Assignment 2

Part(A):

1-

Reading the data

```r
dataset <- read.csv(file = 'C:/Users/Genius/Downloads/Assignment 2/Churn Dataset.csv')
head(dataset )
df=as.data.frame(do.call(cbind, dataset))
head(df)
```

This is the result

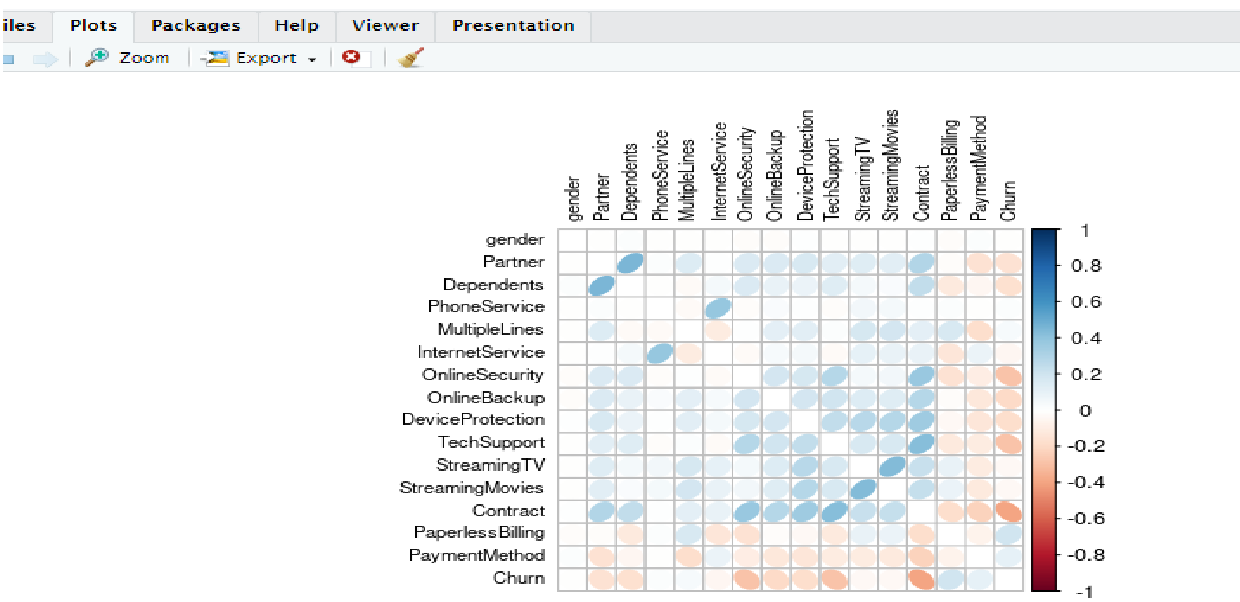
```
> head(df)
  customerID gender SeniorCitizen Partner Dependents tenure PhoneService    MultipleLines
1 7590-VHVEG Female             0     Yes         No      1           No No phone service
2 5575-GNVDE   Male             0      No         No     34          Yes               No
3 3668-QPYBK   Male             0      No         No      2          Yes               No
4 7795-CFOCW   Male             0      No         No     45           No No phone service
5 9237-HQITU Female             0      No         No      2          Yes               No
6 9305-CDSKC Female             0      No         No      8          Yes              Yes
  InternetService OnlineSecurity OnlineBackup DeviceProtection TechSupport StreamingTV StreamingMovies
1             DSL             No          Yes               No          No          No              No
2             DSL            Yes           No              Yes          No          No              No
3             DSL            Yes          Yes               No          No          No              No
4             DSL            Yes           No              Yes         Yes          No              No
5     Fiber optic             No           No               No          No          No              No
6     Fiber optic             No           No              Yes          No         Yes             Yes
        Contract PaperlessBilling             PaymentMethod MonthlyCharges TotalCharges Churn
1 Month-to-month              Yes          Electronic check          29.85        29.85    No
2       One year               No              Mailed check          56.95       1889.5    No
3 Month-to-month              Yes              Mailed check          53.85       108.15   Yes
4       One year               No Bank transfer (automatic)           42.3      1840.75    No
5 Month-to-month              Yes          Electronic check           70.7       151.65   Yes
6 Month-to-month              Yes          Electronic check          99.65        820.5   Yes
>
```

Scatter blot matrix

```r
install.packages("corrplot")
library(corrplot)
library(reshape2)
library(ggplot2)
print((df[c(3,6,19,20)]))

corrplot(cor(df[, sapply(df, is.numeric)],method = "pearson"),diag = FALSE,
         method = "ellipse",
         tl.cex = 0.7, tl.col = "black", cl.ratio = 0.2
)
```

And this is the scatter blot matrix of the numerical columns in the churn dataset

And this is scatter blot matrix to all data after convert categorical values to numerical



