Data Science & Analytics ASSIGNMENT

## SHIVAANI KATRAGADDA

## R00183214

# part 1- a,b,c,d,e questions

In this assignment, I was given a Credit\_Risk6\_final.xlsx file which contains two sheets, which were named Scoring\_Data and Training\_Data.

The Scoring\_Data i.e sheet1 contains thirteen columns and 13 rows of data.The Training\_Data i.e Sheet2 contains fourteen columns, 780 rows of data.

The thirteen columns in the Scoring\_Data are named as follows, ID, Checking Acct, Credit History, Loan Reason, Savings Acct, Employment, Personal Status, Housing, Job Type, Foreign National, Months since Checking Acct opened, Residence Time, Age.

In Training\_Data there are fourteen columns which are named as follows ID, Checking Acct(have different levels they are as follows No acct, 0balance, low (balance), high (balance)),Credit History(have different levels/domains are as follows All paid – no credit taken or all credit paid back duly,Bank Paid – All credit at this bank paid back, Current – Existing loan/credit paid back duly till now,Critical – Risky account or other credits at other banks, Delay – Delay in paying back credit/loan in the past ), Loan Reason, Savings Acct, Employment, Personal Status, Housing, Job Type, Foreign National, Months since Checking Acct opened(The number of months the customer has an account with the bank), Residence Time (In current district), Age, Credit Standing.

Training\_Data contains the same columns that are present in Scoring\_Data but there is one extra column which is different that is Credit Standing column which is a outcome/label variable classifying each case as either a good loan or bad loan. This is the little introduction about the given dataset.

# install.packages("readxl")  
# install.packages("ggplot2")  
# install.packages("DataExplorer")  
# install.packages("caret")  
# install.packages("rpart")  
# install.packages("rpart.plot")  
# install.packages("dplyr")  
# install.packages("C50")  
# install.packages("randomForest")  
# install.packages("GGally")  
  
library(readxl)#readxl package useful to get the data from excel to R easily.  
  
library(ggplot2)#ggplot2 package is particularly useful for visualizing the data.  
  
library(DataExplorer)#DataExplorer package is useful for visualizing and analysing the data  
  
library(caret)#Caret(Classification And REgression Training) package contains set of functions that are useful for creating predictive models

## Loading required package: lattice

library(rpart)#rpart(Recursive Partitioning And Regression Trees) package which is useful for creating decision tree  
  
library(rpart.plot)#rpart.plot packages which will scales and adjusts the displayed tree for best fit  
  
library(dplyr)#dplyr package which is useful for manipulating datasets in R very effectively.

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

library(C50)#c50 package which is useful to build classification trees in R  
  
library(randomForest)#randomForest package which is useful to build classification trees in R

## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##   
## Attaching package: 'randomForest'

## The following object is masked from 'package:dplyr':  
##   
## combine

## The following object is masked from 'package:ggplot2':  
##   
## margin

#GGally package which extends ggplot2 by adding several functions to reduce the complexity of combining geoms with transformed data.  
library(GGally)

## Registered S3 method overwritten by 'GGally':  
## method from   
## +.gg ggplot2

##   
## Attaching package: 'GGally'

## The following object is masked from 'package:dplyr':  
##   
## nasa

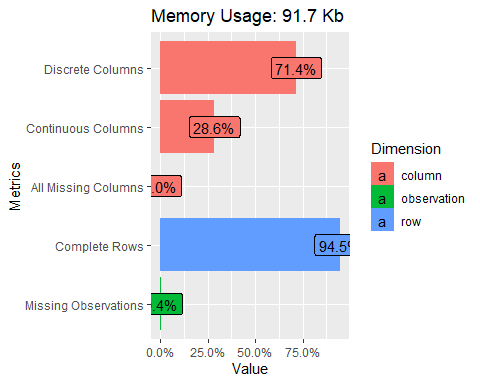
### QUESTION(a)

Exploratory Data Analysis (EDA): - Carry out some EDA on the data set; carry out at least one trivariate analysis; do you notice anything unusual or any patterns with the data set? Detail these and outline any actions you propose to take before you start model building in part b). Max word count 500 words.

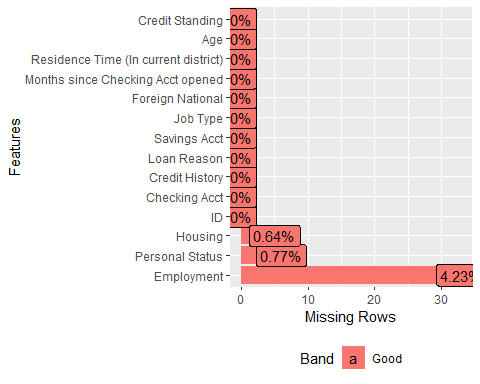
#Reading the scoring\_Data of Credit\_Risk6\_final.xlsx file by using read\_excel() function and it is assigned to Credit\_Risk6\_final1  
  
Credit\_Risk6\_final1 <- read\_excel("F:/Ds Assignment/Credit\_Risk6\_final.xlsx",sheet = "Scoring\_Data")  
  
#Generating the dataframe for Credit\_Risk6\_final1 using as.data.frame() function and that dataframe is named as Credit\_Scoring\_Data   
Credit\_Scoring\_Data<- as.data.frame(Credit\_Risk6\_final1)  
  
#Reading the Training\_Data of Credit\_Risk6\_final.xlsx file by using read\_excel() function and it is assigned to Credit\_Risk6\_final2  
Credit\_Risk6\_final2 <- read\_excel("F:/Ds Assignment/Credit\_Risk6\_final.xlsx", sheet = "Training\_Data")  
  
#Generating the dataframe for Credit\_Risk6\_final2 using as.data.frame() function and that dataframe is named as Credit\_Training\_Data  
Credit\_Training\_Data<- as.data.frame(Credit\_Risk6\_final2)  
  
#Performing Exploratory Data Analysis  
  
#DATA EXPLORER Package provides some good functions to know about the data  
#Introduce() function will give the outline of the data   
#it tells about the number of rows,columns,missing values,discrete columns,continous columns,all missing columns,complete rows,total observations and memory usage   
introduce(Credit\_Training\_Data)

## rows columns discrete\_columns continuous\_columns all\_missing\_columns  
## 1 780 14 10 4 0  
## total\_missing\_values complete\_rows total\_observations memory\_usage  
## 1 44 737 10920 93888

#plot\_intro() will plot a graph which will give the percentage of discrete columns,continous columns,All missing columns,complete rows and missing observations  
plot\_intro(Credit\_Training\_Data)



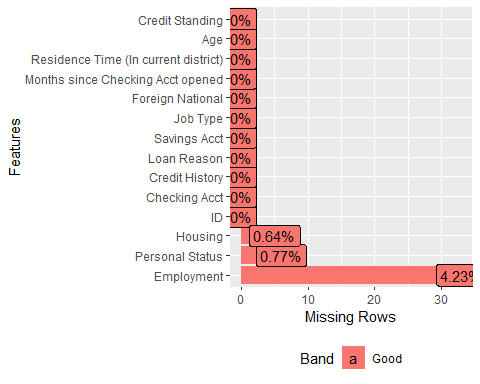
#plot\_missing() function shows the percentage of missing values of each column present in the dataset  
plot\_missing(Credit\_Training\_Data)



#Filling the NA values  
  
#The missing data is a categorical data so i am replacing the missing values with the mode   
  
dim(Credit\_Training\_Data) #dim() will give the number of rows and columns in the dataset

## [1] 780 14

#plot\_missing() function shows the percentage of missing values of each column present in the dataset  
plot\_missing(Credit\_Training\_Data)



#from plot\_missing() we know that we have missing values in three columns they are Housing,Personal status and Employment.  
  
table(is.na(Credit\_Training\_Data$Housing))#checking how many misssig values are there in the column

##   
## FALSE TRUE   
## 775 5

sort(table(Credit\_Training\_Data$Housing))#sorting the column in order to get the total number of observation for each domain in the column in ascending order

##   
## Other Rent Own   
## 94 157 524

names(table(Credit\_Training\_Data$Housing))[table(Credit\_Training\_Data$Housing)==max(table(Credit\_Training\_Data$Housing))]#now taking the domain which is having the high value

## [1] "Own"

Credit\_Training\_Data$Housing[is.na(Credit\_Training\_Data$Housing)] <- "Own"#Now assigining the highest value to the missing rows in the column  
table(is.na(Credit\_Training\_Data$Housing))#checking whether all the missing values are replace by running this line again

##   
## FALSE   
## 780

table(is.na(Credit\_Training\_Data$Employment))#checking how many misssig values are there in the column

##   
## FALSE TRUE   
## 747 33

sort(table(Credit\_Training\_Data$Employment))#sorting the column in order to get the total number of observation for each domain in the column in ascending order

##   
## Retired Unemployed Very Short Medium Long Short   
## 2 43 134 140 186 242

names(table(Credit\_Training\_Data$Employment))[table(Credit\_Training\_Data$Employment)==max(table(Credit\_Training\_Data$Employment))] #now taking the domain which is having the high value

## [1] "Short"

Credit\_Training\_Data$Employment[is.na(Credit\_Training\_Data$Employment)] <- "Short"#Now assigining the highest value to the missing rows in the column  
table(is.na(Credit\_Training\_Data$Employment))#checking whether all the missing values are replace by running this line again

##   
## FALSE   
## 780

table(is.na(Credit\_Training\_Data$`Personal Status`))#checking how many misssig values are there in the column

##   
## FALSE TRUE   
## 774 6

sort(table(Credit\_Training\_Data$`Personal Status`))#sorting the column in order to get the total number of observation for each domain in the column in ascending order

##   
## Married Divorced Single   
## 70 273 431

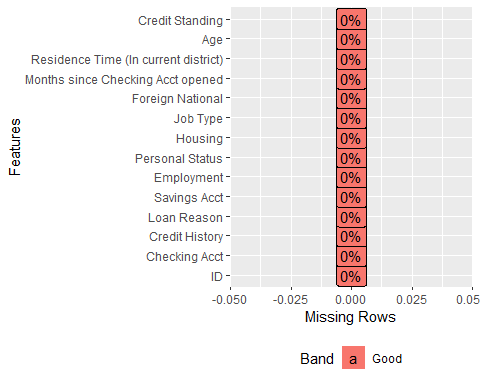
names(table(Credit\_Training\_Data$`Personal Status`))[table(Credit\_Training\_Data$`Personal Status`)==max(table(Credit\_Training\_Data$`Personal Status`))]#now taking the domain which is having the high value

## [1] "Single"

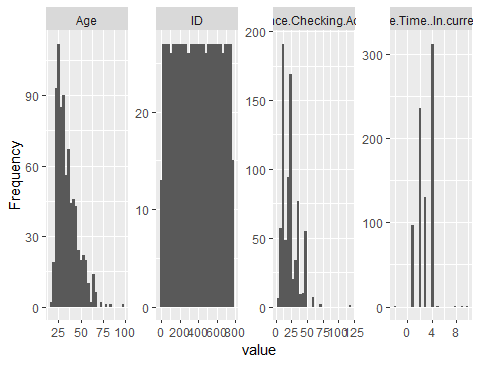
Credit\_Training\_Data$`Personal Status`[is.na(Credit\_Training\_Data$`Personal Status`)] <- "Single"#Now assigining the highest value to the missing rows in the column  
table(is.na(Credit\_Training\_Data$`Personal Status`))#checking whether all the missing values are replace by running this line again

