uOttawa

Faculté de génie
Faculty of Engineering

Report Assignment 2

Classification and Clustering

DT15126 [EG] Fundamentals /Applied Data Sci 20219

Name: Abdallah Medhat Mohamed Rashed

Email: arash015@uOttawa.ca

ID: 300273110

Objectives

The purpose of this assignment to apply classification and clustering on the different data using R.

Part A: Classification

1) Import the data (Churn Dataset.csv) by using read.csv and use some functions (str and glimpse) to see internal structure about data but glimpse is used to see more data from str**.**

```
# Read the Data
Data <- read.csv("Churn Dataset.csv",header=TRUE,stringsAsFactors = TRUE)
str(Data)
glimpse(Data)
```

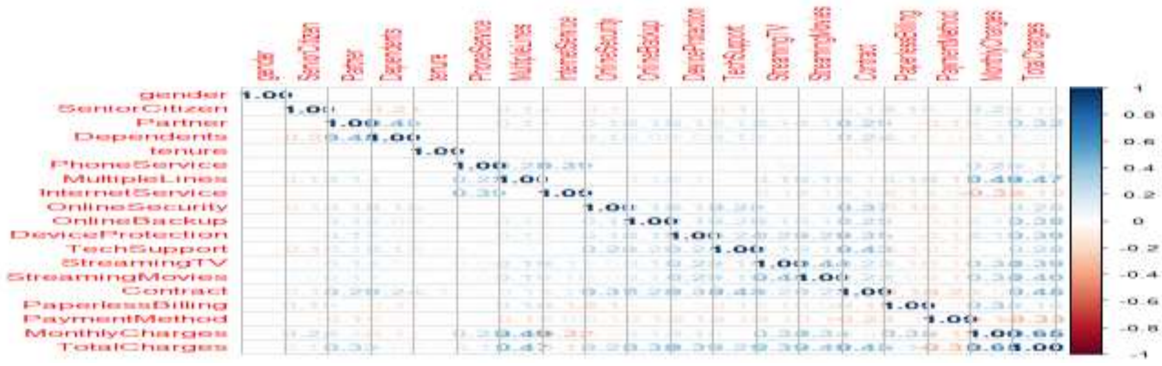
2) After using str and glimpse we need do some changing in data such as delete the null value , delete some column not important and convert some value in some column in data to correct format or value such as convert 'No phone service' to 'No' in column MultipleLines, change value '0','1' to 'NO', 'yes' in column SeniorCitizen and convert column tenure from more number to interval because the tenure has more values.

```
#some changes in data
Data$MultipleLines[Data$MultipleLines == "No phone service"]<-'No'
Data$SeniorCitizen <- as.factor(mapvalues(Data$SeniorCitizen,
                                          from=c("0","1"),
                                          to=c("No", "Yes")))
```

```
collect_tenure <- function(tenure){
  if (tenure >= 0 & tenure <= 12){
    return('0-12 Month')
  }else if(tenure > 12 & tenure <= 24){
    return('12-24 Month')
  }else if (tenure > 24 & tenure <= 48){
    return('24-48 Month')
  }else if (tenure > 48 & tenure <=60){
    return('48-60 Month')
  }else if (tenure > 60){
    return('> 60 Month')
  }
}
Data$tenure <- sapply(Data$tenure,collect_tenure)
Data$tenure <- as.factor(Data$tenure)
Data$customerID <- NULL
```

```
#missing Data and drop the missing data
summary(is.na(Data))
sum(is.na(Data))
Data=na.omit(Data)
```

3) Plot scatterplot matrix to show the relationships between the variables and a correlation matrix to determine correlated attributes by two method. The first method convert categorical data to numerical data to plot matrix. The second plot the numerical feature only in the code file.

