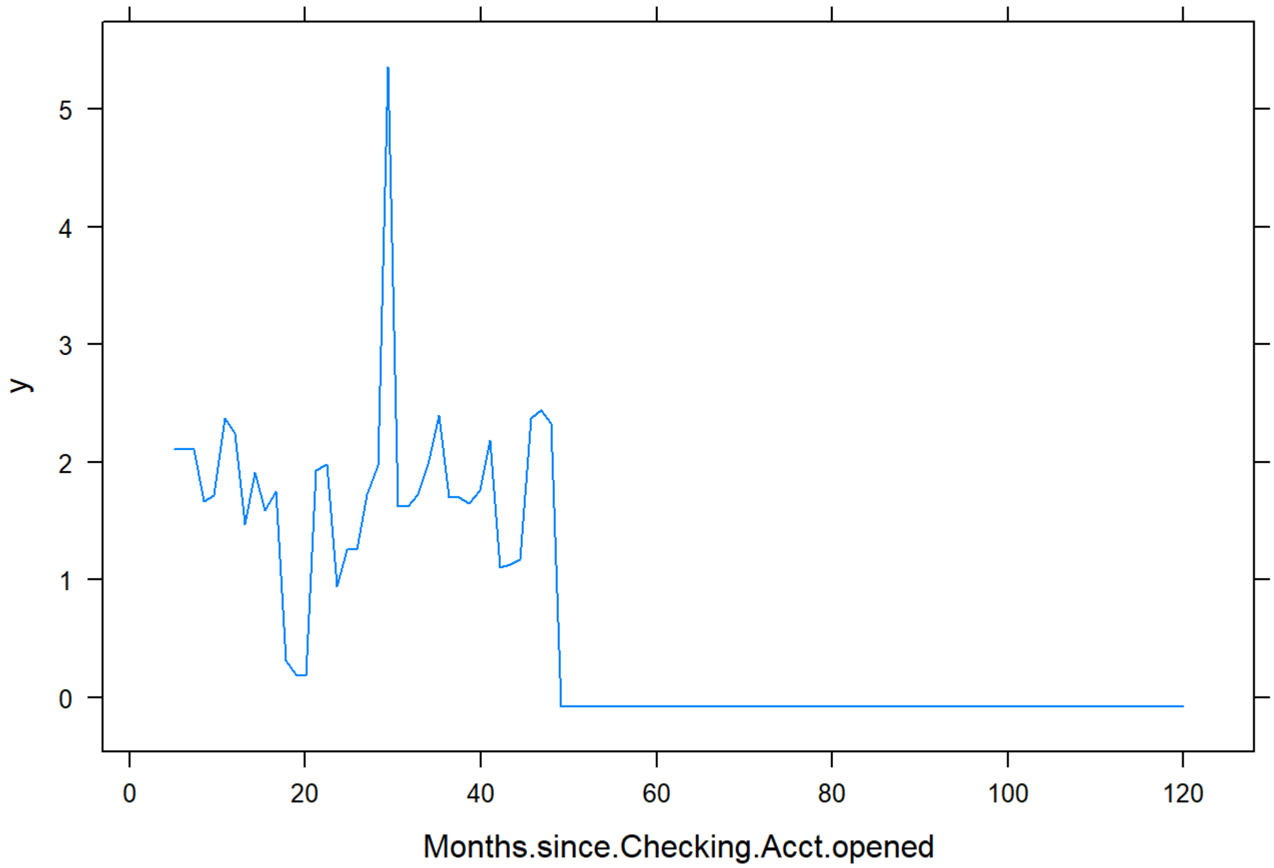
```
gbm.boost.df1
```

```
## gbm(formula = Credit.Standing1 ~ . - Credit.Standing, distribution = "bernoulli",
##     data = df1[train1, ], n.trees = 5000, interaction.depth = 4)
## A gradient boosted model with bernoulli loss function.
## 5000 iterations were performed.
## There were 12 predictors of which 12 had non-zero influence.
```

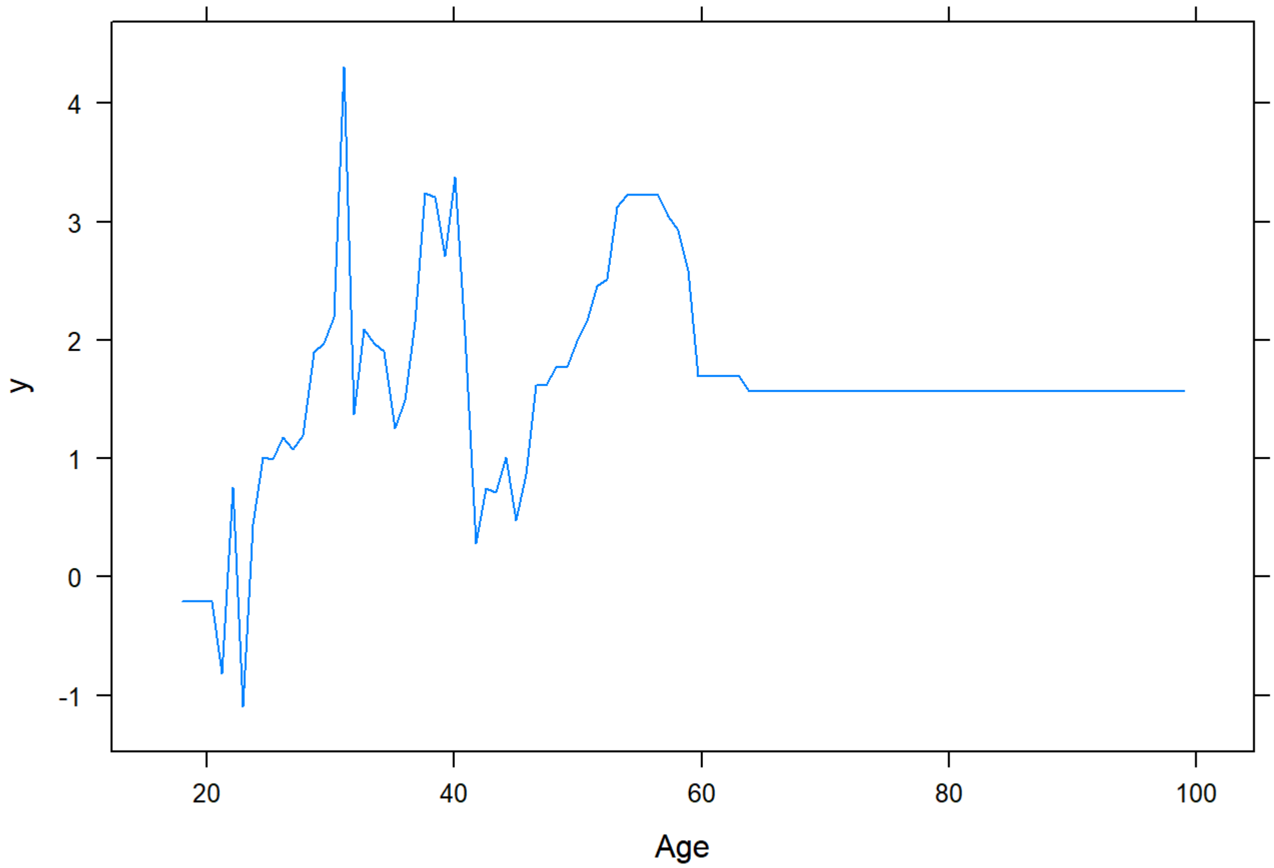
```
#par(mfrow=c(1,2))
#Plot for important variables
plot(gbm.boost.df1,i="Loan.Reason")
```