##   
## FALSE   
## 780

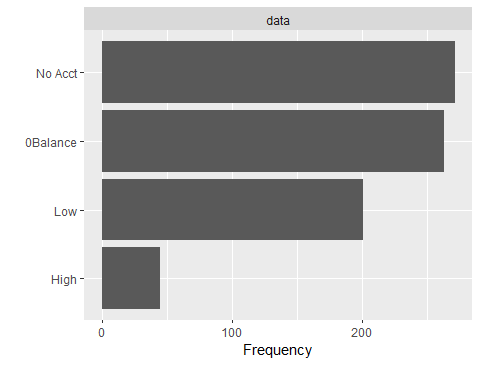
#All the missing data is imputated with mode check again whether any missimg data is present in the datase by using plot\_missing()  
plot\_missing(Credit\_Training\_Data)



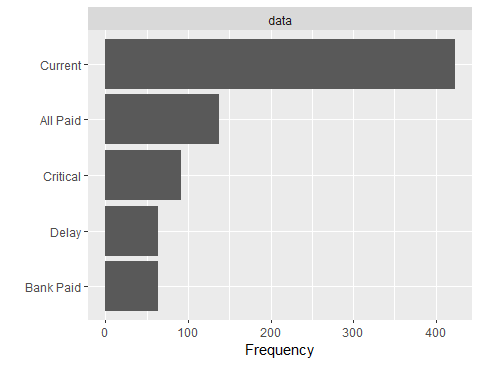
#Performing some Exploratory data analysis  
#plot\_histogram() is provided by DataExplorer package which will provide histograms of all columns which are containing the continuous data  
plot\_histogram(Credit\_Training\_Data)#This will give histogram for 4 columns they are AGE,ID,Months since checking account opened,Residence time(in current district)



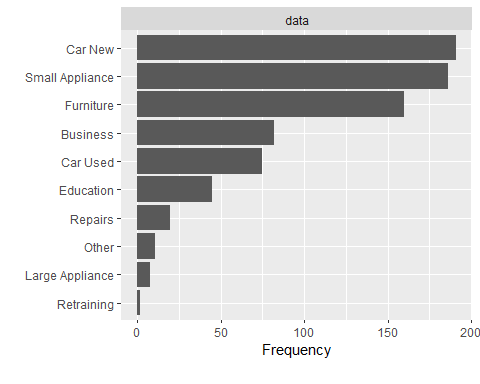
#plot\_bar() is provided by DataExplorer package which will provide barchart for discrete data  
  
plot\_bar(Credit\_Training\_Data$`Checking Acct`)#plot\_bar() for checking acct columnn of credit\_Training\_Data



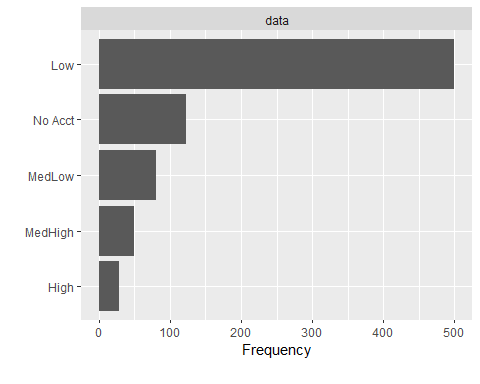
plot\_bar(Credit\_Training\_Data$`Credit History`)#plot\_bar() for credit History columnn of credit\_Training\_Data



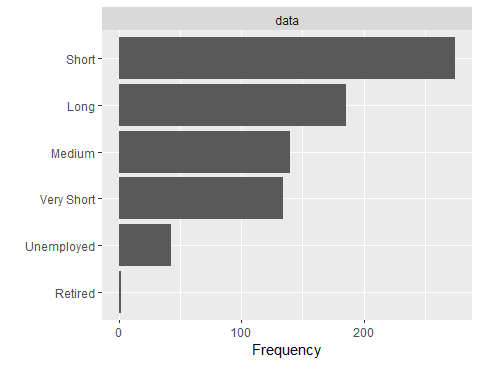
plot\_bar(Credit\_Training\_Data$`Loan Reason`)#plot\_bar() for Loan Reason columnn of credit\_Training\_Data



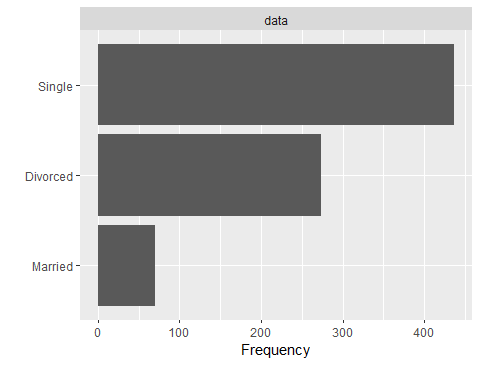
plot\_bar(Credit\_Training\_Data$`Savings Acct`)#plot\_bar() for Savings Acct columnn of credit\_Training\_Data



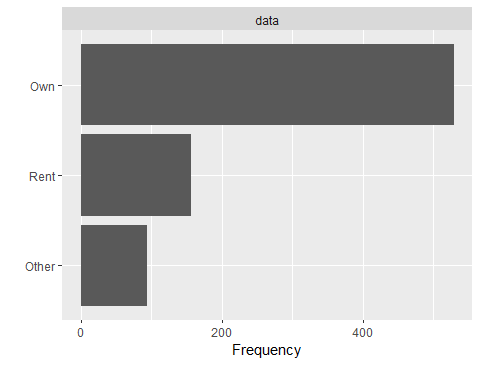
plot\_bar(Credit\_Training\_Data$Employment)#plot\_bar() for Employment columnn of credit\_Training\_Data



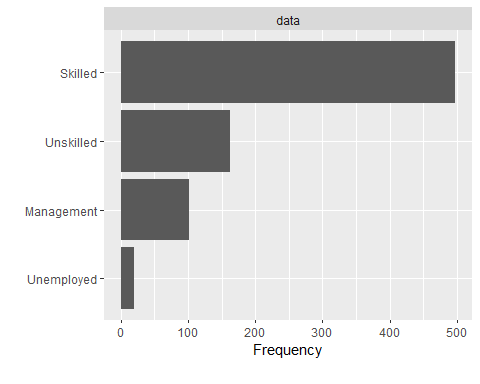
plot\_bar(Credit\_Training\_Data$`Personal Status`)#plot\_bar() for Personal Status columnn of credit\_Training\_Data



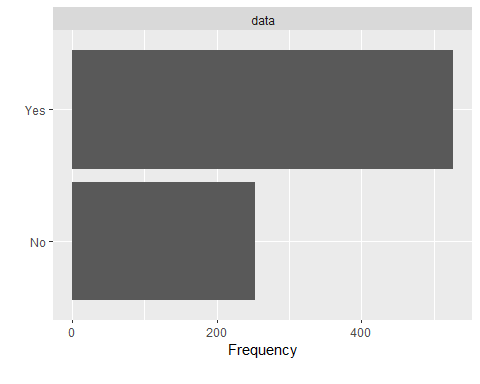
plot\_bar(Credit\_Training\_Data$Housing)#plot\_bar() for Housing columnn of credit\_Training\_Data



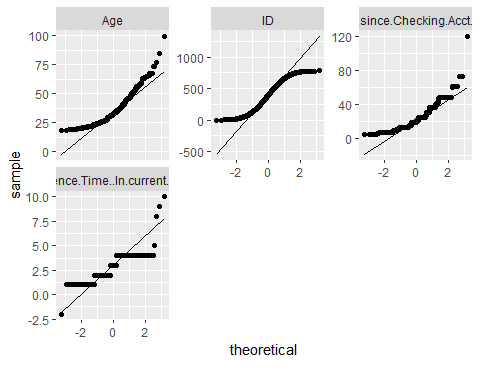
plot\_bar(Credit\_Training\_Data$`Job Type`)#plot\_bar() for Job Type columnn of credit\_Training\_Data



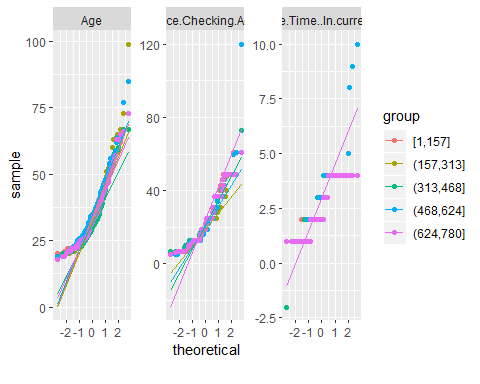
plot\_bar(Credit\_Training\_Data$`Foreign National`)#plot\_bar() for Foreign National columnn of credit\_Training\_Data



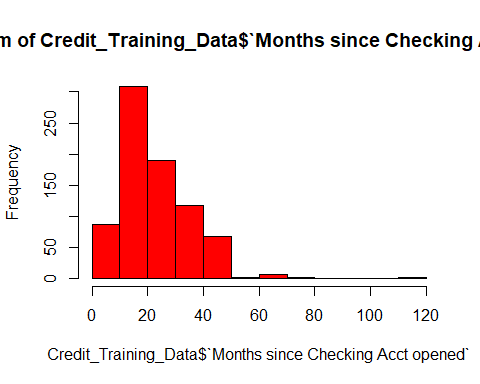
#qq plot i.e Quantile-Quantile plot is used to visualize the deviation from a specific probability distribution  
#qq plot for ID,AGE,MONTHS SINCE CHECKING ACCT OPENED,RESIDENCE TIME all these columns are assigned to qq\_data  
qq\_data <- Credit\_Training\_Data[, c("ID", "Age","Months since Checking Acct opened","Residence Time (In current district)")]  
plot\_qq(qq\_data)#plotting qq plot for qq\_data



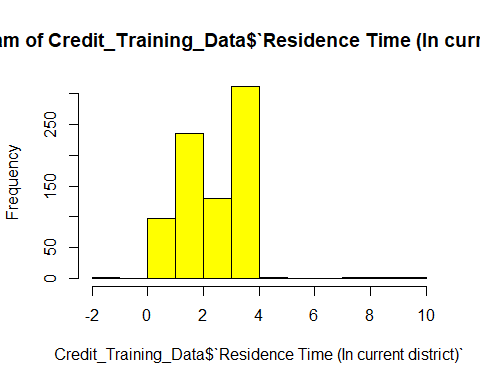
plot\_qq(qq\_data, by = "ID")#plotting qq plot by using id for qq\_data



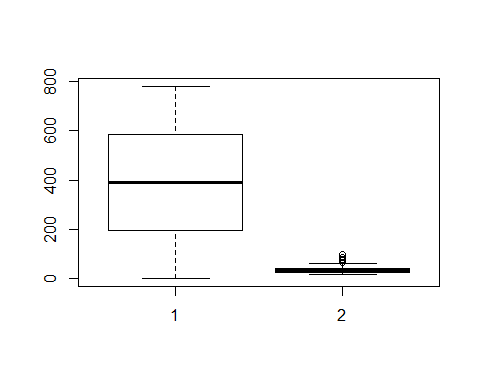
#==========univariate  
#univariate plots  
hist(Credit\_Training\_Data$`Months since Checking Acct opened`,col="red")#histogram of Months since checking Acct opened column of Credit\_Training\_Data dataset