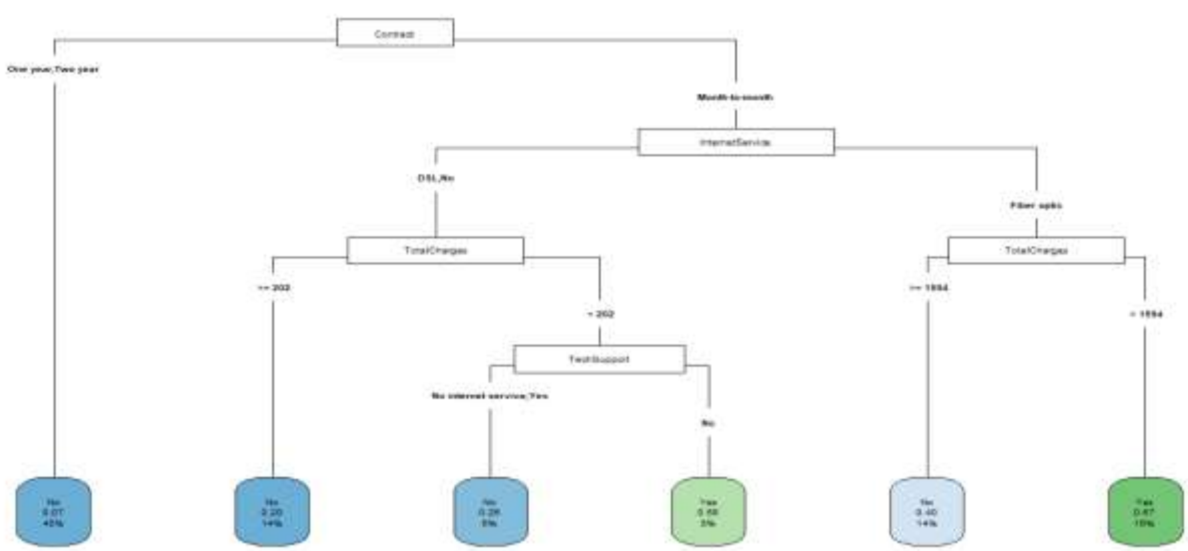


4) Apply decision tree with split data set into 80 training /20 test set and plot decision tree and plot ROC.
   Note: apply decision tree on the categorical data because decision tree can be applied on categorical data or numerical data but the neural network and XGboost must data is been numerical.

```
set.seed(42)
split = sample.split(Y =Data$Churn, SplitRatio = 0.8)
train_set = subset(Data, split == TRUE)
test_set = subset(Data, split == FALSE)
model <- rpart(Churn ~ ., data = train_set, method = "class")
#important features
importance <- varImp(model, scale=FALSE)
print(importance)
# Predicting the Test set results
y_pred = predict(model, newdata = test_set[-20], type = 'class')
# Making the Confusion Matrix and calculate accuracy
cm = table(test_set$Churn, y_pred)
accuracy=sum(diag(cm))/sum(cm)
#plot tree
rpart.plot(model,type = 5)
#ROC
roc(test_set$Churn,c(y_pred),plot=TRUE,direction="<",
    percent=TRUE,legacy.axes=TRUE,xlab="false positive",ylab='True positive',
    main='decision tree')
```

```
      y_pred
       No Yes
  No  920 113
  Yes 174 200
```

Accuracy = 0.796(79.6%).



5) Describe the first few splits in the decision tree.

If (contract == 'month-to-month')

If (internelservice=='fiber optic')

If (TotalCharges>=1549)

Class (No)

If (TotalCharges<1549)

Class (Yes)

If (internelservice=='Dsl' and internelservice=='No')

If (TotalCharges>=202)

Class (No)

If (TotalCharges<202)

If (TechSupport=='NO')

Class (yes)

If (TechSupport=='Yes' and TechSupport=='No internet service')

Class (No)

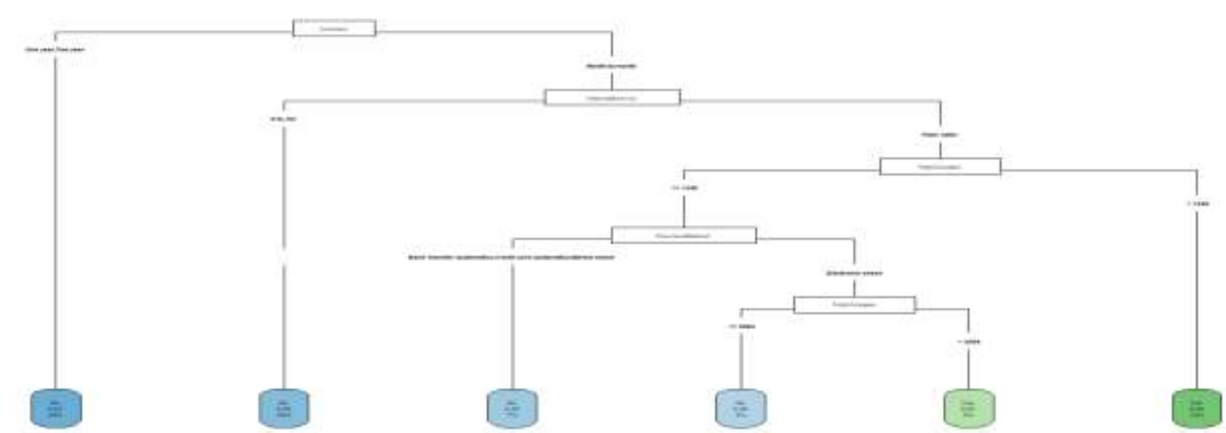
If (contract == 'one year' and contract == 'two year')

Class (No)

6) Improve decision tree by splitting and pruning by two method. The first method splitting by gini and apply pruning after splitting. The second method splitting by entropy and apply pruning after splitting.