2-Heatmap

```
#---------------------------------------------------------------------------------------
#load reshape2 package to use melt() function
library(reshape2)

#melt data into long format
melt_ChurnDSet<- melt(cor(dataset[, sapply(dataset, is.numeric)],
                          method = "pearson"))
#create heatmap using rescaled values
ggplot(melt_ChurnDSet, aes(Var1, Var2)) +
  geom_tile(aes(fill = value), colour = "white") +
  scale_fill_gradient(low = "white", high = "red")
```

And that the heatmap to determine correlated attributes

Convert categorical to numerical values

```r
df$gender <- as.numeric(as.factor(df$gender))
df$Partner <- as.numeric(as.factor(df$Partner))
df$Dependents <- as.numeric(as.factor(df$Dependents))
df$PhoneService <- as.numeric(as.factor(df$PhoneService))
df$MultipleLines<- as.numeric(as.factor(df$MultipleLines))
df$InternetService<- as.numeric(as.factor(df$InternetService))
df$OnlineSecurity<- as.numeric(as.factor(df$OnlineSecurity))
df$OnlineBackup<- as.numeric(as.factor(df$OnlineBackup))
df$DeviceProtection<- as.numeric(as.factor(df$DeviceProtection))
df$TechSupport<- as.numeric(as.factor(df$TechSupport))
df$StreamingTV<- as.numeric(as.factor(df$StreamingTV))
df$StreamingMovies<- as.numeric(as.factor(df$StreamingMovies))
df$Contract<- as.numeric(as.factor(df$Contract))
df$PaperlessBilling<- as.numeric(as.factor(df$PaperlessBilling))
df$Contract<- as.numeric(as.factor(df$Contract))
df$PaymentMethod<- as.numeric(as.factor(df$PaymentMethod))
df$Churn<- as.numeric(as.factor(df$Churn))
head(df)
```

And this the data after convert categorical to numerical

```
> head(df)
  customerID gender SeniorCitizen Partner Dependents tenure PhoneService MultipleLines
1 7590-VHVEG      1             0       2          1      1            1             2
2 5575-GNVDE      2             0       1          1     34            2             1
3 3668-QPYBK      2             0       1          1      2            2             1
4 7795-CFOCW      2             0       1          1     45            1             2
5 9237-HQITU      1             0       1          1      2            2             1
6 9305-CDSKC      1             0       1          1      8            2             3
  InternetService OnlineSecurity OnlineBackup DeviceProtection TechSupport StreamingTV
1               1              1            3                1           1           1
2               1              3            1                3           1           1
3               1              3            3                1           1           1
4               1              3            1                3           3           1
5               2              1            1                1           1           1
6               2              1            1                3           1           3
  StreamingMovies Contract PaperlessBilling PaymentMethod MonthlyCharges TotalCharges Churn
1               1        1                2             3          29.85        29.85     1
2               1        2                1             4          56.95       1889.5     1
3               1        1                2             4          53.85       108.15     2
4               1        2                1             1           42.3      1840.75     1
5               1        1                2             3           70.7       151.65     2
6               3        1                2             3          99.65        820.5     2
> 
```

Then I will remove the customerID and TotalChrage features from the data

```r
df<-df %>% select(-customerID)
df<-df %>% select(-TotalCharges)
```

Check if there is a missing values in the data

```r
sum(is.na(df))
colSums(is.na(df))
```

Give me this result, there is no missing values

```
> sum(is.na(df))
[1] 0
> colSums(is.na(df))
          gender    SeniorCitizen          Partner       Dependents           tenure     PhoneService    MultipleLines 
               0                0                0                0                0                0                0 
 InternetService   OnlineSecurity     OnlineBackup DeviceProtection      TechSupport      StreamingTV  StreamingMovies 
               0                0                0                0                0                0                0 
        Contract PaperlessBilling    PaymentMethod   MonthlyCharges            Churn 
               0                0                0                0                0 
> 
```