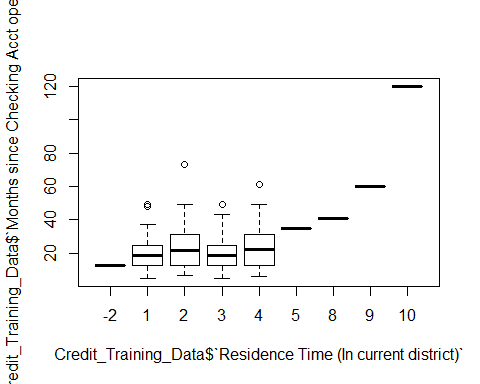
#histogram of Residence Time (In current district) column of Credit\_Training\_Data dataset  
hist(Credit\_Training\_Data$`Residence Time (In current district)`,col="Yellow")



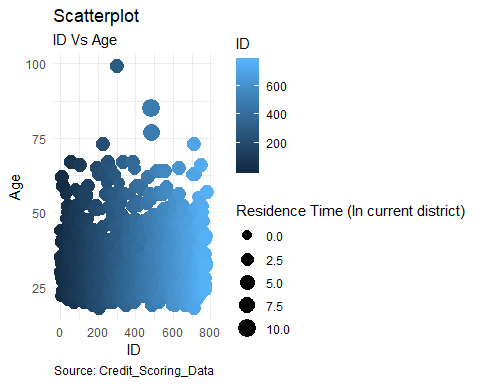
#==========bivariate  
#bivariate plots  
  
#boxplot of ID and Age columns of Credit\_Training\_Data dataset  
boxplot(Credit\_Training\_Data$ID,Credit\_Training\_Data$Age)



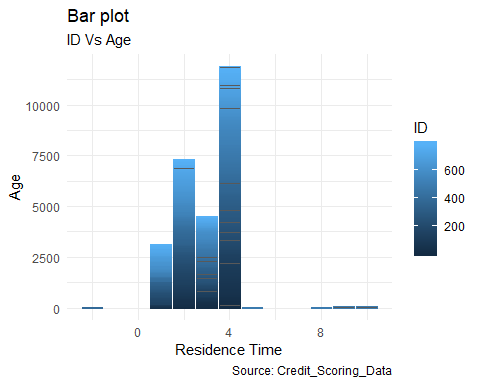
#boxplot of Months since Checking Acct opened and Residence Time (In current district) columns of Credit\_Training\_Data dataset  
boxplot(Credit\_Training\_Data$`Months since Checking Acct opened`~Credit\_Training\_Data$`Residence Time (In current district)`)



#=========trivariant   
#trivariant Analysis   
  
#plotting scatterplot between ID,Age, and Residence Time (In current district) using ggplot  
ggplot(Credit\_Training\_Data, aes(x=ID, y=Age)) + geom\_point(aes(col=ID,size=`Residence Time (In current district)`))+  
 theme\_minimal()+labs(subtitle="ID Vs Age", x="ID",y="Age",title="Scatterplot",caption = "Source: Credit\_Scoring\_Data")



#plotting barplot between Age,Residence Time (In current district) and ID using ggplot  
  
ggplot(Credit\_Training\_Data, aes(x=`Residence Time (In current district)`, y=Age)) + geom\_bar(stat="identity",aes(col=ID))+  
 theme\_minimal()+labs(subtitle="ID Vs Age", x="Residence Time",y="Age",title="Bar plot",caption = "Source: Credit\_Scoring\_Data")



#======proportion tables  
#1-D table  
#forming the table for Foreign National column of Credit\_training\_Data dataset and assigining to variable t1  
t1 <- table(Credit\_Training\_Data$`Foreign National`)  
t1#printing the t1 which gives the number of domains and its values

##   
## No Yes   
## 253 527

table(Credit\_Training\_Data$`Foreign National`)/nrow(Credit\_Training\_Data)#finding proportion table

##   
## No Yes   
## 0.324359 0.675641

#or  
prop.table(table(Credit\_Training\_Data$`Foreign National`)/nrow(Credit\_Training\_Data))#finding proportion table

##   
## No Yes   
## 0.324359 0.675641

# 2-D table  
#forming the table for Checking Acct and Credit History column of Credit\_training\_Data dataset and assigining to variable t2  
  
t2 <- table(Credit\_Training\_Data$`Checking Acct`,Credit\_Training\_Data$`Credit History`)  
t2#printing the t2 which gives the number of domains and its values

##   
## All Paid Bank Paid Critical Current Delay  
## 0Balance 28 30 38 157 10  
## High 6 5 8 22 4  
## Low 18 20 28 115 20  
## No Acct 86 8 18 129 30

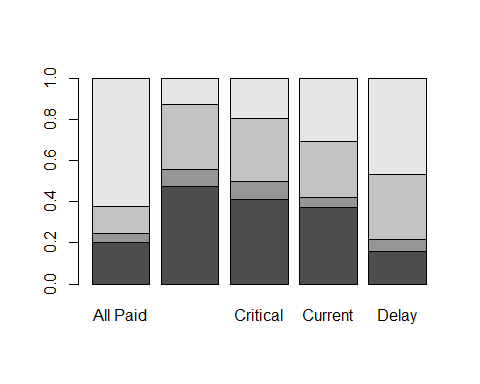
prop.table(t2, margin = 2)#printing lproportion table for t2

##   
## All Paid Bank Paid Critical Current Delay  
## 0Balance 0.20289855 0.47619048 0.41304348 0.37115839 0.15625000  
## High 0.04347826 0.07936508 0.08695652 0.05200946 0.06250000  
## Low 0.13043478 0.31746032 0.30434783 0.27186761 0.31250000  
## No Acct 0.62318841 0.12698413 0.19565217 0.30496454 0.46875000

#using round function to round the value to 2 decimal points and printing first few lines by using head() function for columns Checking Acct and Credit History by using margin=2   
head(round(prop.table(table(Credit\_Training\_Data$`Checking Acct`,Credit\_Training\_Data$`Credit History`),2),2))

##   
## All Paid Bank Paid Critical Current Delay  
## 0Balance 0.20 0.48 0.41 0.37 0.16  
## High 0.04 0.08 0.09 0.05 0.06  
## Low 0.13 0.32 0.30 0.27 0.31  
## No Acct 0.62 0.13 0.20 0.30 0.47

# Now making barplot for Checking Acct and Credit History columns  
barplot(prop.table(table(Credit\_Training\_Data$`Checking Acct`,Credit\_Training\_Data$`Credit History`), margin = 2))



# 3 way pivot table #  
#forming the table for Savings Acct,Job Type and Foreign National columns of Credit\_training\_Data dataset and assigining to variable t3  
  
t3 <- table(Credit\_Training\_Data$`Savings Acct`,Credit\_Training\_Data$`Job Type`,Credit\_Training\_Data$`Foreign National`)  
t3#printing the t3 which gives the number of domains and its values

## , , = No  
##   
##   
## Management Skilled Unemployed Unskilled  
## High 3 5 1 0  
## Low 24 106 3 30  
## MedHigh 1 10 0 7  
## MedLow 1 13 0 8  
## No Acct 8 27 1 5  
##   
## , , = Yes  
##   
##   
## Management Skilled Unemployed Unskilled  
## High 1 16 1 2  
## Low 39 211 11 76  
## MedHigh 3 22 2 4  
## MedLow 9 31 0 18  
## No Acct 12 57 0 12

#using round function to round the value to 2 decimal points by using margin=2 for columns Savings Acct,Job Type and Foreign National of Credit\_training\_Data dataset  
round(prop.table(t3, margin = 2),2)

## , , = No  
##   
##   
## Management Skilled Unemployed Unskilled  
## High 0.03 0.01 0.05 0.00  
## Low 0.24 0.21 0.16 0.19  
## MedHigh 0.01 0.02 0.00 0.04  
## MedLow 0.01 0.03 0.00 0.05  
## No Acct 0.08 0.05 0.05 0.03  
##   
## , , = Yes  
##   
##   
## Management Skilled Unemployed Unskilled  
## High 0.01 0.03 0.05 0.01  
## Low 0.39 0.42 0.58 0.47  
## MedHigh 0.03 0.04 0.11 0.02  
## MedLow 0.09 0.06 0.00 0.11  
## No Acct 0.12 0.11 0.00 0.07

#forming the table for Loan Reason,Personal Status and Housing columns of Credit\_training\_Data dataset and assigining to variable t3\_1  
  
t3\_1 <- table(Credit\_Training\_Data$`Loan Reason`,Credit\_Training\_Data$`Personal Status`,Credit\_Training\_Data$Housing)  
t3\_1#printing the t3\_1 which gives the number of domains and its values

## , , = Other  
##   
##   
## Divorced Married Single  
## Business 1 0 2  
## Car New 4 0 23  
## Car Used 4 0 24  
## Education 5 0 7  
## Furniture 0 0 7  
## Large Appliance 0 0 0  
## Other 2 0 1  
## Repairs 0 0 2  
## Retraining 0 0 0  
## Small Appliance 2 0 10  
##   
## , , = Own  
##   
##   
## Divorced Married Single  
## Business 13 8 37  
## Car New 45 12 68  
## Car Used 4 0 33  
## Education 15 2 7  
## Furniture 45 11 45  
## Large Appliance 2 2 4  
## Other 2 0 6  
## Repairs 10 0 6  
## Retraining 0 0 0  
## Small Appliance 36 20 96  
##   
## , , = Rent  
##   
##   
## Divorced Married Single  
## Business 11 0 10  
## Car New 18 2 19  
## Car Used 4 4 2  
## Education 4 1 4  
## Furniture 35 0 17  
## Large Appliance 0 0 0  
## Other 0 0 0  
## Repairs 0 0 2  
## Retraining 0 2 0  
## Small Appliance 11 6 5

#using round function to round the value to 2 decimal points by using margin=1 for Loan Reason,Personal Status and Housing columns of Credit\_training\_Data dataset  
round(prop.table(t3\_1,margin = 1),2)

## , , = Other  
##   
##   
## Divorced Married Single  
## Business 0.01 0.00 0.02  
## Car New 0.02 0.00 0.12  
## Car Used 0.05 0.00 0.32  
## Education 0.11 0.00 0.16  
## Furniture 0.00 0.00 0.04  
## Large Appliance 0.00 0.00 0.00  
## Other 0.18 0.00 0.09  
## Repairs 0.00 0.00 0.10  
## Retraining 0.00 0.00 0.00  
## Small Appliance 0.01 0.00 0.05  
##   
## , , = Own  
##   
##   
## Divorced Married Single  
## Business 0.16 0.10 0.45  
## Car New 0.24 0.06 0.36  
## Car Used 0.05 0.00 0.44  
## Education 0.33 0.04 0.16  
## Furniture 0.28 0.07 0.28  
## Large Appliance 0.25 0.25 0.50  
## Other 0.18 0.00 0.55  
## Repairs 0.50 0.00 0.30  
## Retraining 0.00 0.00 0.00  
## Small Appliance 0.19 0.11 0.52  
##   
## , , = Rent  
##   
##   
## Divorced Married Single  
## Business 0.13 0.00 0.12  
## Car New 0.09 0.01 0.10  
## Car Used 0.05 0.05 0.03  
## Education 0.09 0.02 0.09  
## Furniture 0.22 0.00 0.11  
## Large Appliance 0.00 0.00 0.00  
## Other 0.00 0.00 0.00  
## Repairs 0.00 0.00 0.10  
## Retraining 0.00 1.00 0.00  
## Small Appliance 0.06 0.03 0.03