Splitting (gini):

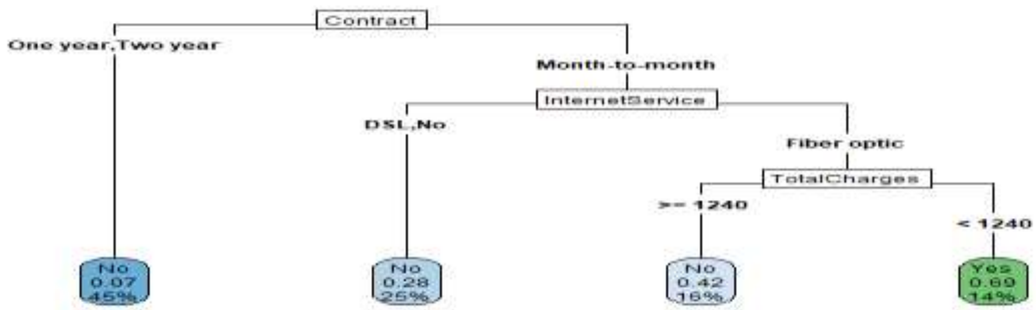
```
set.seed(43)
split_new = sample.split(Y =Data$Churn, SplitRatio = 0.8)
train_set_new = subset(Data, split_new == TRUE)
test_set_new = subset(Data, split_new == FALSE)
model_new <-rpart(Churn ~ ., data = train_set_new, method = "class",
                  parms=list(split = "gini"), minsplit = 10, minbucket=2)
y_pred_new = predict(model_new, newdata = test_set_new[-20], type = 'class')

# Making the Confusion Matrix and calculate accuracy
cm_new = table(test_set_new$Churn, y_pred_new)
accuracy_new=sum(diag(cm_new))/sum(cm_new)

#plot tree
rpart.plot(model_new,type=5)
```



Accuracy=0.794(79.4%)

Pruning:

```
printcp(model_new)
plotcp(model_new)
best_cp <- model_new$cptable[which.min(model_new$cptable[,"xerror"]),"CP"]

#pruned model
Model_new_pruned <- prune(model_new, cp =best_cp)
y_pred_new_pruned = predict(Model_new_pruned, newdata = test_set_new[-20],
                    type = 'class')

# Making the Confusion Matrix and calculate accuracy
cm_new_pruned = table(test_set_new$Churn, y_pred_new_pruned)
accuracy_new_pruned=sum(diag(cm_new_pruned))/sum(cm_new_pruned)
#plot tree
rpart.plot(Model_new_pruned,type = 5)
#Roc
roc(test_set_new$Churn,c(y_pred_new_pruned),plot=TRUE,direction="<",
    percent=TRUE,legacy.axes=TRUE,xlab="false positive",ylab='True positive',
    main='decision tree improve1')
```

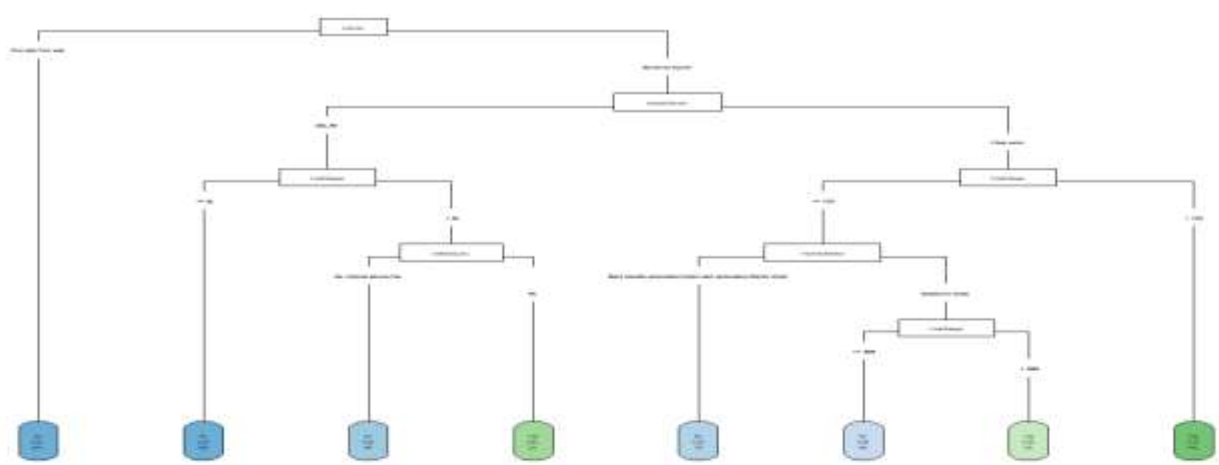


Accuracy=0.786(78.6%). Pruning is effect on the accuracy of the model but the affect is very small. I think this model is good and does not need to the pruning.