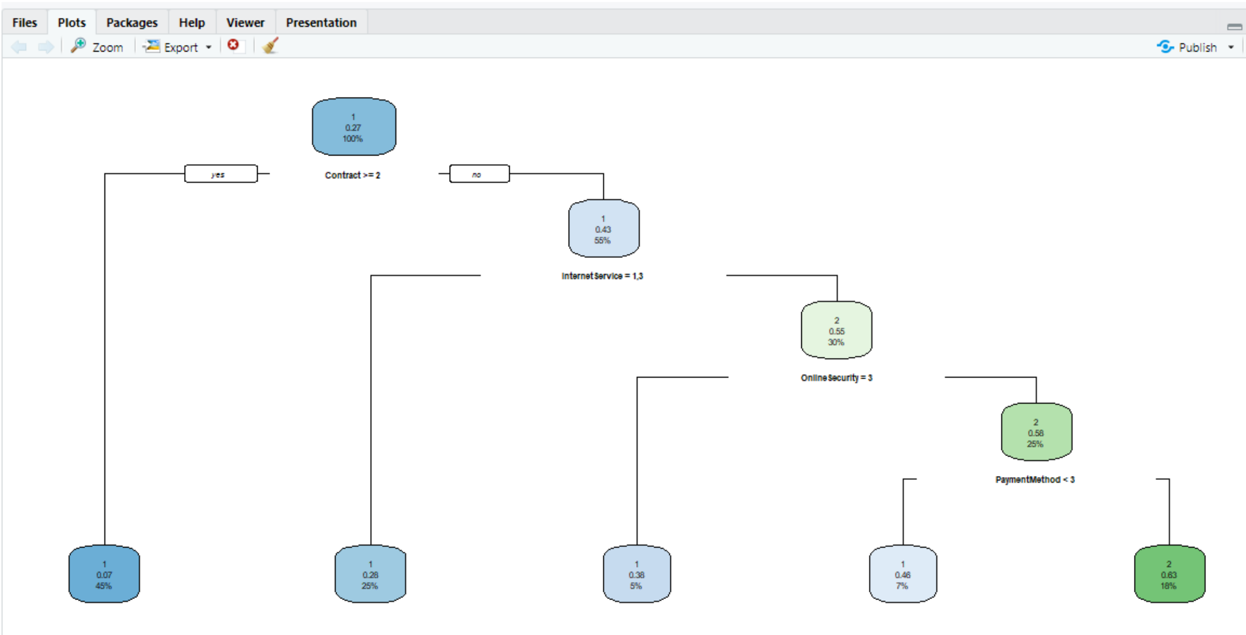
3-First Install some libraries

```r
install.packages("caTools")
library(caTools)
install.packages("rpart")
library(rpart)
install.packages("rpart.plot")
library(rpart.plot)
install.packages("caret")
library(caret)
install.packages("dplyr")
library(dplyr)
install.packages("lattice")
library(lattice)
install.packages("party")
library(party)
```

Then split the data to train set and test set

```r
set.seed(42)
sample_split <- sample.split(Y = df$Churn, SplitRatio = 0.80)
train_set <- subset(x = df, sample_split == TRUE)
test_set <- subset(x = df, sample_split == FALSE)
train_set
model <- rpart(Churn ~ ., data = train_set, method = "class") #specify method as class since we are dealing with
model
#plot the model
rpart.plot(model)
```

This is the tree




```
#Make predictions
preds <- predict(model, newdata = test_set, type = "class") #use the predict() function and pass in t
preds

#Print the confusion Matrix
confusionMatrix(test_set$Churn, preds) #check accuracy
```

And this is the results getting accuracy=77.5%

```
> confusionMatrix(test_set$Churn, preds) #check accuracy
Confusion Matrix and Statistics

          Reference
Prediction   1   2
         1 946  89
         2 228 146

               Accuracy : 0.775
                 95% CI : (0.7523, 0.7966)
    No Information Rate : 0.8332
    P-Value [Acc > NIR] : 1

                  Kappa : 0.3454

 Mcnemar's Test P-Value : 9.128e-15

            Sensitivity : 0.8058
            Specificity : 0.6213
         Pos Pred Value : 0.9140
         Neg Pred Value : 0.3904
             Prevalence : 0.8332
         Detection Rate : 0.6714
   Detection Prevalence : 0.7346
      Balanced Accuracy : 0.7135

       'Positive' Class : 1

> |
```

4-Try different ways to improve the decision tree algorithm (e.g., use different splitting strategies, prune tree after splitting). Does pruning the tree improves the accuracy?

First splitting by Information gain

```
decisionTreeInformation <- rpart(Churn ~ ., data = train_set, method = "class", parms = list(split = "information")) #specify m
decisionTreeInformation
```

```
#Make predictions
preds <- predict(decisionTreeInformation, newdata = test_set, type = "class") #use the predict() function and pass in t
preds
#Print the confusion Matrix
confusionMatrix(test_set$Churn, preds) #check accuracy
```

Getting accuracy =77.5% , and this is the Confusion matrix

```
> confusionMatrix(test_set$Churn, preds) #check accuracy
Confusion Matrix and Statistics

          Reference
Prediction   1   2
         1 946  89
         2 228 146

               Accuracy : 0.775
                 95% CI : (0.7523, 0.7966)
    No Information Rate : 0.8332
    P-Value [Acc > NIR] : 1

                  Kappa : 0.3454

 Mcnemar's Test P-Value : 9.128e-15

            Sensitivity : 0.8058
            Specificity : 0.6213
         Pos Pred Value : 0.9140
         Neg Pred Value : 0.3904
             Prevalence : 0.8332
         Detection Rate : 0.6714
   Detection Prevalence : 0.7346
      Balanced Accuracy : 0.7135

       'Positive' Class : 1
```