# 3 way pivot table - better to use ftable for proportions   
#forming the ftable(frequency table to print the result more clearly and attractively) for Savings Acct,Job Type and Foreign National columns of Credit\_training\_Data dataset and assigining to variable t3\_2  
  
t3\_2 <- ftable(Credit\_Training\_Data$`Savings Acct`,Credit\_Training\_Data$`Job Type`,Credit\_Training\_Data$`Foreign National`)  
t3\_2#printing the t3\_2 which gives the number of domains and its values

## No Yes  
##   
## High Management 3 1  
## Skilled 5 16  
## Unemployed 1 1  
## Unskilled 0 2  
## Low Management 24 39  
## Skilled 106 211  
## Unemployed 3 11  
## Unskilled 30 76  
## MedHigh Management 1 3  
## Skilled 10 22  
## Unemployed 0 2  
## Unskilled 7 4  
## MedLow Management 1 9  
## Skilled 13 31  
## Unemployed 0 0  
## Unskilled 8 18  
## No Acct Management 8 12  
## Skilled 27 57  
## Unemployed 1 0  
## Unskilled 5 12

#using round function to round the value to 2 decimal points by using margin=2 for columns Savings Acct,Job Type and Foreign National of Credit\_training\_Data dataset  
round(prop.table(t3\_2,margin = 2),2)#check this

## No Yes  
##   
## High Management 0.01 0.00  
## Skilled 0.02 0.03  
## Unemployed 0.00 0.00  
## Unskilled 0.00 0.00  
## Low Management 0.09 0.07  
## Skilled 0.42 0.40  
## Unemployed 0.01 0.02  
## Unskilled 0.12 0.14  
## MedHigh Management 0.00 0.01  
## Skilled 0.04 0.04  
## Unemployed 0.00 0.00  
## Unskilled 0.03 0.01  
## MedLow Management 0.00 0.02  
## Skilled 0.05 0.06  
## Unemployed 0.00 0.00  
## Unskilled 0.03 0.03  
## No Acct Management 0.03 0.02  
## Skilled 0.11 0.11  
## Unemployed 0.00 0.00  
## Unskilled 0.02 0.02

#forming the ftable(frequency table to print the result more clearly and attractively) for Loan Reason,Personal Status and Housing columns of Credit\_training\_Data dataset and assigining to variable t3\_3  
  
t3\_3 <- ftable(Credit\_Training\_Data$`Loan Reason`,Credit\_Training\_Data$`Personal Status`,Credit\_Training\_Data$Housing)  
t3\_3#printing the t3\_3 which gives the number of domains and its values

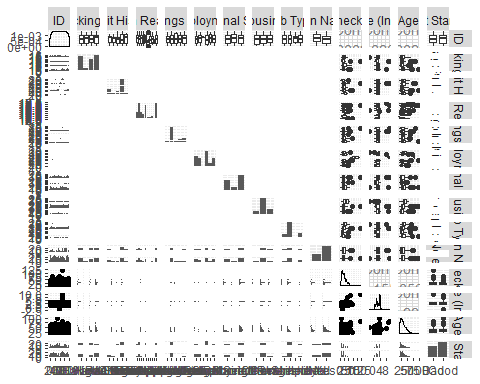
## Other Own Rent  
##   
## Business Divorced 1 13 11  
## Married 0 8 0  
## Single 2 37 10  
## Car New Divorced 4 45 18  
## Married 0 12 2  
## Single 23 68 19  
## Car Used Divorced 4 4 4  
## Married 0 0 4  
## Single 24 33 2  
## Education Divorced 5 15 4  
## Married 0 2 1  
## Single 7 7 4  
## Furniture Divorced 0 45 35  
## Married 0 11 0  
## Single 7 45 17  
## Large Appliance Divorced 0 2 0  
## Married 0 2 0  
## Single 0 4 0  
## Other Divorced 2 2 0  
## Married 0 0 0  
## Single 1 6 0  
## Repairs Divorced 0 10 0  
## Married 0 0 0  
## Single 2 6 2  
## Retraining Divorced 0 0 0  
## Married 0 0 2  
## Single 0 0 0  
## Small Appliance Divorced 2 36 11  
## Married 0 20 6  
## Single 10 96 5

#using round function to round the value to 2 decimal points by using margin=2 for Loan Reason,Personal Status and Housing columns of Credit\_training\_Data dataset  
round(prop.table(t3\_3,margin = 2),2)

## Other Own Rent  
##   
## Business Divorced 0.01 0.02 0.07  
## Married 0.00 0.02 0.00  
## Single 0.02 0.07 0.06  
## Car New Divorced 0.04 0.09 0.11  
## Married 0.00 0.02 0.01  
## Single 0.24 0.13 0.12  
## Car Used Divorced 0.04 0.01 0.03  
## Married 0.00 0.00 0.03  
## Single 0.26 0.06 0.01  
## Education Divorced 0.05 0.03 0.03  
## Married 0.00 0.00 0.01  
## Single 0.07 0.01 0.03  
## Furniture Divorced 0.00 0.09 0.22  
## Married 0.00 0.02 0.00  
## Single 0.07 0.09 0.11  
## Large Appliance Divorced 0.00 0.00 0.00  
## Married 0.00 0.00 0.00  
## Single 0.00 0.01 0.00  
## Other Divorced 0.02 0.00 0.00  
## Married 0.00 0.00 0.00  
## Single 0.01 0.01 0.00  
## Repairs Divorced 0.00 0.02 0.00  
## Married 0.00 0.00 0.00  
## Single 0.02 0.01 0.01  
## Retraining Divorced 0.00 0.00 0.00  
## Married 0.00 0.00 0.01  
## Single 0.00 0.00 0.00  
## Small Appliance Divorced 0.02 0.07 0.07  
## Married 0.00 0.04 0.04  
## Single 0.11 0.18 0.03

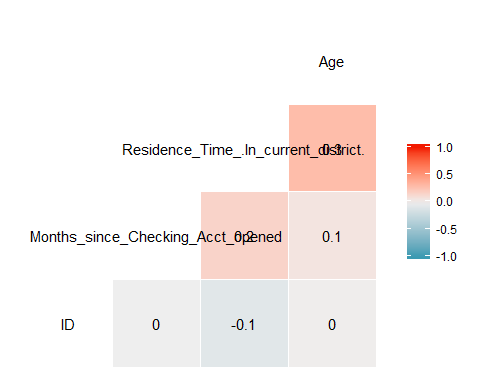
#ggpairs  
#The ggpairs() function produces a matrix of scatter plots for visualizing the correlation between variables  
ggpairs(Credit\_Training\_Data)#Make a matrix of plots with Credit\_Training\_Data data set

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



##ggcorrelation  
#The ggcorr() function draws a correlation matrix plot using ggplot2.  
ggcorr(Credit\_Training\_Data, palette = "RdBu", label = TRUE)#Makes a correlation matrix plot for Credit\_Training\_Data

## Warning in ggcorr(Credit\_Training\_Data, palette = "RdBu", label = TRUE):  
## data in column(s) 'Checking Acct', 'Credit History', 'Loan Reason',  
## 'Savings Acct', 'Employment', 'Personal Status', 'Housing', 'Job Type',  
## 'Foreign National', 'Credit Standing' are not numeric and were ignored



Initially, I loaded both the sheets by using the read\_excel() function which is available in the readxl package.I Read the scoring\_Data of Credit\_Risk6\_final.xlsx file by using read\_excel() function and it is assigned to Credit\_Risk6\_final1 and Generated the dataframe for Credit\_Risk6\_final1 using as.data.frame() function and that dataframe is named as Credit\_Scoring\_Data.

And Read the Training\_Data of Credit\_Risk6\_final.xlsx file by using read\_excel() function and it is assigned to Credit\_Risk6\_final2.Generated the dataframe for Credit\_Risk6\_final2 using as.data.frame() function and that dataframe is named as Credit\_Training\_Data.

I used some functions from the data explore package which will give some idea of a given dataset. Firstly I used the introduce() function will give the outline of the data it tells about the number of rows, columns, missing values, discrete columns, continuous columns, all missing columns, complete rows, total observations, and memory usage.

And then plot\_intro() will plot a graph that will give the percentage of discrete columns, continuous columns, All missing columns, complete rows, and missing observations. The plot\_missing() function shows the percentage of missing values of each column present in the dataset.

After that, I replaced the missing values with the mode because most of the data is categorical so mode will be the best option to replace the missing values(NA).

Later plot\_histogram() is provided by the DataExplorer package which will provide histograms of all columns which are containing the continuous data, from the histogram, I observed that the Months since Checking Acct opened gradually increased to a certain point and it started falling down from a certain point. Residence Time (In current district) column is increasing and decreasing it is not following a particular pattern.

plot\_bar() is provided by the DataExplorer package which will provide a bar chart for discrete data. For Checking Acct the No Account is the highest among all the other categories, the Credit History has the highest percentage the current, for the loan reason car new stands at first place and Retraining stands at the last place. For Savings Acct low category is in the first place and for Employment, most of the people stand at short employment. The personal status most of the people are single and divorced comes in second place. For the housing, job type and foreign nation own, skilled, yes categories have major importance.

Later the qq plot i.e Quantile-Quantile plot is used to visualize the deviation from a specific probability distribution.plot\_qq for ID, AGE, MONTHS SINCE CHECKING ACCT OPENED, RESIDENCE TIME all these columns. And then I performed univariate, bivariate and trivariate analysis and some pivot tables. After that by using the GGally package and I found the ggcorr () function draws a correlation matrix plot using ggplot2 and ggpairs() function produces a matrix of scatter plots for visualizing the correlation between variables.

There are no specific patterns within the dataset or any unusual things which are very important in the data set.

For building a model to the decision tree I used the rpart package. Initially, I will be making two samples by dividing the rows into 80% training data and 20% testing data probability. I will be taking the training data and train the model later for prediction the model I will be considering the testing data and predicting the data. Finally I will find the confusion matrix to know the accuracy of the model.

### QUESTION(b)

Build a decision tree model and give your decision tree, detailing its parameters. Explain how you decided on/fined tuned these parameters. (Include an image of your tree as well as a text output description.). Use set.seed(abc) where abc are the last 3 digits of your student no. Use this set.seed for all other model building below.