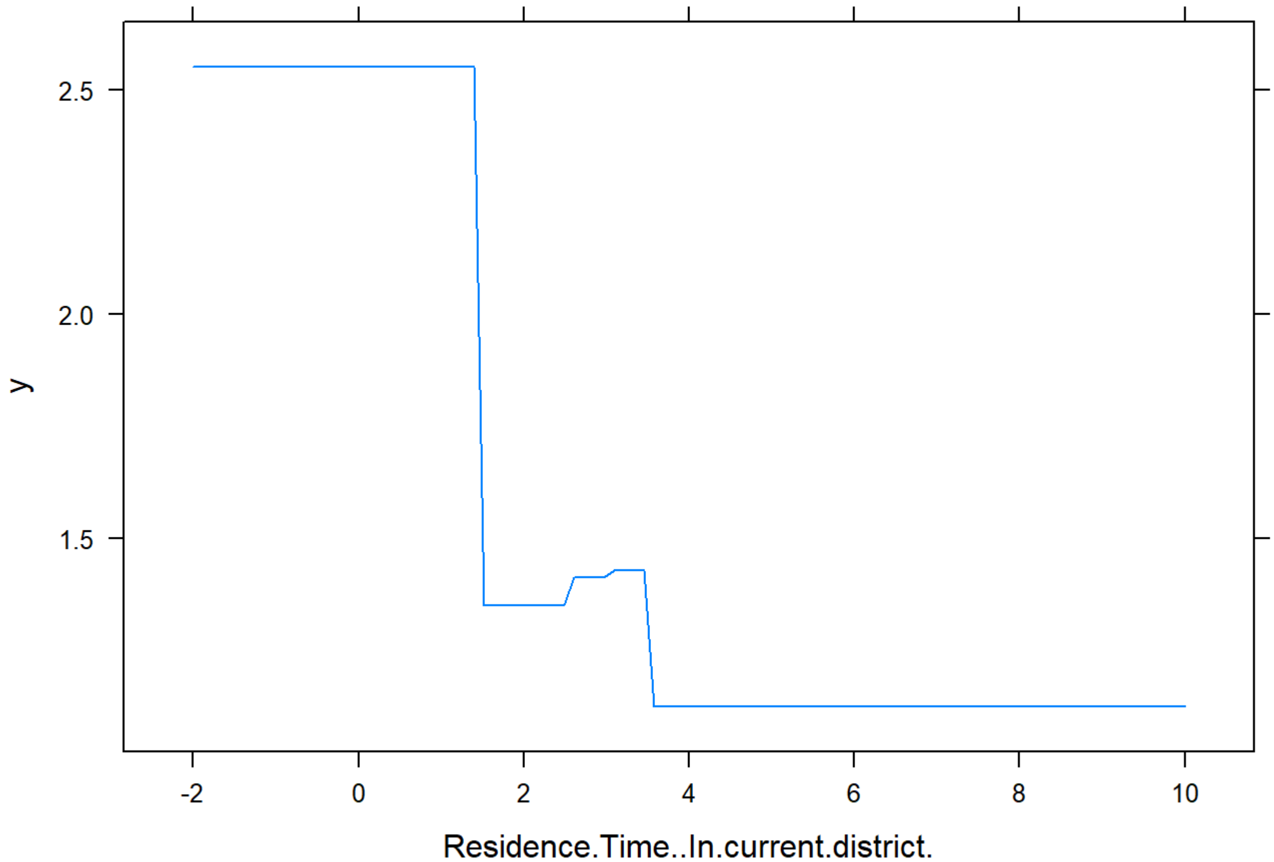

```
plot(gbm.boost.df1,i= "Months.since.Checking.Acct.opened")
```



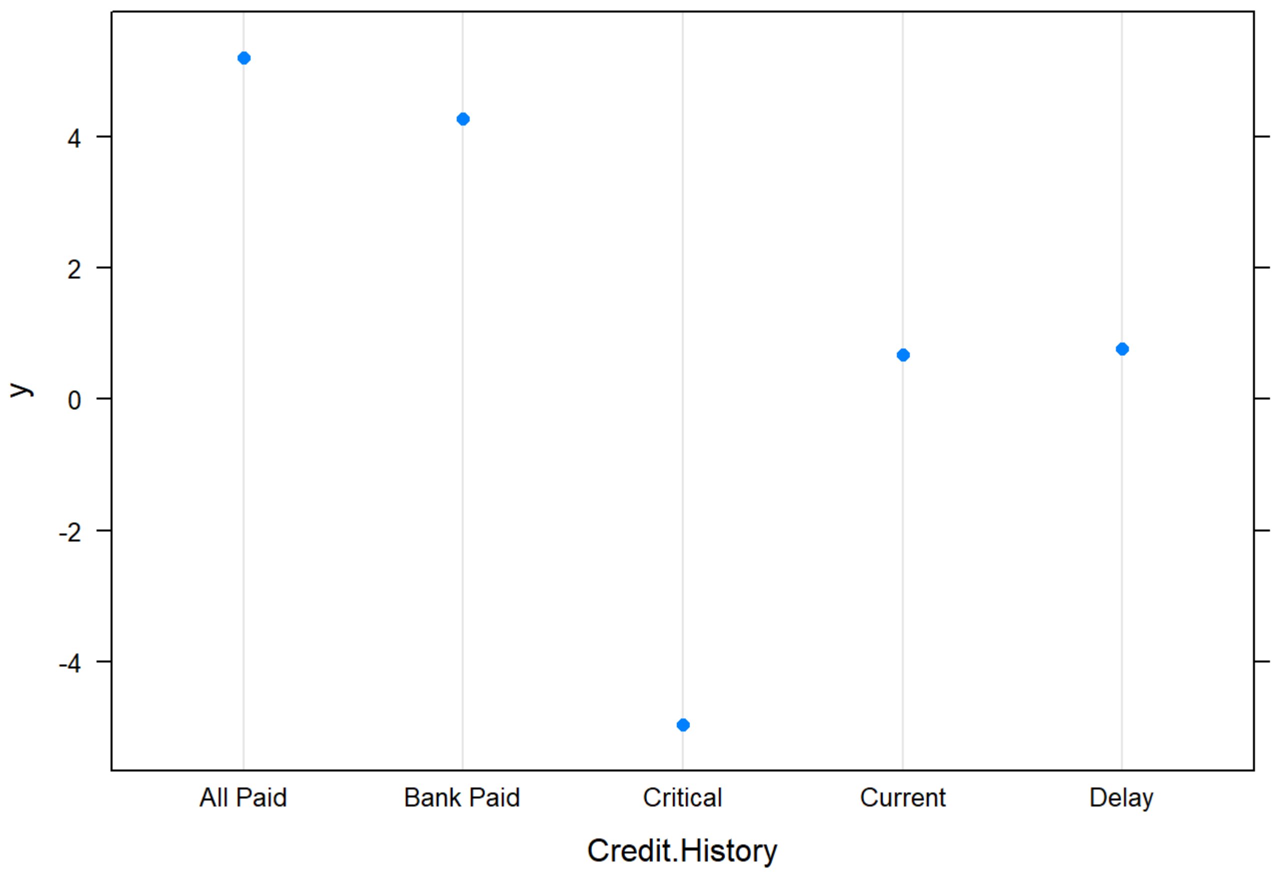

```
plot(gbm.boost.df1,i="Age")
```




```
plot(gbm.boost.df1,i="Residence.Time..In.current.district.")
```