Second splitting the data by Gini Index

```
#Splitting by Gini Index
decisionTreeGini <- train(Churn ~ . , data = test_set,
                          method = "rpart",
                          parms = list(split = "gini"),
                          trControl = ct,
                          tuneLength = 100)
#Make predictions
preds <- predict(decisionTreeGini, newdata = test_set, type = "class") #use the predict() function and pass in the
preds
#Print the confusion Matrix
confusionMatrix(test_set$Churn, preds) #check accuracy
```

Getting accuracy =77.5% after splitting by gini index , and this is the Confusion matrix

```
Console    Terminal ×    Jobs ×
R 4.2.1 · /cloud/project/
> confusionMatrix(test_set$Churn, preds) #check accuracy
Confusion Matrix and Statistics

          Reference
Prediction   1   2
         1 931 104
         2 212 162

               Accuracy : 0.7757
                 95% CI : (0.753, 0.7973)
    No Information Rate : 0.8112
    P-Value [Acc > NIR] : 0.9996

                  Kappa : 0.3665

 Mcnemar's Test P-Value : 1.753e-09

            Sensitivity : 0.8145
            Specificity : 0.6090
         Pos Pred Value : 0.8995
         Neg Pred Value : 0.4332
             Prevalence : 0.8112
         Detection Rate : 0.6608
   Detection Prevalence : 0.7346
      Balanced Accuracy : 0.7118

       'Positive' Class : 1

>
```

Then Puruning the tree

```
Untitled1* ×
Run    Source
#pruning
decitionTreePrune <- rpart(Churn ~ ., data = train_set, method = "class",
                   control = rpart.control(cp = 0.0082, maxdepth = 3,minsplit = 2))
preds<- predict(decitionTreePrune, newdata = test_set, type = "class") #use the predict() function and pass in the testing subset
preds
#Print the confusion Matrix
confusionMatrix(test_set$Churn, preds) #check accuracy

171:1  (Top Level) ÷
Console    Terminal ×    Jobs ×
R 4.2.1 · /cloud/project/
 [ reached getOption("max.print") -- omitted 409 entries ]
Levels: 1 2
> confusionMatrix(test_set$Churn, preds) #check accuracy
```

Getting accuracy =77.7% after puruning , and this is the Confusion matrix

```
> confusionMatrix(test_set$Churn, preds) #check accuracy
Confusion Matrix and Statistics

          Reference
Prediction   1   2
         1 926 109
         2 218 156

               Accuracy : 0.7679
                 95% CI : (0.745, 0.7897)
    No Information Rate : 0.8119
    P-Value [Acc > NIR] : 1

                  Kappa : 0.3438

 Mcnemar's Test P-Value : 2.338e-09

            Sensitivity : 0.8094
            Specificity : 0.5887
         Pos Pred Value : 0.8947
         Neg Pred Value : 0.4171
             Prevalence : 0.8119
         Detection Rate : 0.6572
   Detection Prevalence : 0.7346
      Balanced Accuracy : 0.6991

       'Positive' Class : 1

>
```

Does pruning the tree improves the accuracy? Yes To some extent

5-first installing the libraries then split the train data to (X_train , y_train) and test data to (X_test ,y_test) and create a` xgboost model to train the data and and get the accuracy=78%

```r
install.packages('xgboost')     # for fitting the xgboost model
install.packages('caret')       # for general data preparation and model fitting
install.packages('e1071')
library(xgboost)
library(caret)
library(e1071)
my_data=as.list(df)

X_train = train_set[,-19]                 # independent variables for train
X_train <- matrix(unlist(X_train), ncol = 18, nrow = 5634)
y_train = train_set[,19]                                # dependent variables for train


X_test = test_set[,-19]                   # independent variables for test
X_test <- matrix(unlist(X_test), ncol = 18, nrow = 1409)
y_test = test_set[,19]

typeof(df)
# convert the train and test data into xgboost matrix type.
xgboost_train = xgb.DMatrix(data=X_train, label=y_train)
xgboost_test = xgb.DMatrix(data=X_test, label=y_test)
#Step 4 - Create a xgboost model
# train a model using our training data
model <- xgboost(data = xgboost_train,                    # the data
                 max.depth=3, ,                           # max depth
                 nrounds=70)                              # max number of boosting iterations

summary(model)
```

```r
#use model to make predictions on test data
pred_test = predict(model, xgboost_test)
pred_test

#Step 6 - Convert prediction to factor type
pred_test[(pred_test>3)] = 3
pred_y = as.factor((levels(y_test))[round(pred_test)])
print(pred_y)

#Step 7 - Create a confusion matrix
conf_mat = confusionMatrix(y_test, pred_y)
print(conf_mat)
```