#The seed number is the starting point used in the generation of a sequence of random numbers, and the same results will be given if the same seed number is used.  
set.seed(214)#setting the seed with the last three digits of my student number  
#here I am making two samples by dividing the rows of Credit\_Training\_Data into 80% and 20% probability and storing in to variable id  
id<-sample(2,nrow(Credit\_Training\_Data),prob=c(0.8,0.2),replace=TRUE)  
  
#I am creating training data with the first sample(80%) i.e id==1 and storing that in to credit\_train  
Credit\_train=Credit\_Training\_Data[id==1,]  
#View(Credit\_train)  
nrow(Credit\_train)#here by using nrow we can see how many rows the credit\_train is taking

## [1] 633

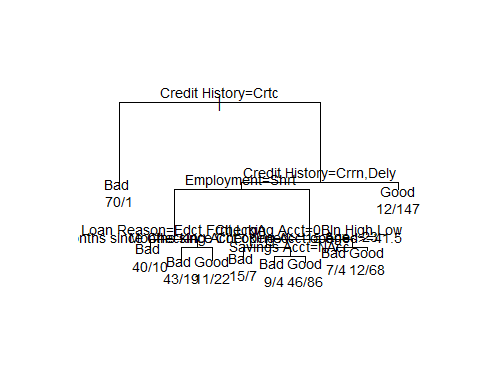
#after execution it show it is taking 633 rows for training the model  
  
#I am creating testing data with the second sample(20%) i.e id==2 and storing that in to credit\_test  
  
Credit\_test=Credit\_Training\_Data[id==2,]  
#View(Credit\_test)  
#here by using nrow we can see how many rows the credit\_train is taking  
nrow(Credit\_test)

## [1] 147

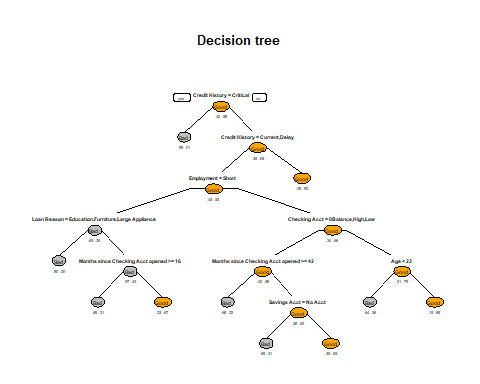
#after execution it show it is taking 147 rows for testing the model  
  
#now creating the model by using the credit standing column with the data credit\_train i.e training data(we will train the model by using this training data)  
#here i am using rpart package for creating the decision tree  
#the model is stored in the variable Credit\_model  
Credit\_model<-rpart(`Credit Standing`~.,data=Credit\_train)  
Credit\_model#viewing the model

## n= 633   
##   
## node), split, n, loss, yval, (yprob)  
## \* denotes terminal node  
##   
## 1) root 633 265 Good (0.41864139 0.58135861)   
## 2) Credit History=Critical 71 1 Bad (0.98591549 0.01408451) \*  
## 3) Credit History=All Paid,Bank Paid,Current,Delay 562 195 Good (0.34697509 0.65302491)   
## 6) Credit History=Current,Delay 403 183 Good (0.45409429 0.54590571)   
## 12) Employment=Short 145 51 Bad (0.64827586 0.35172414)   
## 24) Loan Reason=Education,Furniture,Large Appliance 50 10 Bad (0.80000000 0.20000000) \*  
## 25) Loan Reason=Business,Car New,Car Used,Repairs,Small Appliance 95 41 Bad (0.56842105 0.43157895)   
## 50) Months since Checking Acct opened>=15.5 62 19 Bad (0.69354839 0.30645161) \*  
## 51) Months since Checking Acct opened< 15.5 33 11 Good (0.33333333 0.66666667) \*  
## 13) Employment=Long,Medium,Unemployed,Very Short 258 89 Good (0.34496124 0.65503876)   
## 26) Checking Acct=0Balance,High,Low 167 70 Good (0.41916168 0.58083832)   
## 52) Months since Checking Acct opened>=41.5 22 7 Bad (0.68181818 0.31818182) \*  
## 53) Months since Checking Acct opened< 41.5 145 55 Good (0.37931034 0.62068966)   
## 106) Savings Acct=No Acct 13 4 Bad (0.69230769 0.30769231) \*  
## 107) Savings Acct=High,Low,MedHigh,MedLow 132 46 Good (0.34848485 0.65151515) \*  
## 27) Checking Acct=No Acct 91 19 Good (0.20879121 0.79120879)   
## 54) Age< 23 11 4 Bad (0.63636364 0.36363636) \*  
## 55) Age>=23 80 12 Good (0.15000000 0.85000000) \*  
## 7) Credit History=All Paid,Bank Paid 159 12 Good (0.07547170 0.92452830) \*

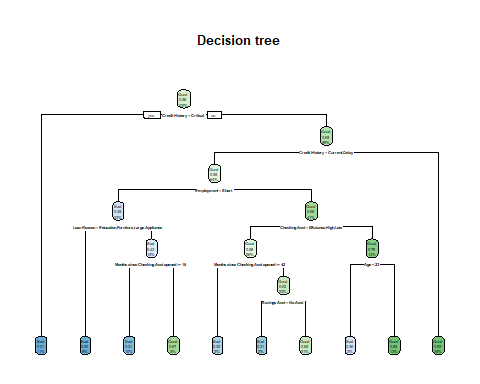
plot(Credit\_model,margin=0.1)#plotting the Credit\_model,the plot will give the outline of the decision treei.e the graphical view of tree   
text(Credit\_model,use.n=TRUE,pretty=TRUE,cex=0.9)#the graphical view of the tree is filled with the text by using the text() function



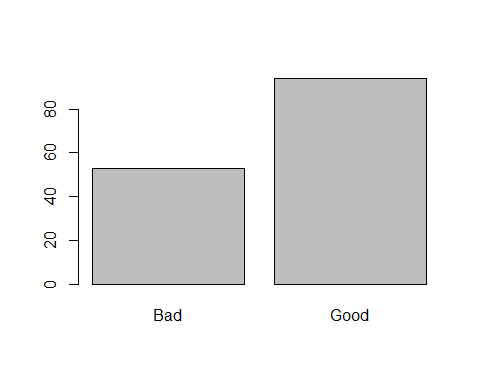
#prp-Plot An Rpart Model.  
#The prp function plots rpart trees. It automatically scales and adjusts the displayed tree for best fit.   
#This function is in the rpart.plot R package.   
#plotting the credit\_model using prp() function  
prp(Credit\_model,box.col=c("Grey", "Orange")[Credit\_model$frame$yval],varlen=0,faclen=0, type=1,extra=4,under=TRUE,main="Decision tree")



# rpart.plot is also automatically scales and adjusts the displayed tree for best fit  
#plotting the credit\_model again by using rpart.plot() function and this function is also available in rpart.plot  
rpart.plot(Credit\_model,main="Decision tree")



#NOw predicting the model by using test data i.e by using credit\_test(we will always predict the values by using testing data)  
#Predicted values are stored in pred\_Credit variable  
pred\_Credit<-predict(Credit\_model,newdata=Credit\_test,type="class")  
plot(pred\_Credit)#I am plotting the predicted values to see the ratio of bad and good



#Now creating the table for predicted values and actual test values i.e pred\_Credit and Credit\_test$`Credit Standing`  
table(pred\_Credit,Credit\_test$`Credit Standing`)

##   
## pred\_Credit Bad Good  
## Bad 34 19  
## Good 20 74

#now finding the confusion matrix by using confusionMatrix() function which is available in caret package.  
##generally confusion matrix is the sum of true positive and true negatives divide by sum of all the values in the table.  
#finding confusion matrix for pred\_Credit and Credit\_test$`Credit Standing`  
confusionMatrix(table(pred\_Credit,Credit\_test$`Credit Standing`))

## Confusion Matrix and Statistics  
##   
##   
## pred\_Credit Bad Good  
## Bad 34 19  
## Good 20 74  
##   
## Accuracy : 0.7347   
## 95% CI : (0.6556, 0.804)  
## No Information Rate : 0.6327   
## P-Value [Acc > NIR] : 0.005698   
##   
## Kappa : 0.427   
##   
## Mcnemar's Test P-Value : 1.000000   
##   
## Sensitivity : 0.6296   
## Specificity : 0.7957   
## Pos Pred Value : 0.6415   
## Neg Pred Value : 0.7872   
## Prevalence : 0.3673   
## Detection Rate : 0.2313   
## Detection Prevalence : 0.3605   
## Balanced Accuracy : 0.7127   
##   
## 'Positive' Class : Bad   
##

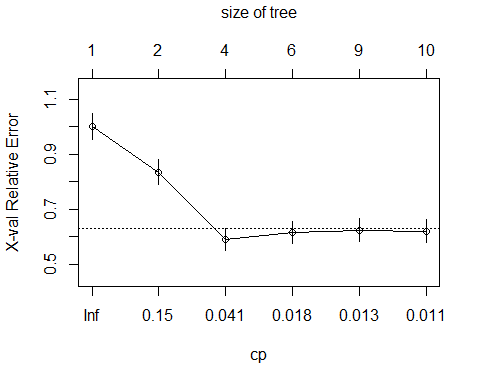
#Now pruning the tree generally it is called cross validation  
  
set.seed(214)#setting the seed with the last three digits of my student number  
  
#To validate the model we use the printcp functions. ‘CP’ stands for Complexity Parameter of the tree.  
printcp(Credit\_model)#finding the complexity parameter of the credit\_model

##   
## Classification tree:  
## rpart(formula = `Credit Standing` ~ ., data = Credit\_train)  
##   
## Variables actually used in tree construction:  
## [1] Age Checking Acct   
## [3] Credit History Employment   
## [5] Loan Reason Months since Checking Acct opened  
## [7] Savings Acct   
##   
## Root node error: 265/633 = 0.41864  
##   
## n= 633   
##   
## CP nsplit rel error xerror xstd  
## 1 0.260377 0 1.00000 1.00000 0.046838  
## 2 0.081132 1 0.73962 0.83396 0.045258  
## 3 0.020755 3 0.57736 0.58868 0.040914  
## 4 0.015094 5 0.53585 0.61509 0.041514  
## 5 0.011321 8 0.48679 0.62264 0.041679  
## 6 0.010000 9 0.47547 0.61887 0.041597

#We prune the tree to avoid any overfitting of the data.  
#The final result is to have a small tree and the one with least cross validated error given by printcp() function i.e. ‘xerror’.  
#From the above printcp(), we can select the one value which have least cross-validated error and use it to prune the tree.  
Credit\_model$cptable[which.min(Credit\_model$cptable[,"xerror"]),"CP"]#This function returns the optimal cp value associated with the minimum error.