Splitting (entropy):

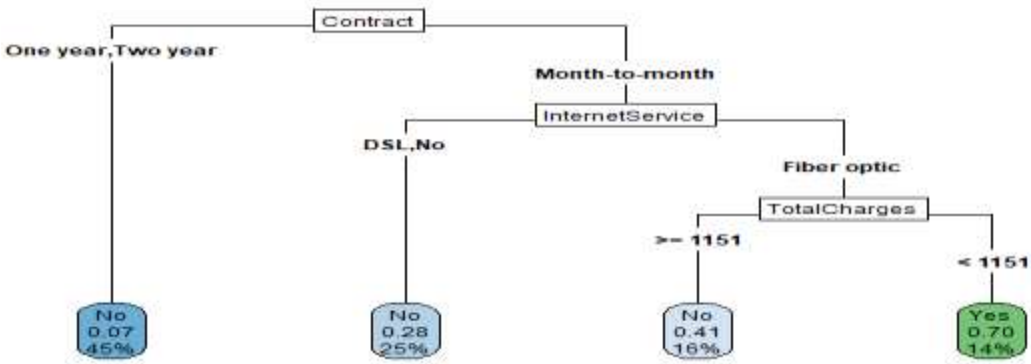
```
#improve2 Decision Tree
set.seed(44)
split_new2 = sample.split(Y =Data$Churn, SplitRatio = 0.8)
train_set_new2 = subset(Data, split_new2 == TRUE)
test_set_new2 = subset(Data, split_new2 == FALSE)
model_new2 <-rpart(Churn ~ ., data = train_set_new2, method = "class",
                   parms=list(split = "entropy"))
y_pred_new2 = predict(model_new2, newdata = test_set_new2[-20], type = 'class')

# Making the Confusion Matrix and calculate accuracy
cm_new2 = table(test_set_new2$Churn, y_pred_new2)
accuracy_new2=sum(diag(cm_new2))/sum(cm_new2)

#plot tree
rpart.plot(model_new2,type = 5)
```

Accuracy=0.803(80.3%)



Pruning:

```
#calculate best cp
printcp(model_new2)
plotcp(model_new2)
best_cp2 <- model_new2$cptable[which.min(model_new2$cptable[,"xerror"]),"CP"]
#pruned model
Model_new_pruned2 <- prune(model_new2, cp =best_cp2)
y_pred_new_pruned2 = predict(Model_new_pruned2, newdata = test_set_new2[-20],
                        type = 'class')
# Making the Confusion Matrix and calculate accuracy
cm_new_pruned2 = table(test_set_new2$Churn, y_pred_new_pruned2)
accuracy_new_pruned2=sum(diag(cm_new_pruned2))/sum(cm_new_pruned2)
#plot tree
rpart.plot(Model_new_pruned2,type = 5)
#Roc
roc(test_set_new2$Churn,c(y_pred_new_pruned2),plot=TRUE,direction="<",
    percent=TRUE,legacy.axes=TRUE,xlab="false positive",ylab='True positive',
    main='decision tree improve2')
```



Accuracy= 0.784 (78.4%). Pruning is effect on the accuracy of the model but the affect is very small. I think this model is good and does not need to the pruning.

7) Apply XGboost model using 10-fold cross-validation repeated 3 times and a hyperparameter grid search to train the optimal model**.**

```
#XGboost
set.seed(45)
split_xg = sample.split(Y =dataset$Churn, SplitRatio = 0.8)
train_set_xg = subset(dataset, split_xg == TRUE)
test_set_xg = subset(dataset, split_xg == FALSE)
xg_trcontrol = trainControl(method = "repeatedcv",number = 10, repeats = 3,
                            search = "grid")
xgbGrid <-  expand.grid(max_depth = c(3, 5, 7),
                        nrounds = c(10,20,5),
                        eta = 0.3,
                        gamma = 0,
                        subsample = 1,
                        min_child_weight = 1,
                        colsample_bytree = 0.6)
xgb_model = train(as.matrix(train_set_xg[-20]),train_set_xg$Churn,
                  trControl = xg_trcontrol,tuneGrid = xgbGrid,method = "xgbTree")
xgb_model$bestTune
```