And this is the confusion matrix

```
177:16   (Untitled) ÷

Console   Terminal ×   Background Jobs ×

R 4.2.1 · ~/assigment2/
> conf_mat=confusionMatrix(y_test,pred_y)
> print(conf_mat)
Confusion Matrix and Statistics

          Reference
Prediction   1   2
         1 917 118
         2 188 186

               Accuracy : 0.7828
                 95% CI : (0.7604, 0.8041)
    No Information Rate : 0.7842
    P-Value [Acc > NIR] : 0.5667

                  Kappa : 0.4077

 Mcnemar's Test P-Value : 7.998e-05

            Sensitivity : 0.8299
            Specificity : 0.6118
         Pos Pred Value : 0.8860
         Neg Pred Value : 0.4973
             Prevalence : 0.7842
         Detection Rate : 0.6508
   Detection Prevalence : 0.7346
      Balanced Accuracy : 0.7209

       'Positive' Class : 1

> |
```

6-Train a deep neural network using Keras with 3 dense layers. Try changing the activation function or dropout rate.

First installing keras

```
#----------------------------------------------------------------------------------------------------
install.packages('devtools')

devtools::install_github("rstudio/keras")
devtools::install_github("rstudio/reticulate")

install.packages("tensorflow")

install.packages('reticulate')
install.packages('keras')
library(reticulate)
library(keras)
library(tensorflow)

#reticulate::use_python("C:/Users/Genius/AppData/Local/Microsoft/WindowsApps/PythonSoftwareFoundation.Python.3.9_
reticulate::use_python("C:/Users/Genius/Anaconda3/")
install_tensorflow()
```

then creating a model has 3 dense layers and fit the training data

```
#defining a keras sequential model
model <- keras_model_sequential()

model %>%
  layer_dense(units = 19, input_shape = 784) %>%
  layer_dropout(rate=0.5)%>%
  layer_activation(activation = 'relu') %>%
  layer_dense(units = 50) %>%
  layer_activation(activation = 'softmax')
layer_dense(units = 2) %>%
  layer_activation(activation = 'softmax')


#compiling the defined model with metric = accuracy and optimiser as adam.
model %>% compile(
  loss = 'categorical_crossentropy',
  optimizer = 'adam',
  metrics = c('accuracy')
)


#fitting the model on the training dataset
model %>% fit(x_train , y_train , epochs = 50, batch_size = 128)

#Evaluating model on the cross validation dataset
loss_and_metrics <- model %>% evaluate(x_test , y_test, batch_size = 128)

```

```
preds_DNN <- predict(model , newdata = test_set, type = "class") #use the predict() function and pass in the testing subset
preds_DNN

#Print the confusion Matrix
confusionMatrix(test_set$Churn, preds_DNN) #check accuracy
```

And this is the confusion matrix , getting accuracy=77.4%

```
44/44 [==============================] - 0s 2ms/step
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 999  34
         1 283  88

               Accuracy : 0.7742
                 95% CI : (0.7514, 0.7959)
    No Information Rate : 0.9131
    P-Value [Acc > NIR] : 1

                  Kappa : 0.2603

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.7793
            Specificity : 0.7213
         Pos Pred Value : 0.9671
         Neg Pred Value : 0.2372
              Precision : 0.9671
                 Recall : 0.7793
                     F1 : 0.8631
             Prevalence : 0.9131
         Detection Rate : 0.7115
   Detection Prevalence : 0.7358
      Balanced Accuracy : 0.7503

       'Positive' Class : 0
```

Is there any sign of overfitting? From confusion matrix ,Sensitivity (also known as recall) and Pos Pred Value(also known as precision). Then F1 can be easily computed, as stated above, as: F1 <- (2 * precision * recall) / (precision + recall) so F1=0.8570 ,**So there is no Overfitting**

7- Compare the performance of the models in terms of the following criteria: precision, recall, accuracy, F measure. Identify the model that performed best and worst according to each criteria

From confusion matrix ,Sensitivity (also known as recall) and Pos Pred Value(also known as precision). Then F1 can be easily computed, as stated above, as: F1 <- (2 * precision * recall) / (precision + recall)

Model	Accuracy	Recall	Precision	F1
Decision Tree	0.775	0.8058	0.9140	0.8564
XGboost	0.7828	0.8299	0.8860	0.8570
DNN	0.7742	0.7793	0.9671	0.8631

-The best model that perform highest Accuracy **is XGboost** ,the worst model is **DNN**