## [1] 0.02075472

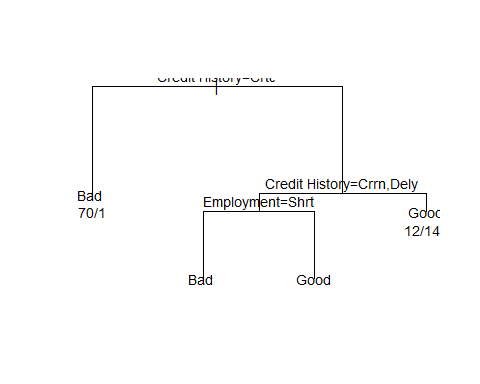
#Plotcp() provides a graphical representation to the cross validated error summary. The cp values are plotted against the geometric mean to depict the deviation until the minimum value is reached.  
plotcp(Credit\_model)#plotting for the model



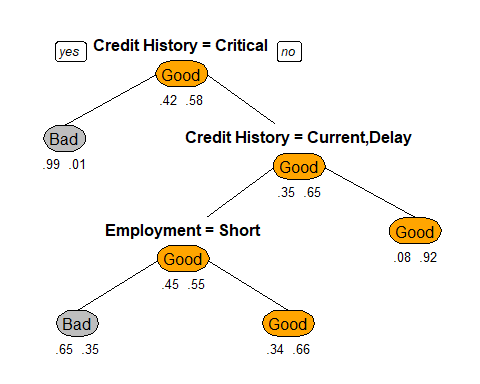
#now pruning the tree by using prune() and storing that in ptree variable  
ptree<- prune(Credit\_model,  
 cp= Credit\_model$cptable[which.min(Credit\_model$cptable[,"xerror"]),"CP"])  
ptree#printing the prune tree

## n= 633   
##   
## node), split, n, loss, yval, (yprob)  
## \* denotes terminal node  
##   
## 1) root 633 265 Good (0.41864139 0.58135861)   
## 2) Credit History=Critical 71 1 Bad (0.98591549 0.01408451) \*  
## 3) Credit History=All Paid,Bank Paid,Current,Delay 562 195 Good (0.34697509 0.65302491)   
## 6) Credit History=Current,Delay 403 183 Good (0.45409429 0.54590571)   
## 12) Employment=Short 145 51 Bad (0.64827586 0.35172414) \*  
## 13) Employment=Long,Medium,Unemployed,Very Short 258 89 Good (0.34496124 0.65503876) \*  
## 7) Credit History=All Paid,Bank Paid 159 12 Good (0.07547170 0.92452830) \*

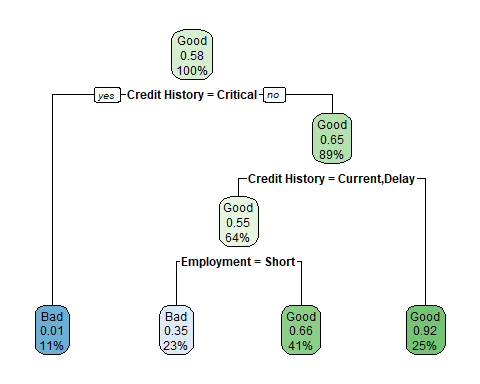
#plotting the ptree,the plot will give the outline of the decision treei.e the graphical view of tree   
plot(ptree)  
  
#the graphical view of the tree is filled with the text by using the text() function  
text(ptree,use.n=TRUE,pretty=TRUE,cex=0.9)



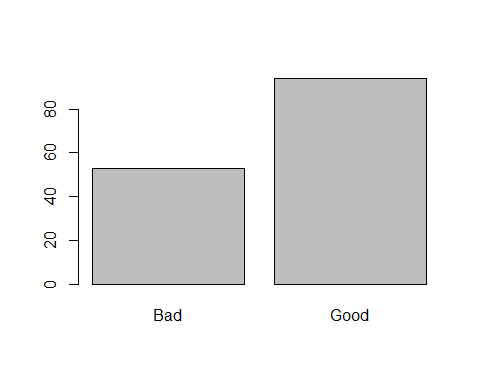
#prp-Plot An Rpart Model.  
#The prp function plots rpart trees. It automatically scales and adjusts the displayed tree for best fit.   
#This function is in the rpart.plot R package.   
#plotting the pruned tree ptree using prp() function  
prp(ptree,box.col=c("Grey", "Orange")[ptree$frame$yval],varlen=0,faclen=0, type=1,extra=4,under=TRUE)



# rpart.plot is also automatically scales and adjusts the displayed tree for best fit  
#plotting the ptree again by using rpart.plot() function and this function is also available in rpart.plot  
  
rpart.plot(ptree)



#NOw predicting the model by using test data i.e by using credit\_test(we will always predict the values by using testing data)  
#Predicted values are stored in tree.pred variable  
  
tree.pred<-predict(ptree,Credit\_test,type="class")  
plot(tree.pred)#I am plotting the predicted values to see the ratio of bad and good



#Now creating the table for predicted values and actual test values i.e tree.pred and Credit\_test$`Credit Standing`  
  
table(tree.pred,Credit\_test$`Credit Standing`)

##   
## tree.pred Bad Good  
## Bad 35 18  
## Good 19 75

#finding confusion matrix for tree.pred and Credit\_test$`Credit Standing`  
  
confusionMatrix(table(tree.pred,Credit\_test$`Credit Standing`))

## Confusion Matrix and Statistics  
##   
##   
## tree.pred Bad Good  
## Bad 35 18  
## Good 19 75  
##   
## Accuracy : 0.7483   
## 95% CI : (0.6701, 0.8162)  
## No Information Rate : 0.6327   
## P-Value [Acc > NIR] : 0.001913   
##   
## Kappa : 0.4564   
##   
## Mcnemar's Test P-Value : 1.000000   
##   
## Sensitivity : 0.6481   
## Specificity : 0.8065   
## Pos Pred Value : 0.6604   
## Neg Pred Value : 0.7979   
## Prevalence : 0.3673   
## Detection Rate : 0.2381   
## Detection Prevalence : 0.3605   
## Balanced Accuracy : 0.7273   
##   
## 'Positive' Class : Bad   
##

I used the rpart package to build a decision tree

1.Initially, I set seed value with the last three digits of my student number i.e 214. The seed number is the starting point used in the generation of a sequence of random numbers, and the same results will be given if the same seed number is used.

2.I am making two samples by dividing the rows of Credit\_Training\_Data into 80% and 20% probability and storing it in variable id.

3.I am creating training data with the first sample(80%) i.e id==1 and storing that into credit\_train. After that, I am using nrow which will show how many rows the credit\_train is taking.

4.I am creating testing data with the second sample(20%) i.e id==2 and storing that into credit\_test and using now which will show how many rows the credit\_test is taking.

5.Now creating the model by using the credit standing column with the data credit\_train i.e training data(we will train the model by using this training data).The model is stored in the variable Credit\_model.

6.Plotting the Credit\_model, the plot will give the outline of the decision tree i.e the graphical view of a tree, the graphical view of the tree is filled with the text by using the text() function. Now plotting the graph very attractively by using prp (Plot An Rpart Model). The prp() function plots rpart trees, It automatically scales and adjusts the displayed tree for best fit. This function is in the rpart. plot R package. The rpart.plot is also automatically scaled and adjusts the displayed tree for best fit, now plotting the model again by using rpart.plot() function and this function is also available in rpart.plot

7.Later predicting the model by using test data i.e by using credit\_test(we will always predict the values by using testing data) and Predicted values are stored in pred\_Credit variable

8.And then creating the table for predicted values and actual test values i.e pred\_Credit and Credit\_test$Credit Standing.

9.To find the confusion matrix I used the confusionMatrix() function which is available in the caret package. Generally confusion matrix is the sum of true positives and true negatives divided by the sum of all the values in the table.

After finding the confusion matrix for pred\_Credit and Credit\_test$Credit Standing accuracy of the model is “73.47%”

Pruning the tree generally is called cross-validation

The process for pruning the tree is as follows:

1.set the seed with the last three digits of my student number

1. To validate the model we use the printcp functions. ‘CP’ stands for Complexity Parameter of the tree. Finding the complexity parameter of the credit\_model
2. We prune the tree to avoid any overfitting of the data. The final result is to have a small tree and the one with least cross validated error given by printcp() function i.e. ‘xerror’.From the above printcp(), we can select the one value which has the least cross-validated error and use it to prune the tree.

4.Plotcp() provides a graphical representation of the cross validated error summary. The cp values are plotted against the geometric mean to depict the deviation until the minimum value is reached. Now pruning the tree by using prune() and storing that in ptree variable.

1. Plot the tree and fill the plot with the text, to make the decision tree attractive use PRP() and rpart.plot() functions and make the good attractive trees.

6.NOw predicting the model by using test data i.e by using credit\_test(we will always predict the values by using testing data)and store it in a tree. pred variable, after that, create the table for predicted values and actual test values i.e tree.pred and Credit\_test$Credit Standing and find the confusion matrix the accuracy obtained for pruned decision tree is “74.83%”

After pruning the accuracy increases which means that the pruned decision tree model generalizes well and is more suitable. Pruning is a technique in machine learning and search algorithms that reduces the size of decision trees by removing sections of the tree that provide little power to classify instances. Pruning reduces the complexity of the final classifier, and hence improves predictive accuracy by the reduction of overfitting. This the reason behind the increase in the accuracy from 73.47 to 74.83

### QUESTION(c)

Use the decision tree to predict results for the scoring set. Choose 5 different potential loan clients and explain to Kate in plain English how the decision tree works (15 marks) and how the accuracy/probabilities of these being a good/bad loan was calculated by the decision tree, outling your assumptions (5 marks). Max word count 500 words.

#renaming the Residence Time column to Residence Time (In current district) in Credit\_scoring\_data dataset because the same column have different names in both the sheets in a file  
  
#Inorder to predict the values all the column names should be same in both sheets  
  
Credit\_Scoring\_Data= Credit\_Scoring\_Data%>% rename(`Residence Time (In current district)` = `Residence Time`)  
  
colnames(Credit\_Scoring\_Data)#Prints all the column names present in credit\_scoring\_data dataset

## [1] "ID"   
## [2] "Checking Acct"   
## [3] "Credit History"   
## [4] "Loan Reason"   
## [5] "Savings Acct"   
## [6] "Employment"   
## [7] "Personal Status"   
## [8] "Housing"   
## [9] "Job Type"   
## [10] "Foreign National"   
## [11] "Months since Checking Acct opened"   
## [12] "Residence Time (In current district)"  
## [13] "Age"

set.seed(214)#setting the seed with the last three digits of my student number  
  
#predicting the values by using the tree predict values with the Credit\_scoring\_data values  
  
pred\_Credit\_score<-predict(Credit\_model,newdata=Credit\_Scoring\_Data,type="class")  
  
pred\_Credit\_score#printing the predicted values of credit\_scoring\_data

## 1 2 3 4 5 6 7 8 9 10 11 12 13   
## Good Good Good Good Good Bad Good Good Good Good Bad Bad Bad   
## Levels: Bad Good

In the question, it is given that use the decision tree to predict the results for the scoring\_data i.e Credit\_Scoring\_Data.

So let me describe the decision tree and how it works first

Initially, I made two samples by dividing the rows into 80% training data and 20% testing data probability. I considered the training data and train the model, later for prediction the model I will be considering the testing data and predicting the data. Finally, I will find the confusion matrix to know the accuracy of the model. This is how I modeled the decision tree

Now let me explain the root node, parent node and child nodes of decision tree

The root node of my decision tree is Credit History.

1. If Credit History is equal to critical it will follow the left-hand side of the root node which will give the result ”BAD”.If Credit History is not equal to critical it will follow the right side path of the root node.

2.on On the right side of the root node if the credit history is not equal to current or Delay it will follow the right path of credit History which gives the result” GOOD”. If credit history is equal to current or Delay I will follow the left path there one condition occurs if Employment is equal to short It will follow the left path where loan reason is equal to Education,Furniture or large appliances the result is “GOOD” otherwise it will follow right path where Months since checking Acct opened greater than or equal to 15.5 result is “BAD” else the result is “GOOD” If Employment is not equal to short it will follow the right path where one condition occurs i.e checking Acct equal to zero balance, high, low it will follow left path where Months since checking Acct opened greater than equal to 41.5 results is “BAD” else if savings Acct equal to no account result is “BAD” otherwise “GOOD”. If checking Acct is not equal to zero balance, high, low it follows the right path where the condition occurs Age less than 23 the result is “BAD” otherwise “GOOD”

This is how my decision tree works.