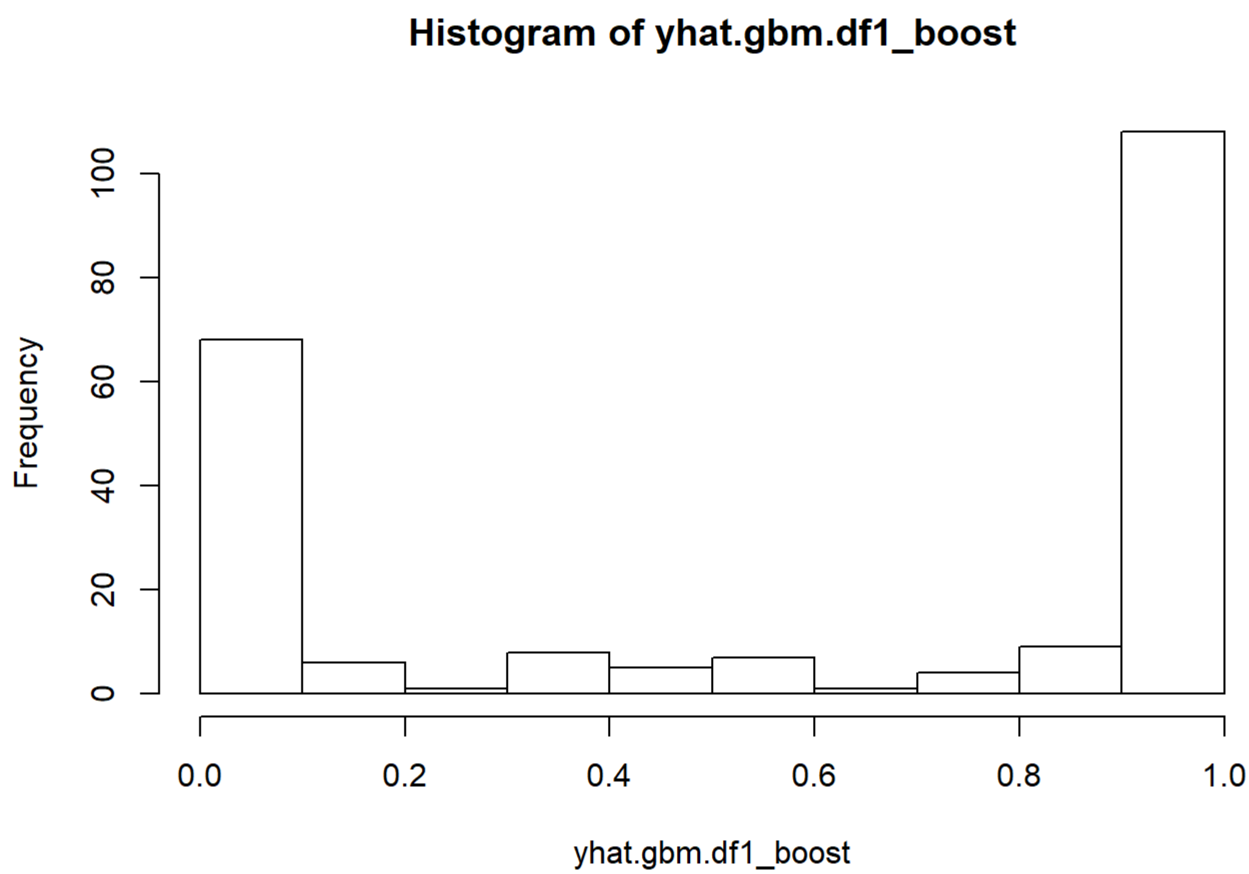



```
plot(gbm.boost.df1,i="Credit.History")
```




```
yhat.gbm.df1_boost=predict(gbm.boost.df1,newdata=df1[-train1,],n.trees=5000,type = "response")

hist(yhat.gbm.df1_boost)
```




```
predict_class1 <- ifelse(yhat.gbm.df1_boost<0.5,"Bad","Good")

table(predict_class1,df1[-train1,"Credit.Standing1"])
```

```
##
## predict_class1  0  1
##           Bad  65 23
##           Good 33 96
```