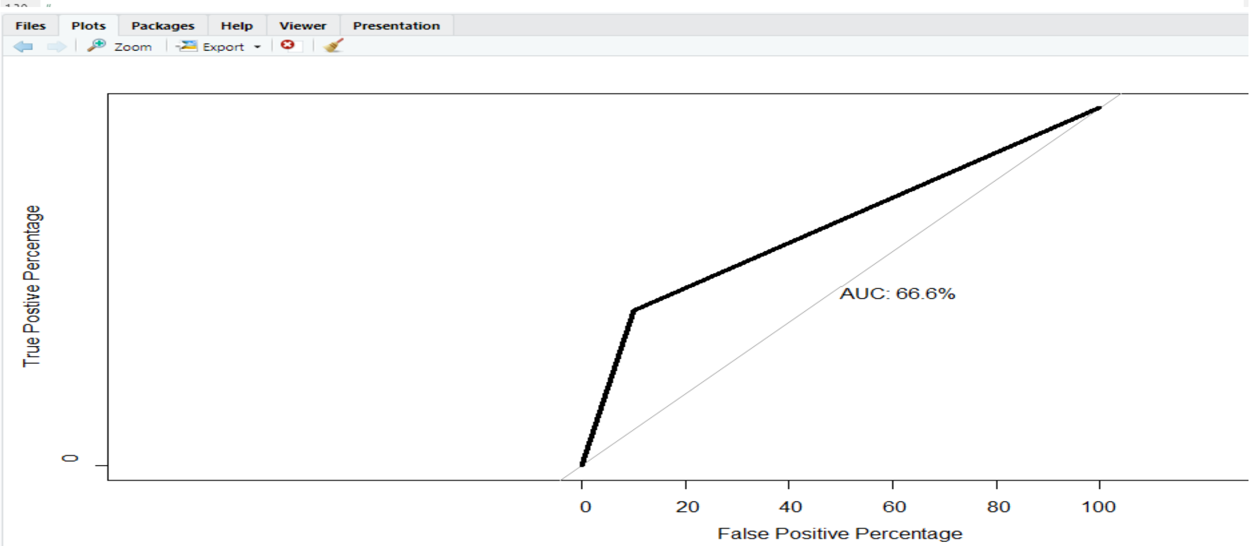
-The best model that perform highest Recall is **XGboost** ,the worst model is **DNN**

-The best model that perform highest Precision is DNN ,the worst model is **XGboost**

-The best model that perform highest F1 score is **DNN** ,the worst model is **Decision Tree**

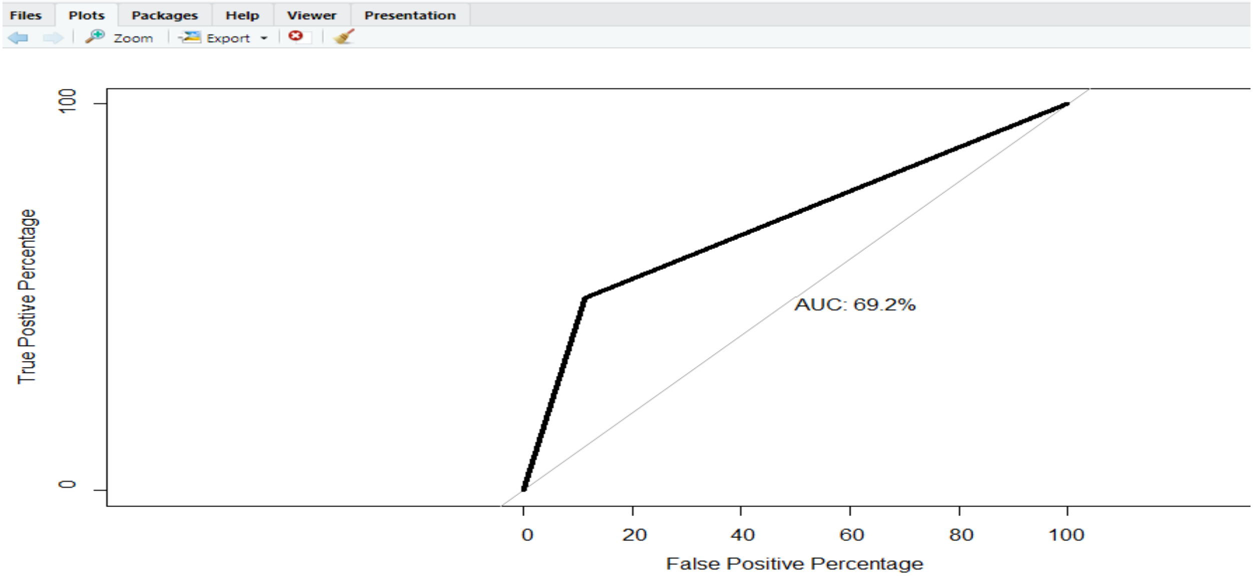
8- first ROC for Decision tree

```
install.packages("pROC")
library(pROC)
roc(test_set$Churn, as.numeric(as.character(preds)), plot=TRUE, legacy.axes=TRUE, percent=TRUE, xlab="False Positive Percentage", ylab="True Postive Percenta
legend("bottomright", legend=c("model_name"), col=c("#000000"), lwd=4)
```