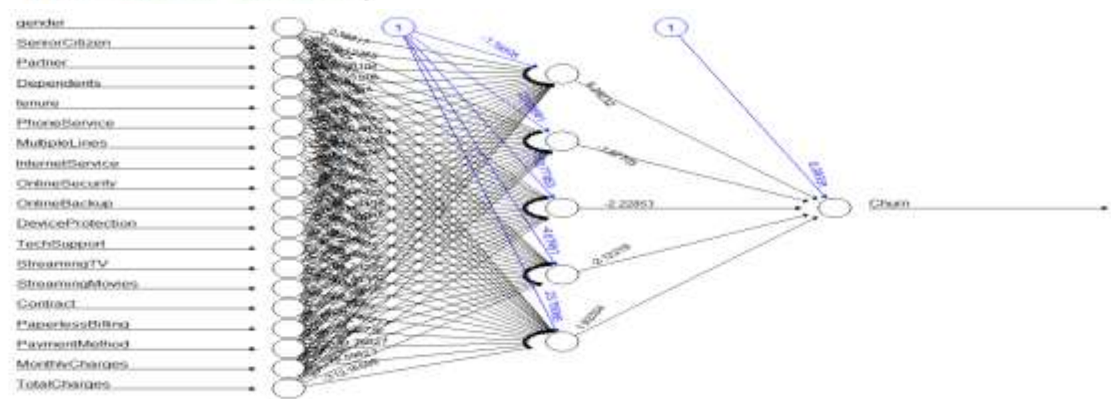
```
xgb_model$bestTune
xgb_model
y_pred_xgb = predict(xgb_model, newdata = as.matrix(test_set_xg[-20]))
cm_xgb = table(test_set_xg[, 20], y_pred_xgb)
accuracy_xbg=sum(diag(cm_xgb))/sum(cm_xgb)

#Roc
roc(test_set_xg$Churn,c(y_pred_xgb),plot=TRUE,direction="<",
    percent=TRUE,legacy.axes=TRUE,xlab="false positive",ylab='True positive',
    main='XGboost')
```

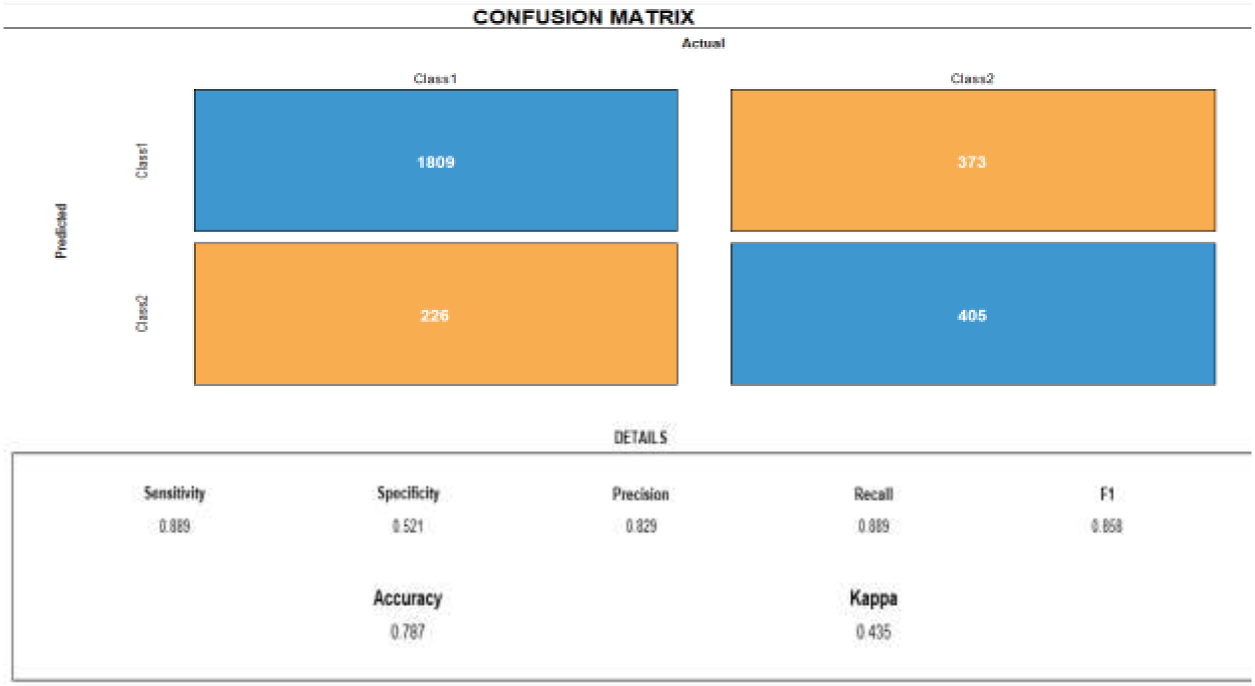
Hyperparameter: nrounds=20, max_depth=3, eta=0.3, gamma=0, colsample_bytree=0.6, min_child_weight=1, subsample=1.

Accuracy=0.805(80.5%).

8) Build a multilayer perceptron with 5 nodes at the hidden layer. Use a standard or normalization to scale the variables. Try changing the activation function, varying the neurons, learning rate, epochs or removing the bias. What effects does any of these have on the result? With a confusion matrix, evaluate the performance of the NN model based on sensitivity, specificity & accuracy.