Now I will predict the values for Scoring\_data by taking the decision tree

Firstly I renamed the Residence Time column to Residence Time (In current district) in Credit\_scoring\_data dataset because the same column has different names in both the sheets in a file. Inorder to predict the values all the column names should be the same in both sheets. predicting the values by using the tree predict values with the Credit\_scoring\_data values

In question, it is given that Choose 5 different potential loan clients and explain to Kate in plain English how the decision tree works

So I will be considering the ID’s randomly i.e considering IDs 782,786,788,790,793.

Now I am going to explain how the decision tree works for the above-selected values.

For ID 782:

The credit History value is current so it will not satisfy the root node condition it will follow the right path of root node where there is a condition if Credit History is equal to Current and Delay so I will follow left path where Employment is equal to short or not, no Id 782 have employment is Medium so it will follow the right path where checking Acct is equal to 0 balance,high,low the condition is satisfied here ID 782 have checking Acct is low so left path is followed where condition occurs i.e Months since Checking Acct opened >=41.5,No the condition is not satisfied so I will take right side where savings Acct=no Acct condition is not satisfied so the result is “GOOD”

For id 786:

The credit History value is current so it will not satisfy the root node condition it will follow the right path of root node where there is a condition if Credit History is equal to Current and Delay so I will follow left path where Employment is equal to short or not ,no Id 786 is unemployment so it will follow the right path where checking Acct is equal to 0 balance, high,low the condition is satisfied here ID 786 have checking Acct is low so left path is followed where condition occurs i.e Months since Checking Acct opened >=41.5,yes the condition is satisfied so the result is “BAD”

For id 788:

The credit History value is All Paid so it will not satisfy the root node condition it will follow the right path of root node where there is a condition if Credit History is equal to Current and Delay so I will follow the right which gives the result “GOOD”

For id 790:

The credit History value is current so it will not satisfy the root node condition it will follow the right path of root node where there is a condition if Credit History is equal to Current and Delay so I will follow left path where Employment is equal to short or not ,no Id 790 have employment Medium so it will follow the right path where checking Acct is equal to 0 balance,high,low the condition is not satisfied here ID 790 have checking Acct is No account so right path is followed where condition occurs i.e Age<23, No the condition is satisfied so the result is “GOOD”

For id 793:

The credit History value is current so it will not satisfy the root node condition it will follow the right path of root node where there is a condition if Credit History is equal to Current and Delay so I will follow left path where Employment is equal to short or not ,no Id 793 have short employment so it will follow the left path where Loan Reason is equal to Education, furniture, large appliances the condition is not satisfied here ID 793 have Loan Reason car New so it will take right path where condition occurs i.e e Months since Checking Acct opened >=15.5, yes the condition is satisfied so the result is “BAD”

This is how the decision works for the IDS 782,786,788,790,793.

The Good and Bad values are calculated based on the column Credit History, Employment, Loan Reason, Checking Acct, Months since checking Acct opened, saving Acct, Age.

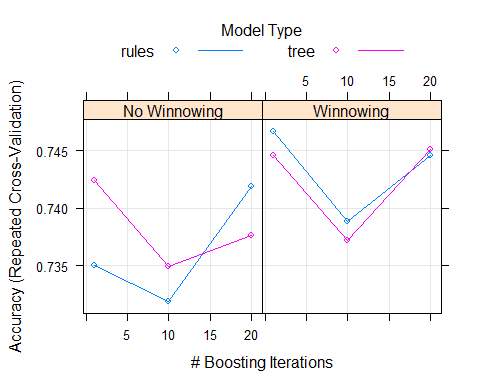
If the credit History –critical,current,Delay Employment–short Loan Reason–Education, Furniture, Large Appliances Checking Account–0 Balance, Low, High Months since checking Acct opened– >=15.5 and >=41.5 Saving Acc–No Account Age–<23

By considering the above columns and conditions only the good or bad loan Is decided. These conditions are satisfied and followed correctly we will get the good accuracy/probabilities of these being a good/bad loan was calculated by the decision tree correctly.so whatever the assumptions are mentioned above are followed we get the good accuracy of being good or bad loan is calculated

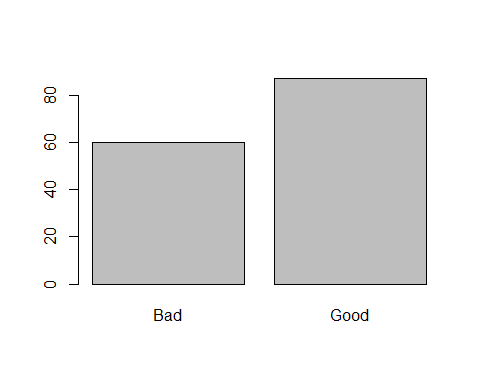
### QUESTION(d)

Now try and improve your model using 2 other approaches, e.g. ensemble technique, boosting or a different model. Explain your training/validation/testing methodology. Comment on your results and analyse why your model is giving better/worse results.

# Boosting Algorithms  
  
#trainControl is the computational quality that is not easy to notice but may be important to the train function  
  
#trainControl that allow us to perform variety of cross validation  
control <- trainControl(method="repeatedcv", number=10, repeats=3)  
  
set.seed(214)#setting the seed with the last three digits of my student number  
  
  
#now creating the model by using the credit standing column with the data credit\_train i.e training data(we will train the model by using this training data)  
#here i am using c50 package for boosting algorithm improving the decision tree model  
#the model is stored in the variable Boost\_model  
Boost\_model <- train(`Credit Standing`~., data=Credit\_train, method="C5.0", metric="Accuracy", trControl=control)  
  
plot(Boost\_model)#plotting the model



#NOw predicting the model by using test data i.e by using credit\_test(we will always predict the values by using testing data)  
  
#Predicting the values by using Boost\_model and Credit\_test data and storing in the variable predict\_boost  
  
predict\_boost=predict(Boost\_model,newdata=Credit\_test)  
  
plot(predict\_boost)#I am plotting the predicted values to see the ratio of bad and good



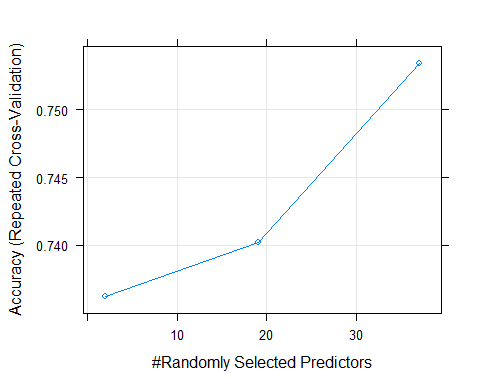
#Now creating the table for predicted values and actual test values i.e predict\_boost and Credit\_test$`Credit Standing`  
  
table(predict\_boost,Credit\_test$`Credit Standing`)

##   
## predict\_boost Bad Good  
## Bad 41 19  
## Good 13 74

#finding confusion matrix for predict\_boost and Credit\_test$`Credit Standing`  
  
confusionMatrix(table(predict\_boost,Credit\_test$`Credit Standing`))

## Confusion Matrix and Statistics  
##   
##   
## predict\_boost Bad Good  
## Bad 41 19  
## Good 13 74  
##   
## Accuracy : 0.7823   
## 95% CI : (0.7068, 0.8461)  
## No Information Rate : 0.6327   
## P-Value [Acc > NIR] : 6.843e-05   
##   
## Kappa : 0.5423   
##   
## Mcnemar's Test P-Value : 0.3768   
##   
## Sensitivity : 0.7593   
## Specificity : 0.7957   
## Pos Pred Value : 0.6833   
## Neg Pred Value : 0.8506   
## Prevalence : 0.3673   
## Detection Rate : 0.2789   
## Detection Prevalence : 0.4082   
## Balanced Accuracy : 0.7775   
##   
## 'Positive' Class : Bad   
##

#BAGGING Algorithm  
  
#trainControl the computational quality that is not easy to notice but may be important to the train function  
#trainControl that allow us to perform variety of cross validation  
  
control <- trainControl(method="repeatedcv", number=10, repeats=3)  
  
  
set.seed(214)#setting the seed with the last three digits of my student number  
  
#now creating the model by using the credit standing column with the data credit\_train i.e training data(we will train the model by using this training data)  
#here i am using random forest package for bagging algorithm improving the decision tree model  
#the model is stored in the variable Bagging\_model  
  
Bagging\_model <- train(`Credit Standing`~., data=Credit\_train, method="rf", metric="Accuracy", trControl=control)  
  
plot(Bagging\_model)#printing the Bagging\_model



#NOw predicting the model by using test data i.e by using credit\_test(we will always predict the values by using testing data)  
#Predicting the values by using Bagging\_model and Credit\_test data and storing in the variable Bagging\_Predict  
  
Bagging\_Predict=predict(Bagging\_model,newdata=Credit\_test)  
  
#Now creating the table for predicted values and actual test values i.e Bagging\_Predict and Credit\_test$`Credit Standing`  
  
table(Bagging\_Predict,Credit\_test$`Credit Standing`)

##   
## Bagging\_Predict Bad Good  
## Bad 41 20  
## Good 13 73

#finding confusion matrix for Bagging\_Predict and Credit\_test$`Credit Standing`  
  
confusionMatrix(table(Bagging\_Predict,Credit\_test$`Credit Standing`))

## Confusion Matrix and Statistics  
##   
##   
## Bagging\_Predict Bad Good  
## Bad 41 20  
## Good 13 73  
##   
## Accuracy : 0.7755   
## 95% CI : (0.6994, 0.8402)  
## No Information Rate : 0.6327   
## P-Value [Acc > NIR] : 0.0001432   
##   
## Kappa : 0.5298   
##   
## Mcnemar's Test P-Value : 0.2962699   
##   
## Sensitivity : 0.7593   
## Specificity : 0.7849   
## Pos Pred Value : 0.6721   
## Neg Pred Value : 0.8488   
## Prevalence : 0.3673   
## Detection Rate : 0.2789   
## Detection Prevalence : 0.4150   
## Balanced Accuracy : 0.7721   
##   
## 'Positive' Class : Bad   
##

Ensembles can give you a boost inaccuracy on your dataset. To increase accuracy on your dataset is to combine the predictions of multiple different models together. This is called an ensemble prediction.

Ensemble learning is using multiple learning algorithms at the same time to obtain prediction with an aim to have better predictions of the individual value.

We use ensemble learning because Better accuracy(low rate) High consistency avoids overfitting Reduces bias and variance errors

When and where to use ensemble learning -single model overfits -results worth the extra training -can be used for classifications as well as regression

1.Bagging or Bootstrap aggregation

It is an ensemble method where we use multiple models of same learning algorithm trained with subsets of dataset randomly picked from the training dataset

We select a subset of training dataset in to bags we do it randomly one point at a time and once a point is selected I cannot be removed from training data set so it is eligible to be selected again so fill the bag with subset of dataset and train the learning models and take the vote on their output. It is simple defined as Building multiple models (typically of the same type) from different subsamples of the training dataset.