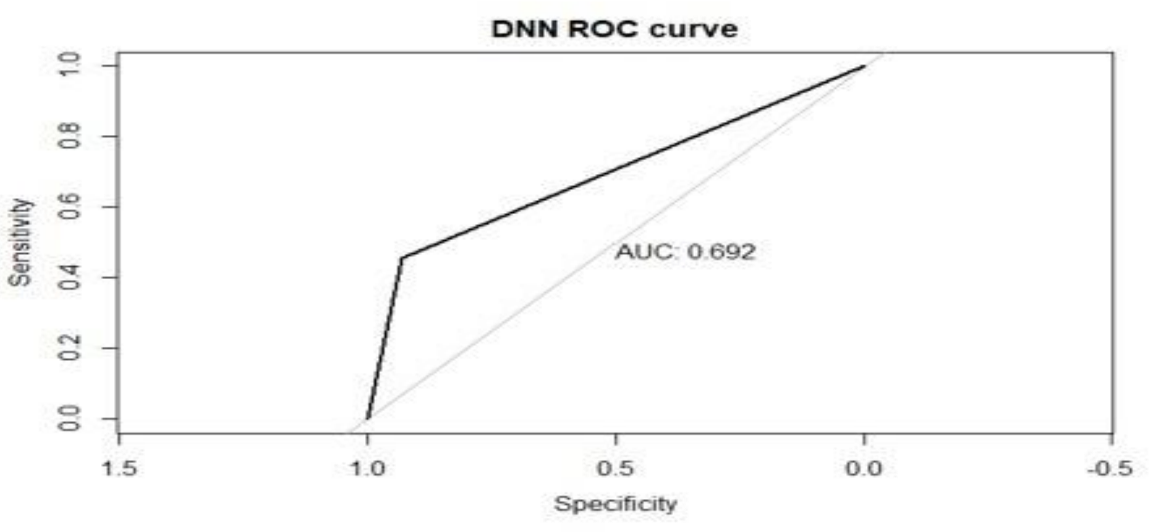
Second For ROC **XGboost**

```
roc(test_set$Churn, as.numeric(as.character(pred_y)), plot=TRUE, legacy.axes=TRUE, percent=TRUE, xlab="False Positive Percentage", ylab="True Post
legend("bottomright", legend=c("model_name"), col=c("#000000"), lwd=4)
```



Third For ROC DNN

```r
#ROC graph For DNN
roc(test_set$Churn, as.numeric(as.character(preds_DNN)), plot=TRUE, legacy.axes=TRUE, percent=TRUE, xlab="False Positive Percentage", ylab="True Postive Pe
legend("bottomright", legend=c("model_name"), col=c("#000000"), lwd=4)
```



Part(B)

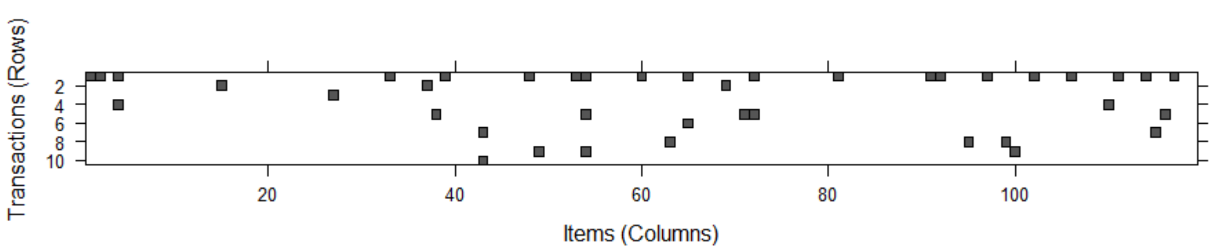
(a)-First reading the transactions data set

```r
#-----------------------------------------------------------------------------------------------------------
#Part (B) in the Assigment
#first reading the transactions

install.packages("arules")
library(arules)
install.packages("arulesviz")
library(arulesviz)
install.packages("readr")
library(readr)
install.packages("RColorBrewer")
library(RColorBrewer)

d <- read.transactions('C:/Users/Genius/Downloads/Assignment 2/transactions.csv', format = 'basket', sep = ',')
head(d )
typeof(d)
summary(d)
plot(head(d,10))
```

And this is the summary of the data

```
> summary(d)
transactions as itemMatrix in sparse format with
 7501 rows (elements/itemsets/transactions) and
 119 columns (items) and a density of 0.03288973

most frequent items:
mineral water          eggs     spaghetti  french fries     chocolate        (Other)
         1788          1348          1306          1282          1229          22405

element (itemset/transaction) length distribution:
sizes
   1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16   18   19   20
1754 1358 1044  816  667  493  391  324  259  139  102   67   40   22   17    4    1    2    1

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.000   2.000   3.000   3.914   5.000  20.000

includes extended item information - examples:
            labels
1          almonds
2 antioxydant juice
3        asparagus
>
```

And this is a plot of the top 10 transactions



(b)-

Generate association rules using minimum support of 0.002, minimum confidence of 0.20, and maximum length of 3

```r
# set better support and confidence levels to learn more rules
transaction_rules <- apriori(d, parameter = list(support =0.002, confidence =0.20, maxlen = 3))
summary(transaction_rules)
plot(head(d,10))
```