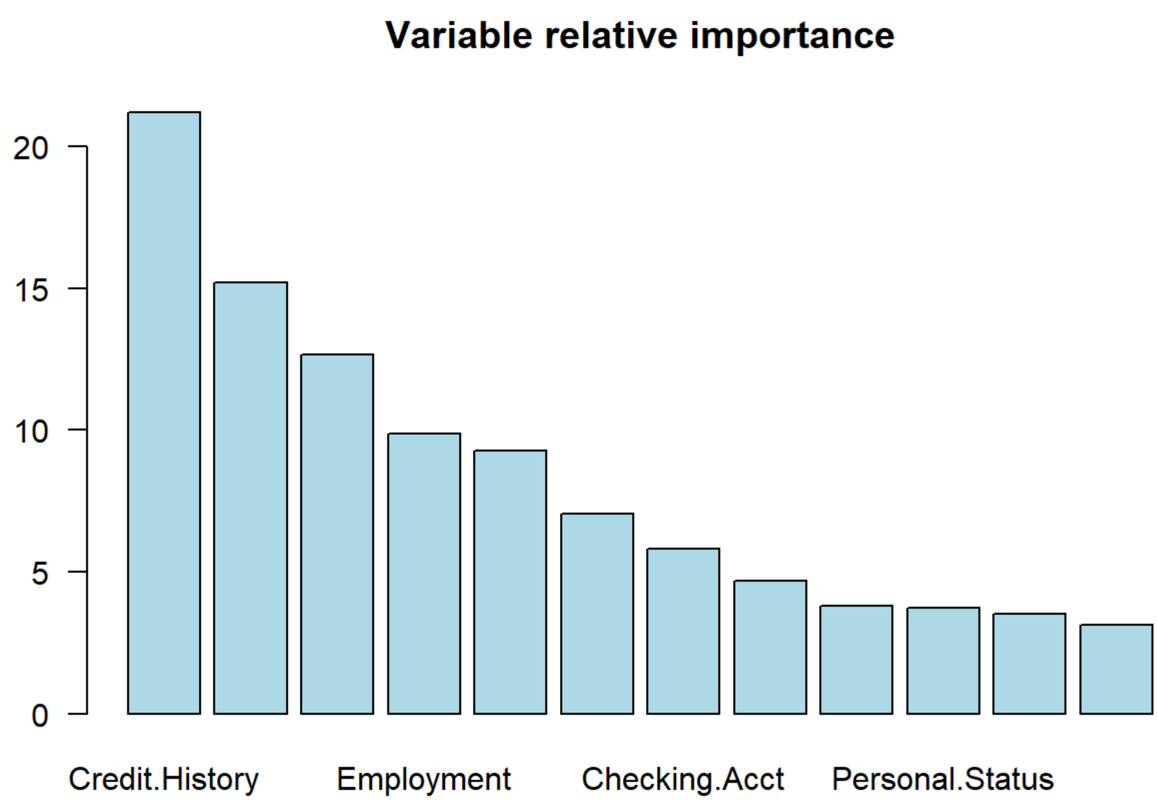
```
#(65+96)/217= 74.19


##Ada-Boost##

df1.adabag <- boosting(Credit.Standing~.-Credit.Standing1, data = df1, boos = TRUE, mfinal=100)
sort(df1.adabag$importance)
```

```
##                               Job.Type                         Housing
##                               3.126476                        3.524676
##                        Personal.Status                Foreign.National
##                               3.746266                        3.815147
## Residence.Time..In.current.district.                    Checking.Acct
##                               4.687985                        5.814018
##                           Savings.Acct  Months.since.Checking.Acct.opened
##                               7.057789                        9.261889
##                             Employment                     Loan.Reason
##                               9.882416                       12.662348
##                                    Age                  Credit.History
##                              15.214993                       21.205997
```

```
importanceplot(df1.adabag)
```



```
#set.seed(850)
#df1.adabag_cv <- boosting.cv(Credit.Standing~.-Credit.Standing1, data = df1[train1,],v=10, boos
= TRUE, mfinal=20)
#df1.adabag_cv
#(140+250)/520=75
#0.75 this is good than gbm boosting with nfinal = 20.

set.seed(850)
df1.adabag_cv <- boosting.cv(Credit.Standing~.-Credit.Standing1, data = df1[train1,],v=10, boos
 = TRUE, mfinal=50)
```

```
## i:  1 Thu Dec 05 16:12:55 2019
## i:  2 Thu Dec 05 16:13:07 2019
## i:  3 Thu Dec 05 16:13:18 2019
## i:  4 Thu Dec 05 16:13:30 2019
## i:  5 Thu Dec 05 16:13:42 2019
## i:  6 Thu Dec 05 16:13:54 2019
## i:  7 Thu Dec 05 16:14:16 2019
## i:  8 Thu Dec 05 16:14:44 2019
## i:  9 Thu Dec 05 16:15:11 2019
## i:  10 Thu Dec 05 16:15:39 2019
```

```
df1.adabag_cv
```

```
## $class
##   [1] "Good" "Good" "Good" "Good" "Bad"  "Bad"  "Good" "Bad"  "Good" "Good"
##  [11] "Good" "Good" "Good" "Good" "Good" "Good" "Bad"  "Bad"  "Bad"  "Good"
##  [21] "Bad"  "Bad"  "Good" "Bad"  "Bad"  "Good" "Good" "Good" "Good" "Bad"
##  [31] "Good" "Bad"  "Bad"  "Good" "Good" "Good" "Good" "Good" "Bad"  "Good"
##  [41] "Bad"  "Good" "Good" "Good" "Bad"  "Bad"  "Bad"  "Good" "Good" "Good"
##  [51] "Bad"  "Bad"  "Bad"  "Good" "Bad"  "Good" "Bad"  "Bad"  "Good" "Bad"
##  [61] "Good" "Good" "Good" "Bad"  "Bad"  "Good" "Good" "Bad"  "Bad"  "Bad"
##  [71] "Good" "Bad"  "Good" "Good" "Good" "Good" "Bad"  "Good" "Bad"  "Good"
##  [81] "Good" "Good" "Good" "Bad"  "Good" "Good" "Good" "Bad"  "Good" "Good"
##  [91] "Bad"  "Bad"  "Good" "Bad"  "Good" "Good" "Bad"  "Bad"  "Good" "Bad"
## [101] "Bad"  "Good" "Good" "Good" "Good" "Good" "Bad"  "Bad"  "Good" "Good"
## [111] "Good" "Good" "Bad"  "Good" "Bad"  "Good" "Bad"  "Bad"  "Good" "Good"
## [121] "Good" "Good" "Good" "Good" "Good" "Good" "Bad"  "Good" "Good" "Bad"
## [131] "Good" "Good" "Bad"  "Good" "Good" "Good" "Good" "Bad"  "Bad"  "Good"
## [141] "Bad"  "Good" "Good" "Bad"  "Bad"  "Bad"  "Bad"  "Good" "Good" "Good"
## [151] "Good" "Bad"  "Good" "Good" "Good" "Bad"  "Bad"  "Bad"  "Bad"  "Good"
## [161] "Good" "Good" "Bad"  "Bad"  "Good" "Good" "Good" "Good" "Good" "Bad"
## [171] "Bad"  "Good" "Bad"  "Good" "Bad"  "Bad"  "Good" "Bad"  "Good" "Bad"
## [181] "Bad"  "Good" "Good" "Bad"  "Bad"  "Good" "Good" "Good" "Good" "Good"
## [191] "Bad"  "Good" "Good" "Good" "Good" "Bad"  "Good" "Bad"  "Good" "Bad"
## [201] "Good" "Good" "Bad"  "Bad"  "Good" "Good" "Good" "Good" "Good" "Good"
## [211] "Good" "Bad"  "Bad"  "Bad"  "Bad"  "Bad"  "Bad"  "Bad"  "Good" "Bad"
## [221] "Bad"  "Bad"  "Good" "Good" "Bad"  "Good" "Bad"  "Good" "Bad"  "Bad"
## [231] "Bad"  "Good" "Good" "Bad"  "Good" "Good" "Bad"  "Bad"  "Good" "Bad"
## [241] "Bad"  "Bad"  "Bad"  "Good" "Good" "Good" "Bad"  "Good" "Good" "Bad"
## [251] "Good" "Good" "Good" "Good" "Bad"  "Good" "Good" "Good" "Bad"  "Bad"
## [261] "Good" "Good" "Good" "Good" "Bad"  "Good" "Bad"  "Good" "Bad"  "Bad"
## [271] "Bad"  "Good" "Good" "Good" "Bad"  "Good" "Bad"  "Good" "Good" "Good"
## [281] "Bad"  "Good" "Bad"  "Good" "Good" "Bad"  "Good" "Good" "Bad"  "Good"
## [291] "Good" "Good" "Bad"  "Good" "Good" "Good" "Good" "Good" "Good" "Bad"
## [301] "Good" "Good" "Good" "Good" "Good" "Good" "Good" "Good" "Good" "Good"
## [311] "Good" "Bad"  "Good" "Bad"  "Bad"  "Good" "Bad"  "Good" "Good" "Bad"
## [321] "Good" "Bad"  "Good" "Good" "Bad"  "Good" "Good" "Good" "Good" "Bad"
## [331] "Bad"  "Good" "Good" "Bad"  "Good" "Bad"  "Bad"  "Good" "Good" "Good"
## [341] "Good" "Good" "Good" "Bad"  "Good" "Bad"  "Good" "Good" "Good" "Good"
## [351] "Bad"  "Bad"  "Good" "Good" "Bad"  "Bad"  "Good" "Good" "Good" "Bad"
## [361] "Bad"  "Good" "Good" "Good" "Good" "Good" "Bad"  "Bad"  "Bad"  "Bad"
## [371] "Good" "Good" "Good" "Bad"  "Good" "Bad"  "Bad"  "Bad"  "Bad"  "Good"
## [381] "Good" "Good" "Bad"  "Good" "Good" "Good" "Bad"  "Good" "Bad"  "Bad"
## [391] "Bad"  "Bad"  "Good" "Bad"  "Good" "Bad"  "Bad"  "Bad"  "Good" "Good"
## [401] "Good" "Good" "Good" "Good" "Good" "Bad"  "Good" "Good" "Good" "Good"
## [411] "Good" "Bad"  "Good" "Bad"  "Good" "Good" "Good" "Bad"  "Bad"  "Good"
## [421] "Bad"  "Good" "Good" "Good" "Good" "Good" "Good" "Bad"  "Good" "Good"
## [431] "Good" "Good" "Good" "Good" "Bad"  "Bad"  "Bad"  "Good" "Bad"  "Good"
## [441] "Good" "Bad"  "Good" "Good" "Bad"  "Bad"  "Good" "Bad"  "Good" "Bad"
## [451] "Good" "Bad"  "Bad"  "Bad"  "Bad"  "Good" "Good" "Good" "Bad"  "Good"
## [461] "Bad"  "Bad"  "Good" "Bad"  "Good" "Bad"  "Good" "Good" "Good" "Bad"
## [471] "Good" "Good" "Good" "Bad"  "Bad"  "Good" "Bad"  "Bad"  "Good" "Good"
## [481] "Bad"  "Bad"  "Bad"  "Bad"  "Bad"  "Good" "Bad"  "Good" "Good" "Good"
## [491] "Good" "Good" "Good" "Bad"  "Good" "Good" "Good" "Good" "Bad"  "Good"
## [501] "Good" "Good" "Good" "Bad"  "Bad"  "Bad"  "Bad"  "Good" "Bad"  "Good"
## [511] "Good" "Bad"  "Good" "Good" "Bad"  "Good" "Good" "Good" "Bad"  "Bad"
```

```