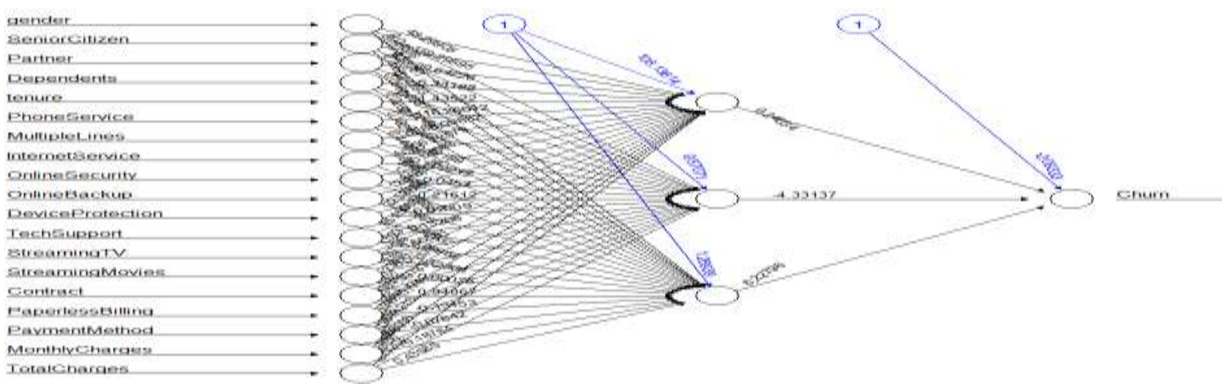
```
dataset_nn=dataset
dataset_nn <- lapply(dataset_nn, as.numeric)
dataset_nn=as.data.frame(dataset_nn)
samplesize = 0.60 * nrow(dataset_nn)
set.seed(80)
index = sample(seq_len(nrow(dataset_nn)), size = samplesize )

# Create training and test set
datatrain_per = dataset_nn[ index, ]
datatest_per = dataset_nn[ -index, ]

#Scale data for neural network
max = apply(dataset_nn , 2 , max)
min = apply(dataset_nn, 2 , min)
scaled = as.data.frame(scale(dataset_nn, center = min, scale = max - min))

# creating training and test set
trainNN = scaled[index , ]
testNN = scaled[-index , ]

# fit neural network
set.seed(2)
NN = neuralnet(Churn ~ ., trainNN, hidden =5  , linear.output = F )
summary(NN)
plot(NN)
predict_testNN = compute(NN, testNN[-20])
NN_p <- predict_testNN$net.result
predict_testNN = (predict_testNN$net.result * (max(dataset_nn$Churn) -
                                                min(dataset_nn$Churn))) +
  min(dataset_nn$Churn)

# calculate values
pred_NN <- ifelse(NN_p>0.5, 1, 0)
cm_NN<- table(pred_NN, testNN$Churn)
accuracy_NN=sum(diag(cm_NN))/sum(cm_NN)
confusionMatrix_NN=confusionMatrix(cm_NN)
draw_confusion_matrix(confusionMatrix_NN)

#ROC
roc(testNN$Churn,c(pred_NN),plot=TRUE,direction="<",percent=TRUE,
    legacy.axes=TRUE,xlab="false positive",ylab='True positive',
    main='neural network')
```

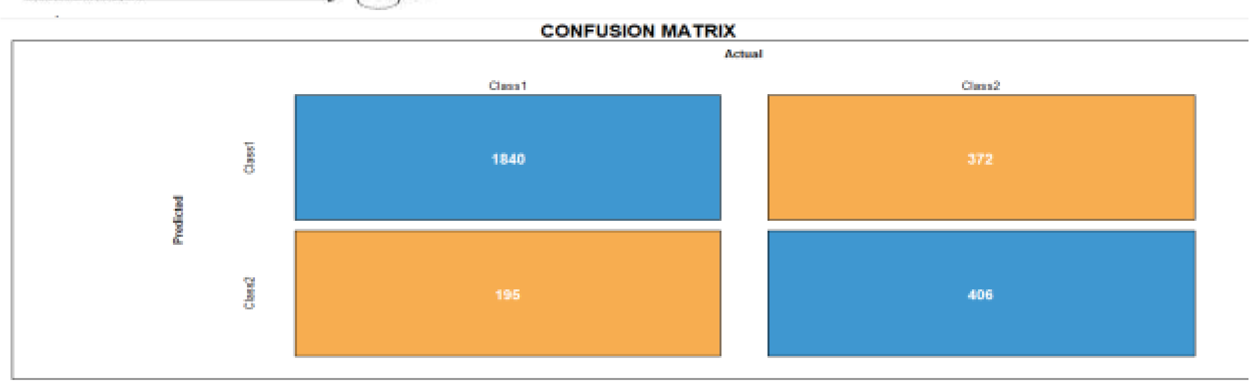




Accuracy=0.787(78.7%).