Display the rules: this is a summary to rules

```
> summary(transaction_rules)
set of 2023 rules

rule length distribution (lhs + rhs):sizes
   1    2    3 
   1  357 1665 

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  1.000   3.000   3.000   2.823   3.000   3.000 

summary of quality measures:
    support           confidence        coverage             lift             count       
 Min.   :0.002133   Min.   :0.2000   Min.   :0.002666   Min.   : 0.8595   Min.   :  16.0  
 1st Qu.:0.002533   1st Qu.:0.2405   1st Qu.:0.008266   1st Qu.: 1.5377   1st Qu.:  19.0  
 Median :0.003466   Median :0.2941   Median :0.011465   Median : 1.8674   Median :  26.0  
 Mean   :0.005292   Mean   :0.3177   Mean   :0.018647   Mean   : 2.0415   Mean   :  39.7  
 3rd Qu.:0.005599   3rd Qu.:0.3774   3rd Qu.:0.019064   3rd Qu.: 2.3381   3rd Qu.:  42.0  
 Max.   :0.238368   Max.   :0.9500   Max.   :1.000000   Max.   :28.0881   Max.   :1788.0  

mining info:
 data ntransactions support confidence                                                                      call
    d          7501   0.002        0.2 apriori(data = d, parameter = list(support = 0.002, confidence = 0.2, maxlen = 3))
> 
```

And this is the first two rules

```
> inspect(transaction_rules[1:2])
    lhs            rhs             support     confidence coverage   lift     count
[1] {}          => {mineral water} 0.238368218 0.2383682  1.00000000 1.000000 1788 
[2] {asparagus} => {mineral water} 0.002133049 0.4444444  0.00479936 1.864529   16 
> 
```

Sorting the transactions rules by descending lift value

```r
# sorting  descending transactions rules by lift to determine actionable rules
top.lift <- sort(transaction_rules, decreasing = TRUE, na.last = NA, by = "lift")
inspect(top.lift[1:5])
```

```
> top.lift1 <- sort(transaction_rules, decreasing = TRUE, na.last = NA, by = "lift")
> inspect(top.lift[1:5])
    lhs                                rhs                     support     confidence coverage    lift      count
[1] {escalope, mushroom cream sauce}  => {pasta}               0.002532996 0.4418605  0.005732569 28.088096 19   
[2] {escalope, pasta}                 => {mushroom cream sauce} 0.002532996 0.4318182  0.005865885 22.650826 19   
[3] {mushroom cream sauce, pasta}     => {escalope}            0.002532996 0.9500000  0.002666311 11.976387 19   
[4] {parmesan cheese, tomatoes}       => {frozen vegetables}   0.002133049 0.6666667  0.003199573  6.993939 16   
[5] {mineral water, whole wheat pasta} => {olive oil}          0.003866151 0.4027778  0.009598720  6.115863 29   
> 
```

(c)

first Generating a transaction rule for second case with maximum length of 2

```r
transaction_rules2 <- apriori(d, parameter = list(support =0.002, confidence =0.20, maxlen = 2))
inspect(transaction_rules2[1:5])
```

```
> inspect(transaction_rules2[1:5])
    lhs              rhs             support     confidence coverage    lift      count
[1] {}            => {mineral water} 0.238368218 0.2383682  1.000000000 1.0000000 1788
[2] {asparagus}   => {mineral water} 0.002133049 0.4444444  0.004799360 1.8645290   16
[3] {candy bars}  => {mineral water} 0.002266364 0.2328767  0.009732036 0.9769621   17
[4] {shallot}     => {green tea}     0.002266364 0.2931034  0.007732302 2.2185358   17
[5] {shallot}     => {french fries}  0.002666311 0.3448276  0.007732302 2.0175910   20
>
```

here is the two rules t1 is the rule from QII-b with the greatest lift, t2 is rule with the highest lift rule for maximum length of 2

```
t1<-inspect(top.lift[1])
t2<-inspect(transaction_rules2[1])
```

```
> t1<-inspect(top.lift[1])
    lhs                                rhs     support     confidence coverage    lift    count
[1] {escalope, mushroom cream sauce} => {pasta} 0.002532996 0.4418605  0.005732569 28.0881 19
> t2<-inspect(transaction_rules2[1])
    lhs    rhs             support   confidence coverage lift count
[1] {}  => {mineral water} 0.2383682 0.2383682  1        1    1788
>
```

(i) The **first** one has a **better left** than the second one, The **second** one has **a better support** than the first one

(ii)

(iii) Selecting the rule with **highest lift** because Lift is a relative strength indicator showing the association between ,so I will select

```
> t1<-inspect(top.lift[1])
    lhs                                rhs     support     confidence coverage    lift    count
[1] {escalope, mushroom cream sauce} => {pasta} 0.002532996 0.4418605  0.005732569 28.0881 19
```