The algorithm is implemented using random forest package

1. Initially I defined train control that allows us to perform a variety of cross-validation, trainControl is the computational quality that is not easy to notice but may be important to the train function.
2. Setting the seed with the last three digits of my student number

3 now creating the model by using the credit standing column with the data credit\_train i.e training data(we will train the model by using this training data). here i am using random forest package for bagging algorithm improving the decision tree model. And the model is stored in the variable Bagging\_model

4.NOw predicting the model by using test data i.e by using credit\_test(we will always predict the values by using testing data) Predicting the values by using Bagging\_model and Credit\_test data and storing in the variable Bagging\_Predict Now creating the table for predicted values and actual test values i.e Bagging\_Predict and Credit\_test$Credit Standing and find the confusion matrix which gives the accuracy “77.55%”

Boosting:

Incase of boosting we give more emphasis on selecting datasets or datapoint which give wrong predictions in order to improve the accuracy so we select the first sub data set in the same way as we did for bagging and train the model then test the trained model with training dataset and for each data point where prediction is wrong we put It in second data set along with randomly selected point from training dataset again train the model with new dataset and combine it with the previously trained model to form ensemble .we again test ensemble of two models on the training dataset and again select points which give the wrong prediction along with randomly selected points from training dataset and repeat the process .this results in good accuracy. It is simply defined as Building multiple models (typically of the same type) each of which learns to fix the prediction errors of a prior model in the chain.

The algorithm is implemented using the c50 package

1. Initially I defined train control that allows us to perform a variety of cross-validation, train control is the computational quality that is not easy to notice but may be important to the train function.
2. Setting the seed with the last three digits of my student number
3. now creating the model by using the credit standing column with the data credit\_train i.e training data(we will train the model by using this training data) here I am using c50 package for boosting algorithm improving the decision tree model, and the model is stored in the variable Boost\_model
4. NOw predicting the model by using test data i.e by using credit\_test(we will always predict the values by using testing data) Predicting the values by using Boost\_model and Credit\_test data and storing in the variable predict\_boost Now creating the table for predicted values and actual test values i.e predict\_boost and Credit\_test$Credit Standing and find the confusion matrix which gives the accuracy “78.23%”

Among both, the models Boosting Algorithm gives more accuracy when compared to Bagging Algorithm

Both are ensemble methods to get N learners from 1 learner, but, while they are built independently for Bagging, Boosting tries to add new models that do well where previous models fail.

Both generate several training data sets by random sampling, but only Boosting determines weights for the data to tip the scales in favor of the most difficult cases.

Both make the final decision by averaging the N learners (or taking the majority of them), but it is an equally-weighted average for Bagging and a weighted average for Boosting, more weight to those with better performance on training data.

Both are good at reducing variance and provide higher stability, but only Boosting tries to reduce bias. On the other hand, Bagging may solve the over-fitting problem, while Boosting can increase it.

### 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.

set.seed(214)#setting the seed with the last three digits of my student number  
  
class(pred\_Credit)#class of pred\_credit is factor

## [1] "factor"

class(Credit\_test$`Credit Standing`)#class of Credit\_test$`Credit Standing` is character

## [1] "character"

Credit\_test$`Credit Standing`<-as.factor(Credit\_test$`Credit Standing`)#so converting Credit\_test$`Credit Standing as factor  
class(Credit\_test$`Credit Standing`)#now the class of Credit\_test$`Credit Standing is factor

## [1] "factor"

#Making one list and placing both Pred\_credit and Credit\_test$`Credit Standing in to list  
lst <- list(  
 one = pred\_Credit,  
 two = Credit\_test$`Credit Standing`)  
lst#displaying the list

## $one  
## 2 5 7 19 30 40 45 46 51 53 57 68 73 74 75   
## Bad Good Good Good Good Bad Bad Bad Bad Good Good Good Good Good Good   
## 77 89 96 105 108 112 122 124 135 144 156 164 165 166 175   
## Good Bad Bad Good Good Bad Bad Bad Good Good Bad Bad Bad Good Bad   
## 180 188 197 198 200 203 219 229 232 237 238 241 245 249 270   
## Bad Bad Good Good Good Good Good Good Good Good Good Good Good Good Good   
## 278 282 283 285 296 301 303 315 325 329 343 345 353 355 357   
## Bad Bad Good Bad Bad Good Bad Good Good Bad Good Good Bad Good Good   
## 366 367 379 383 385 394 396 404 410 413 417 426 427 431 436   
## Good Bad Good Bad Good Bad Bad Bad Bad Good Good Good Good Good Good   
## 438 440 446 463 469 473 482 485 493 501 504 505 508 510 519   
## Bad Bad Bad Good Good Good Good Good Good Good Bad Good Good Good Good   
## 522 529 531 533 535 546 547 548 550 553 554 560 561 562 563   
## Good Good Good Good Good Good Good Good Good Good Good Good Good Good Good   
## 568 574 576 578 582 587 588 595 605 607 609 611 625 627 630   
## Good Bad Good Good Bad Good Bad Bad Bad Good Bad Good Good Bad Good   
## 647 650 652 658 669 671 676 677 678 681 685 699 701 702 707   
## Good Good Good Good Good Bad Bad Bad Bad Bad Good Bad Bad Good Bad   
## 712 713 719 737 742 747 749 750 753 764 777 779   
## Good Bad Good Bad Good Good Bad Good Good Good Bad Bad   
## Levels: Bad Good  
##   
## $two  
## [1] Bad Good Good Good Good Bad Bad Good Bad Good Good Good Good Good  
## [15] Good Good Good Good Good Good Bad Bad Bad Good Good Bad Bad Bad   
## [29] Good Bad Good Bad Good Bad Bad Bad Good Good Bad Good Good Bad   
## [43] Good Bad Good Bad Good Good Bad Good Good Bad Bad Bad Bad Good  
## [57] Good Good Good Good Bad Bad Bad Good Good Bad Bad Bad Bad Bad   
## [71] Bad Good Good Good Good Bad Bad Bad Good Good Good Good Good Good  
## [85] Good Good Good Good Good Good Good Good Good Good Good Good Good Good  
## [99] Good Good Good Good Good Bad Good Bad Good Bad Bad Bad Good Bad   
## [113] Good Bad Good Bad Good Bad Bad Bad Good Good Good Good Good Good  
## [127] Bad Bad Bad Good Bad Good Good Bad Good Good Good Good Good Good  
## [141] Good Good Good Good Good Bad Bad   
## Levels: Bad Good

lst[1]#displaying list 1

## $one  
## 2 5 7 19 30 40 45 46 51 53 57 68 73 74 75   
## Bad Good Good Good Good Bad Bad Bad Bad Good Good Good Good Good Good   
## 77 89 96 105 108 112 122 124 135 144 156 164 165 166 175   
## Good Bad Bad Good Good Bad Bad Bad Good Good Bad Bad Bad Good Bad   
## 180 188 197 198 200 203 219 229 232 237 238 241 245 249 270   
## Bad Bad Good Good Good Good Good Good Good Good Good Good Good Good Good   
## 278 282 283 285 296 301 303 315 325 329 343 345 353 355 357   
## Bad Bad Good Bad Bad Good Bad Good Good Bad Good Good Bad Good Good   
## 366 367 379 383 385 394 396 404 410 413 417 426 427 431 436   
## Good Bad Good Bad Good Bad Bad Bad Bad Good Good Good Good Good Good   
## 438 440 446 463 469 473 482 485 493 501 504 505 508 510 519   
## Bad Bad Bad Good Good Good Good Good Good Good Bad Good Good Good Good   
## 522 529 531 533 535 546 547 548 550 553 554 560 561 562 563   
## Good Good Good Good Good Good Good Good Good Good Good Good Good Good Good   
## 568 574 576 578 582 587 588 595 605 607 609 611 625 627 630   
## Good Bad Good Good Bad Good Bad Bad Bad Good Bad Good Good Bad Good   
## 647 650 652 658 669 671 676 677 678 681 685 699 701 702 707   
## Good Good Good Good Good Bad Bad Bad Bad Bad Good Bad Bad Good Bad   
## 712 713 719 737 742 747 749 750 753 764 777 779   
## Good Bad Good Bad Good Good Bad Good Good Good Bad Bad   
## Levels: Bad Good

lst[2]#displaying list 2

## $two  
## [1] Bad Good Good Good Good Bad Bad Good Bad Good Good Good Good Good  
## [15] Good Good Good Good Good Good Bad Bad Bad Good Good Bad Bad Bad   
## [29] Good Bad Good Bad Good Bad Bad Bad Good Good Bad Good Good Bad   
## [43] Good Bad Good Bad Good Good Bad Good Good Bad Bad Bad Bad Good  
## [57] Good Good Good Good Bad Bad Bad Good Good Bad Bad Bad Bad Bad   
## [71] Bad Good Good Good Good Bad Bad Bad Good Good Good Good Good Good  
## [85] Good Good Good Good Good Good Good Good Good Good Good Good Good Good  
## [99] Good Good Good Good Good Bad Good Bad Good Bad Bad Bad Good Bad   
## [113] Good Bad Good Bad Good Bad Bad Bad Good Good Good Good Good Good  
## [127] Bad Bad Bad Good Bad Good Good Bad Good Good Good Good Good Good  
## [141] Good Good Good Good Good Bad Bad   
## Levels: Bad Good

#writing for loop for printing mismatched values of predicted values and actual values   
for (index in 1:147) {  
 x\_cand <- lst$one[index]#storing the values of list one in x\_cand  
 y\_cand <- lst$two[index]#storing the values of list two in y\_cand  
   
 #if x\_cand(first list) value is not equal to y\_cand(second list)  
 if(x\_cand!=y\_cand){  
 print(Credit\_test$ID[index])#the printing the ID's of all mismatched values  
 }  
}

## [1] 46  
## [1] 89  
## [1] 96  
## [1] 180  
## [1] 198  
## [1] 200  
## [1] 203  
## [1] 232  
## [1] 241  
## [1] 249  
## [1] 282  
## [1] 296  
## [1] 315  
## [1] 325  
## [1] 353  
## [1] 366  
## [1] 379  
## [1] 383  
## [1] 413  
## [1] 417  
## [1] 504  
## [1] 562  
## [1] 568  
## [1] 574  
## [1] 576  
## [1] 578  
## [1] 595  
## [1] 625  
## [1] 630  
## [1] 671  
## [1] 681  
## [1] 685  
## [1] 699  
## [1] 701  
## [1] 702  
## [1] 707  
## [1] 713  
## [1] 737  
## [1] 749

Initially, I am setting the seed value with the last three digits of my student number Then checking the class of predicted value(pred\_credit) and actual values (Credit\_test$Credit Standing)of decision tree

Predicted values are factors and actual values are characters so I changed actual values to factors and then I made one list and kept these actual values and predicted values in to list I implemented one for loop in that loop I assigned list one values to x\_cand, list two values to y\_cand If the x\_cand Value is not equal to y\_cand values then I am printing the ID’s i.e the mismatched values ID’s

Therefore I got mismatched values. Now I had a look at the IDs from 681 to 707 totally six consecutive IDs produced the mismatched values i.e the incorrect values.

This is the method I used to find the incorrect pattern and I got six consecutive values from 681 to 707 shows an incorrect pattern

References:

* 1. <https://www.edureka.co/blog/implementation-of-decision-tree/>
  2. https://machinelearningmastery.com/machine-learning-ensembles-with-r/