##
## $confusion
##                Observed Class
## Predicted Class Bad Good
##            Bad  144   63
##            Good  63  250
##
## $error
## [1] 0.2423077
```

```
#(144+250)/520 =75.76
# 0.75.76, This is quite better with mfinal = 50.

#set.seed(850)
#df1.adabag_cv <- boosting.cv(Credit.Standing~.-Credit.Standing1, data = df1[train1,],v=10, boos
= TRUE, mfinal=30)
#df1.adabag_cv
#(144+252)/520=76.15
# 0.7615 this is almost same to mfinal = 50

#set.seed(850)
#df1.adabag_cv <- boosting.cv(Credit.Standing~.-Credit.Standing1, data = df1[train1,],v=10, boos
= TRUE, mfinal=15)
#df1.adabag_cv
#(145+256)/520=77.11
# 0.7711 this is the better for mfinal = 50


# Changging rpart to maxdepth = 5

#set.seed(850)
#df1.adabag_cv <- boosting.cv(Credit.Standing~.-Credit.Standing1, data = df1[train1,],v=10, boos
= TRUE, mfinal=20,control=rpart.control(maxdepth=5))
#df1.adabag_cv
#(139+251)/520=75
#0.75 this is the worst of all.

#set.seed(850)
#df1.adabag_cv <- boosting.cv(Credit.Standing~.-Credit.Standing1, data = df1[train1,],v=10, boos
= TRUE, mfinal=30,control=rpart.control(maxdepth=5))
#df1.adabag_cv
#(146+251)/520=76.34
#0.7634 this is the best so far.

set.seed(850)
df1.adabag_cv <- boosting.cv(Credit.Standing~.-Credit.Standing1, data = df1[train1,],v=10, boos
 = TRUE, mfinal=50,control=rpart.control(maxdepth=5))
```

```
## i:  1 Thu Dec 05 16:16:06 2019
## i:  2 Thu Dec 05 16:16:34 2019
## i:  3 Thu Dec 05 16:17:01 2019
## i:  4 Thu Dec 05 16:17:29 2019
## i:  5 Thu Dec 05 16:17:56 2019
## i:  6 Thu Dec 05 16:18:22 2019
## i:  7 Thu Dec 05 16:18:50 2019
## i:  8 Thu Dec 05 16:19:09 2019
## i:  9 Thu Dec 05 16:19:20 2019
## i:  10 Thu Dec 05 16:19:32 2019
```

```
df1.adabag_cv
```

```
## $class
##   [1] "Good" "Good" "Good" "Good" "Bad"  "Bad"  "Good" "Bad"  "Good" "Good"
##  [11] "Good" "Good" "Good" "Good" "Good" "Good" "Bad"  "Bad"  "Bad"  "Good"
##  [21] "Bad"  "Bad"  "Good" "Bad"  "Good" "Good" "Good" "Good" "Bad"  "Bad"
##  [31] "Good" "Bad"  "Bad"  "Good" "Good" "Good" "Good" "Good" "Bad"  "Good"
##  [41] "Bad"  "Good" "Good" "Bad"  "Bad"  "Bad"  "Bad"  "Good" "Good" "Good"
##  [51] "Bad"  "Bad"  "Good" "Good" "Bad"  "Good" "Bad"  "Bad"  "Good" "Bad"
##  [61] "Good" "Good" "Good" "Bad"  "Bad"  "Good" "Good" "Bad"  "Bad"  "Bad"
##  [71] "Good" "Bad"  "Good" "Good" "Good" "Good" "Bad"  "Good" "Bad"  "Good"
##  [81] "Good" "Good" "Good" "Bad"  "Good" "Good" "Good" "Bad"  "Good" "Good"
##  [91] "Good" "Bad"  "Good" "Bad"  "Good" "Good" "Bad"  "Bad"  "Good" "Bad"
## [101] "Bad"  "Good" "Good" "Good" "Good" "Bad"  "Bad"  "Bad"  "Good" "Good"
## [111] "Good" "Good" "Bad"  "Good" "Bad"  "Good" "Bad"  "Bad"  "Good" "Good"
## [121] "Good" "Good" "Good" "Bad"  "Good" "Good" "Bad"  "Good" "Good" "Bad"
## [131] "Good" "Good" "Bad"  "Good" "Good" "Good" "Good" "Bad"  "Bad"  "Good"
## [141] "Bad"  "Bad"  "Good" "Bad"  "Bad"  "Good" "Bad"  "Good" "Good" "Good"
## [151] "Good" "Bad"  "Good" "Good" "Good" "Bad"  "Bad"  "Bad"  "Bad"  "Good"
## [161] "Bad"  "Bad"  "Bad"  "Bad"  "Good" "Good" "Good" "Good" "Good" "Bad"
## [171] "Bad"  "Good" "Bad"  "Good" "Bad"  "Bad"  "Good" "Bad"  "Good" "Bad"
## [181] "Bad"  "Good" "Good" "Bad"  "Bad"  "Good" "Good" "Good" "Good" "Good"
## [191] "Bad"  "Good" "Good" "Good" "Good" "Bad"  "Good" "Bad"  "Good" "Bad"
## [201] "Good" "Good" "Bad"  "Bad"  "Good" "Good" "Good" "Good" "Good" "Good"
## [211] "Good" "Bad"  "Good" "Bad"  "Bad"  "Bad"  "Good" "Bad"  "Good" "Bad"
## [221] "Bad"  "Bad"  "Good" "Good" "Bad"  "Bad"  "Bad"  "Good" "Bad"  "Good"
## [231] "Bad"  "Good" "Good" "Bad"  "Good" "Good" "Bad"  "Bad"  "Good" "Bad"
## [241] "Bad"  "Bad"  "Bad"  "Good" "Good" "Good" "Bad"  "Good" "Good" "Bad"
## [251] "Good" "Good" "Good" "Good" "Bad"  "Good" "Good" "Good" "Bad"  "Good"
## [261] "Good" "Good" "Bad"  "Good" "Bad"  "Good" "Bad"  "Good" "Bad"  "Bad"
## [271] "Bad"  "Good" "Good" "Good" "Bad"  "Good" "Bad"  "Good" "Good" "Good"
## [281] "Bad"  "Good" "Bad"  "Good" "Good" "Bad"  "Bad"  "Good" "Bad"  "Good"
## [291] "Good" "Good" "Bad"  "Good" "Good" "Good" "Good" "Good" "Good" "Bad"
## [301] "Good" "Good" "Good" "Good" "Good" "Good" "Good" "Good" "Good" "Good"
## [311] "Good" "Bad"  "Good" "Bad"  "Bad"  "Good" "Bad"  "Good" "Good" "Bad"
## [321] "Good" "Bad"  "Good" "Good" "Good" "Good" "Good" "Good" "Good" "Bad"
## [331] "Bad"  "Good" "Good" "Good" "Good" "Bad"  "Bad"  "Good" "Good" "Good"
## [341] "Good" "Good" "Good" "Bad"  "Good" "Bad"  "Good" "Good" "Good" "Good"
## [351] "Bad"  "Bad"  "Good" "Good" "Good" "Bad"  "Good" "Good" "Good" "Good"
## [361] "Bad"  "Good" "Good" "Good" "Good" "Good" "Bad"  "Bad"  "Bad"  "Good"
## [371] "Good" "Bad"  "Good" "Bad"  "Good" "Bad"  "Bad"  "Bad"  "Bad"  "Good"
## [381] "Good" "Good" "Bad"  "Good" "Good" "Good" "Bad"  "Good" "Bad"  "Bad"
## [391] "Bad"  "Bad"  "Good" "Bad"  "Good" "Bad"  "Bad"  "Bad"  "Good" "Good"
## [401] "Good" "Good" "Good" "Good" "Good" "Bad"  "Good" "Good" "Good" "Good"
## [411] "Good" "Bad"  "Good" "Bad"  "Good" "Good" "Good" "Good" "Bad"  "Good"
## [421] "Bad"  "Good" "Good" "Good" "Good" "Good" "Good" "Good" "Good" "Good"
## [431] "Good" "Good" "Good" "Good" "Bad"  "Bad"  "Bad"  "Good" "Bad"  "Good"
## [441] "Good" "Bad"  "Good" "Good" "Bad"  "Bad"  "Good" "Bad"  "Bad"  "Bad"
## [451] "Good" "Bad"  "Bad"  "Bad"  "Bad"  "Good" "Good" "Good" "Bad"  "Good"
## [461] "Bad"  "Bad"  "Good" "Bad"  "Good" "Bad"  "Good" "Good" "Bad"  "Good"
## [471] "Good" "Good" "Good" "Bad"  "Bad"  "Good" "Bad"  "Bad"  "Good" "Bad"
## [481] "Good" "Bad"  "Bad"  "Bad"  "Bad"  "Good" "Bad"  "Good" "Bad"  "Good"
## [491] "Good" "Good" "Good" "Bad"  "Good" "Good" "Good" "Good" "Bad"  "Good"
## [501] "Bad"  "Good" "Good" "Bad"  "Bad"  "Bad"  "Bad"  "Good" "Bad"  "Good"
## [511] "Good" "Bad"  "Good" "Good" "Good" "Good" "Good" "Good" "Bad"  "Bad"
```

```
##
## $confusion
##                Observed Class
## Predicted Class Bad Good
##            Bad  147   58
##            Good  60  255
##
## $error
## [1] 0.2269231
```