Change in neural network (number of node in hidden layer, activation function and stepmax) led to increase the accuracy and sensitivity.





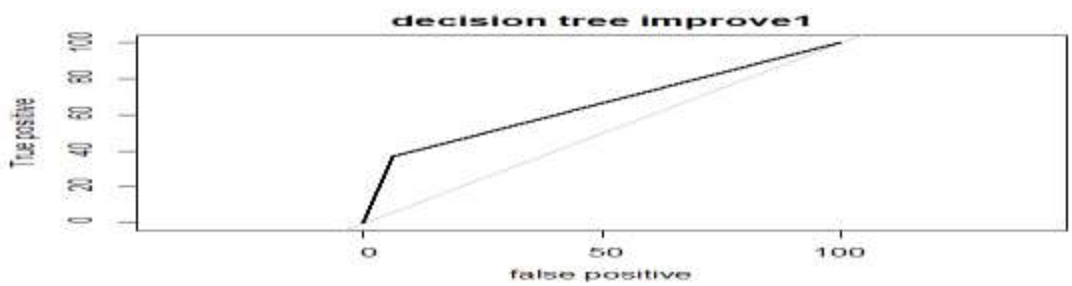
DETAILS				
Sensitivity	Specificity	Precision	Recall	F1
0.904	0.522	0.832	0.904	0.866

Accuracy	Kappa
0.798	0.458

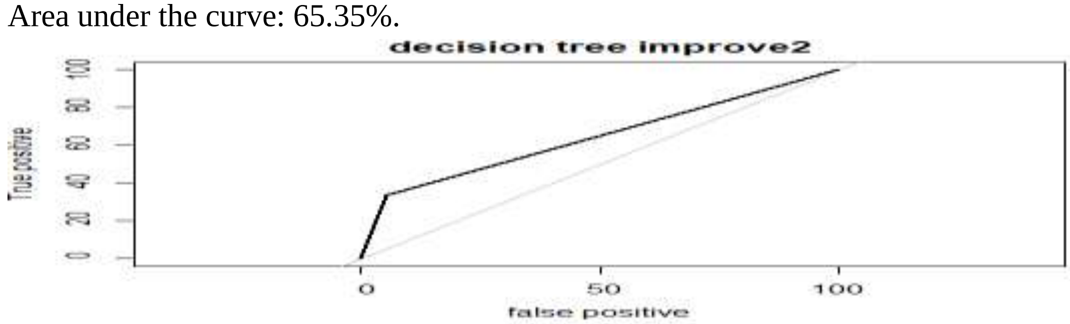
Accuracy=0.798(79.8%).

9) Carry out a ROC analysis to compare the performance of the DT, XGboost & NN techniques. Plot the ROC graph of the models. Plot ROC to every model alone and compare by AUC. The more AUC is the best model.

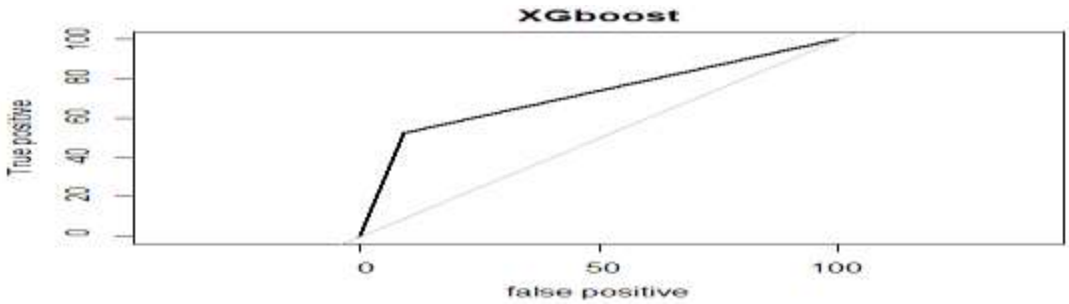
Area under the curve: 71.27%.



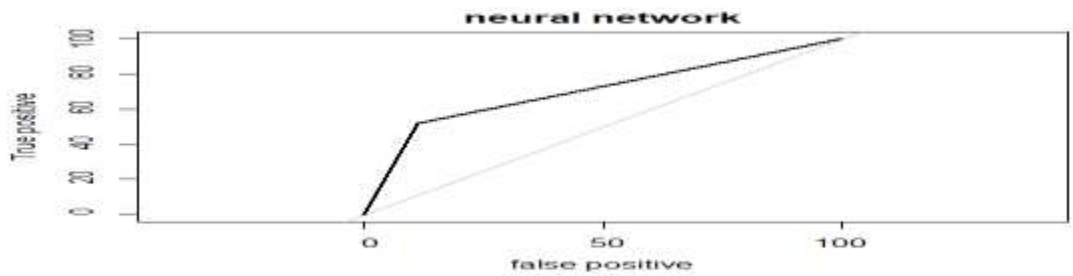
Area under the curve: 65.35%.