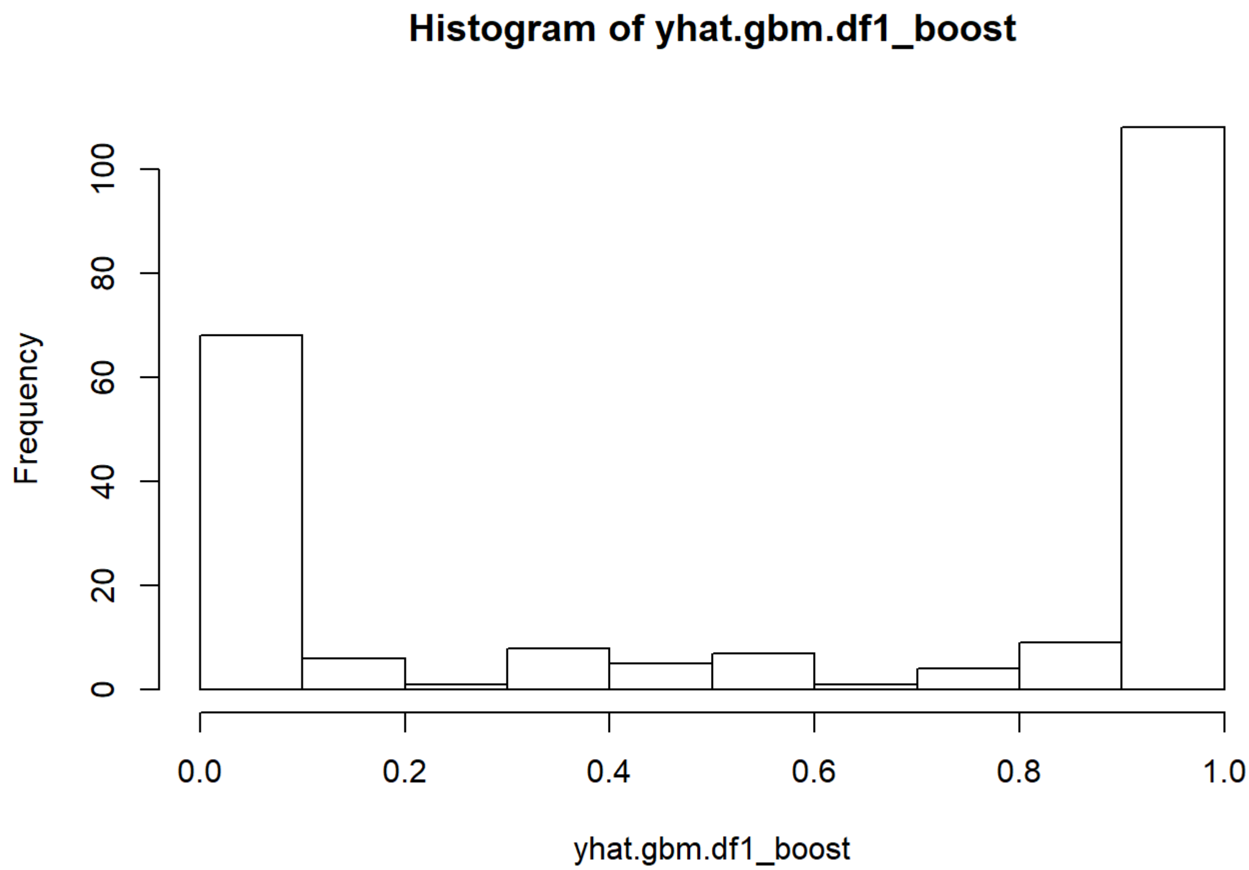
```
#(147+255)/520=77.30
#0.7730 this is the best adabag so far.

#set.seed(850)
#df1.adabag_cv <- boosting.cv(Credit.Standing~.-Credit.Standing1, data = df1[train1,],v=10, boos
= TRUE, mfinal=15,control=rpart.control(maxdepth=5))
#df1.adabag_cv
#(136+259)/520=75.96
#This is worst

# When tuning parameters is finished, check on test set; here best parameters are mfinal =50, ma
xdepth = 5.
set.seed(850)
df1.adabag <- boosting(Credit.Standing~.-Credit.Standing1, data = df1[train1,], boos = TRUE, mfi
nal=50,control=rpart.control(maxdepth=5))

yhat.df1_adabag=predict(df1.adabag,newdata=df1[-train1,],n.trees=50,type = "response")

hist(yhat.gbm.df1_boost)
```




```
predict_class <- ifelse(yhat.gbm.df1_boost<0.5,"Bad","Good")
table(predict_class,df1[-train1,"Credit.Standing1"])
```

```
##
## predict_class  0  1
##          Bad  65 23
##          Good 33 96
```

```
#(65+96)/217=74.19
```

Comment:-

I have tried to improve my model using three approaches i.e. Random forest, Gradient boosting and AdaBoost. In random forest, data is split into train and test with the ratio of 70:30. Firstly, I run the model on train data and calculate the prediction and confusion matrix. There is mismatch between OOB [Out of Bag] (24.81%) and accuracy (95.19%). Then, I run the model on test data and calculate the prediction and confusion matrix. So, I get the accuracy (75.58) which is better than decision tree. Further I tried to improve my model using gradient boosting but my accuracy get decrease and comes down to 74.14%. Since the data is noisy, gradient boosting doesn't perform well and it can result to overfitting. Gradient boosting is hard to tune than random forest. Later, I tried to improve my model using adaboost. I run the model many times by adjusting 'mfinal' and 'maxdepth'. Finally, there is an improvement in the model accuracy with 'mfinal = 50' and 'maxdepth = 5'. So, the accuracy now is 77.30% which is the best among all the models. I think adaboost is the best model for me.

Question E

Kate’s company uses a process that is a mixture of a grading system and human input to grade each past loan as good or bad. Kate is suspicious that during a particular time that this process performed very poorly and produced inaccurate results. Develop a strategy so that you can you find a series of consecutive or nearly consecutive ID numbers of circa 10 or more, i.e. where these gradings show a suspiciously incorrect pattern. Detail how you go about your investigation and how you find this pattern.

```
##Daisy Function using 'Gower Measure'

K_df<-df1
K_df<-subset(df1, select = Checking.Acct:Credit.Standing)
#View(K_df)
#' Compute Gower distance
gower_dist <- daisy(K_df, metric = "gower")

summary(gower_dist)
```

```
## 271216 dissimilarities, summarized :
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.4076  0.4895  0.4890  0.5703  0.9273
## Metric :  mixed ;  Types = N, N, N, N, N, N, N, N, N, I, I, I, N
## Number of objects : 737
```

```
gower_mat <- as.matrix(gower_dist)

#' Print most similar clients
K_df[which(gower_mat == min(gower_mat[gower_mat != min(gower_mat)]), arr.ind = TRUE)[1, ], ]
```

	Checking.Acct <fctr>	Credit.History <fctr>	Loan.Reason <fctr>	Savings.Acct <fctr>	Employm... <fctr>	Personal.Status <fctr>
630	0Balance	Current	Furniture	Low	Long	Single
123	0Balance	Current	Furniture	Low	Long	Single

2 rows | 1-8 of 14 columns

```
#' Print most dissimilar clients
K_df[which(gower_mat == max(gower_mat[gower_mat != max(gower_mat)]), arr.ind = TRUE)[1, ], ]
```

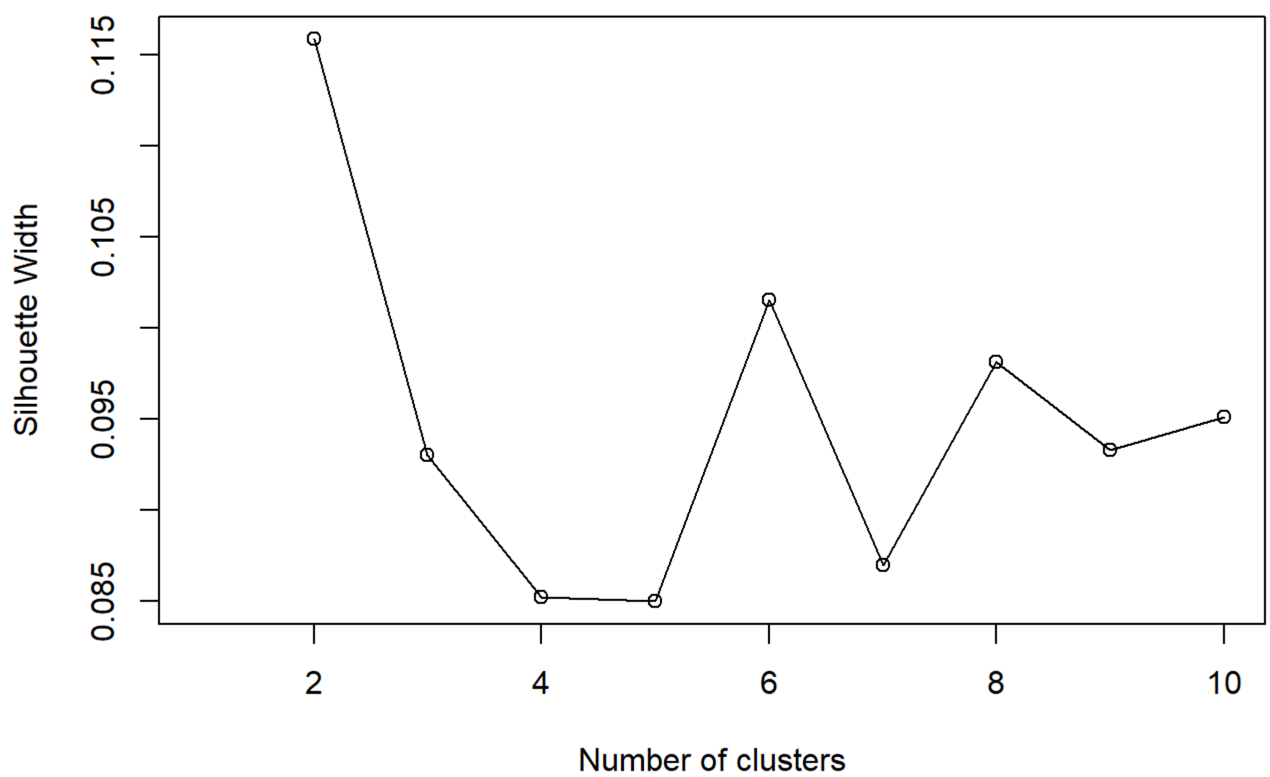
	Checking.Acct <fctr>	Credit.History <fctr>	Loan.Reason <fctr>	Savings.Acct <fctr>	Employm... <fctr>	Personal.Status <fctr>
616	No Acct	Current	Business	Low	Unemployed	Single
441	High	Bank Paid	Car New	High	Retired	Divorced

2 rows | 1-8 of 14 columns

```
# Calculate silhouette width for many k using PAM
set.seed(850)
sil_width <- c(NA)
for(i in 2:10){
  pam_fit <- pam(gower_dist,diss = TRUE,k = i)
  sil_width[i] <- pam_fit$silinfo$avg.width
    }
sil_width
```

```
##  [1]         NA 0.11584841 0.09303920 0.08520688 0.08499484 0.10152188
##  [7] 0.08700349 0.09813214 0.09328577 0.09513239
```

```
# Plot sihouette width (higher is better)
plot(1:10, sil_width,
xlab = "Number of clusters",ylab = "Silhouette Width")
lines(1:10, sil_width)
```



```
#Cluster Interpretation Via Descriptive Statistics
pam_fit <- pam(gower_dist, diss = TRUE, k=2)
pam_results <- K_df %>%
mutate(cluster = pam_fit$clustering) %>%
group_by(cluster) %>%
do(the_summary = summary(.))
pam_results$the_summary
```

```
## [[1]]
##   Checking.Acct   Credit.History          Loan.Reason   Savings.Acct
##  0Balance: 77   All Paid :108    Small Appliance:132   High   : 18
##  High    : 19   Bank Paid: 32    Car New        : 58   Low    :229
##  Low     : 98   Critical : 12    Furniture      : 56   MedHigh: 30
##  No Acct :195   Current  :195    Business       : 51   MedLow : 43
##                 Delay    : 42    Car Used       : 48   No Acct: 69
##                                  Education      : 18
##                                  (Other)        : 26
##       Employment  Personal.Status  Housing          Job.Type
##  Long      :122   Divorced: 63    Other: 48   Management: 68
##  Medium    :119   Married : 32    Own  :280   Skilled   :240
##  Retired   :  1   Single  :294    Rent : 61   Unemployed:  5
##  Short     : 74                               Unskilled : 76
##  Unemployed: 23
##  Very Short: 50
##
##  Foreign.National Months.since.Checking.Acct.opened
##  No :130          Min.   : 5.00
##  Yes:259          1st Qu.:13.00
##                   Median :25.00
##                   Mean   :24.46
##                   3rd Qu.:31.00
##                   Max.   :73.00
##
##  Residence.Time..In.current.district.      Age        Credit.Standing
##  Min.   :-2.000                       Min.   :20.00   Bad : 54
##  1st Qu.: 2.000                       1st Qu.:28.00   Good:335
##  Median : 3.000                       Median :34.00
##  Mean   : 2.897                       Mean   :36.22
##  3rd Qu.: 4.000                       3rd Qu.:42.00
##  Max.   : 9.000                       Max.   :99.00
##
##     cluster
##  Min.   :1
##  1st Qu.:1
##  Median :1
##  Mean   :1
##  3rd Qu.:1
##  Max.   :1
##
##
## [[2]]
##   Checking.Acct   Credit.History          Loan.Reason   Savings.Acct
##  0Balance:169   All Paid : 22    Car New        :120   High   : 10
##  High    : 23   Bank Paid: 28    Furniture      : 97   Low    :241
##  Low     : 92   Critical : 72    Small Appliance: 41   MedHigh: 17
##  No Acct : 64   Current  :207    Business       : 29   MedLow : 35
##                 Delay    : 19    Education      : 25   No Acct: 45
##                                  Car Used       : 23
##                                  (Other)        : 13
##       Employment  Personal.Status  Housing          Job.Type
##  Long      : 59   Divorced:195    Other: 40   Management: 30
```

```
##  Medium    : 19   Married : 36    Own  :219   Skilled   :227
##  Retired   :  1   Single  :117    Rent : 89   Unemployed: 13
##  Short     :166                               Unskilled : 78
##  Unemployed: 20
##  Very Short: 83
##
##  Foreign.National Months.since.Checking.Acct.opened
##  No :104          Min.   :  5.00
##  Yes:244          1st Qu.: 13.00
##                   Median : 19.00
##                   Mean   : 21.61
##                   3rd Qu.: 25.00
##                   Max.   :120.00
##
##  Residence.Time..In.current.district.      Age        Credit.Standing
##  Min.   : 1.000                       Min.   :18.00   Bad :251
##  1st Qu.: 2.000                       1st Qu.:24.00   Good: 97
##  Median : 3.000                       Median :30.00
##  Mean   : 2.793                       Mean   :33.15
##  3rd Qu.: 4.000                       3rd Qu.:39.25
##  Max.   :10.000                       Max.   :85.00
##
##     cluster
##  Min.   :2
##  1st Qu.:2
##  Median :2
##  Mean   :2
##  3rd Qu.:2
##  Max.   :2
##
```

```
table(K_df$Credit.Standing)
```

```
##
##  Bad Good
##  305  432
```

```
#Via Visualization
tsne_obj <- Rtsne(gower_dist, is_distance = TRUE)
tsne_data <- tsne_obj$Y %>%
data.frame() %>%
setNames(c("X", "Y")) %>%
mutate(cluster = factor(pam_fit$clustering))
ggplot(aes(x = X, y = Y), data = tsne_data) +
geom_point(aes(color = cluster))
```

Comment:-

In this question, I am to use ‘Daisy Function’. This function finds the distance/dissimilarity between rows when the variables are not in the same format. It handles the data of mix variables like numeric, binary, nominal and ordinal data. Once we pass the data to daisy function it would return a distance matrix. We can’t apply K-means clustering function on the output of daisy function because K-means function cannot cluster the data on the basis of distance matrix. So, if want to cluster the data using the distance matrix then we can do it by ‘Kmedoid(PAM)’ and hierarchical clustering. Daisy function uses the ‘Gower Measure’ for calculating the distance between rows that contain mixed data. Gower measure apply suitable distance measuring techniques that corresponds with the data type of the columns. For continuous variable, gower measure uses manhattan distance whereas for binary/nominal variables it uses dice distance and for ordinal data, it converts the rank to simple ranks and then uses manhattan distance to calculate the distance between the rows. Daisy function has an argument called ‘metric’ which can be set to distance measure such as Euclidean and Manhattan. Using this function we can tell the function to calculate the distance between rows. If the data is of mixed type then the distance measure would automatically become gower. In this, we are going to use ‘Silhouette coefficient’ to choose the clusters. Objects having high silhouette value are considered well clustered, objects with a low value may be outliers. After the analysis, we see that two cluster is formed. We can see that there are some dissimilarity in the data as some data points are overlapping each other. I tried to find out the patterns using daisy function but I unable to find. There is another way to find the patterns using for loop and then testing the data using train and test. For example, we can split the data into test and train like we take first 15 rows as test data and remaining data as train. We run the model as number of observations we have to get the patterns. So, just looking at the excel sheet I came across one patterns which is incorrect. From observation number 624 to 639 something are the consecutive series having suspicious patterns.