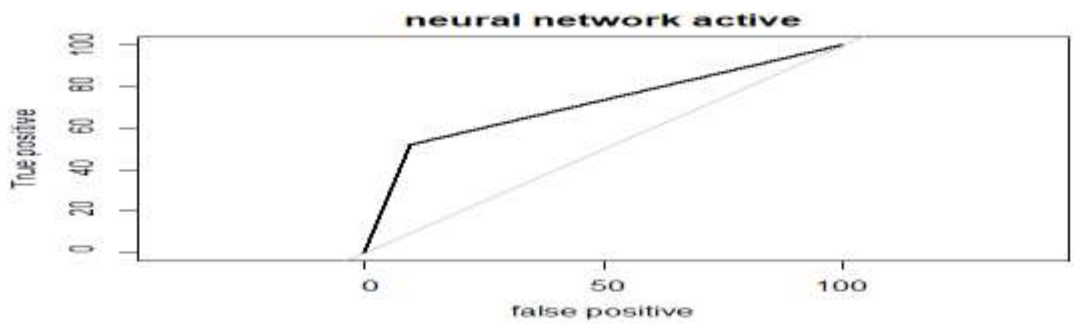
Area under the curve: 64.1%.



Area under the curve: 71.56%.



Area under the curve: 70.48%.



Area under the curve: 71.3%.

After applying the more model and carrying out ROC, XGboost is the best model. The accuracy of XGboost model is more than the other models the result of that is the XCboost the best model.

Part B: Clustering

Part1:

1) Reading data from file csv (Shopping_Customers.csv) by using read.csv and use some functions (str and glimpse) to see internal structure about data but glimpse is used to see more data from str.

```
# Read the Data
Data <- read.csv("Shopping_Customers.csv",header=TRUE)
str(Data)
glimpse(Data)
```

2) The data has five column but remove the first and second column (CustomerID and Gender) by two different ways because of applying clustering on this data (clustering is applied on numerical data but gender column is categorical data) and customerID column don't use in every or some model of machine learning.(the second way in the code)

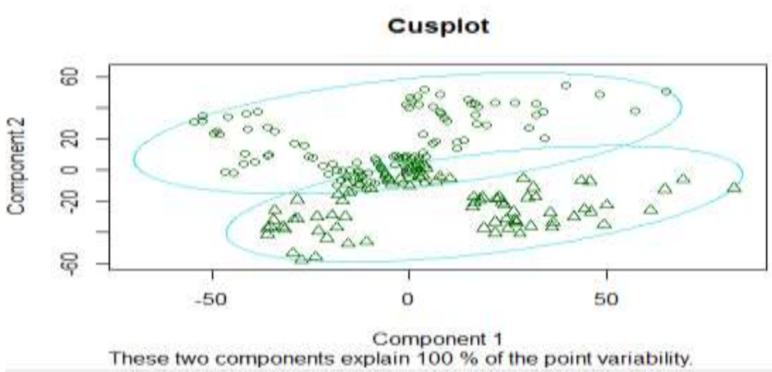
The first way:

```
#other remove column Gender and CustomerID
Data1=Data[,3:5]
str(Data1)
glimpse(Data1)
```

In some model of machine learning convert categorical data to numerical data but I don't use this way because I see gender column isn't important.

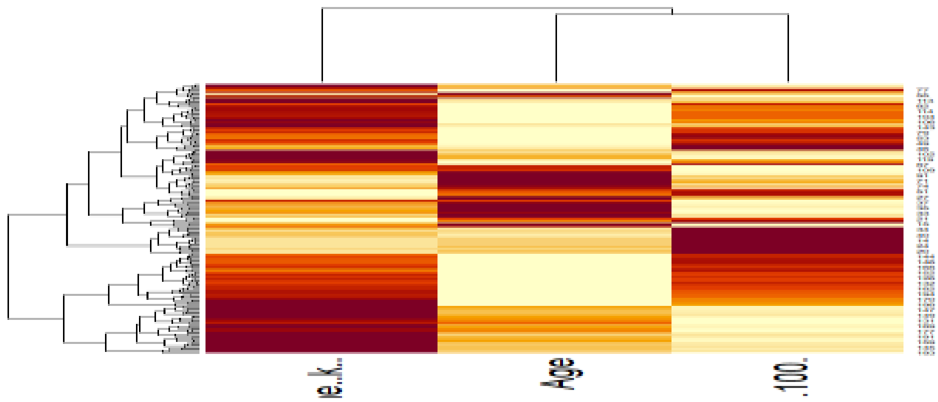
3) Applying k-means clustering on this data (Shopping_Customers.csv) with k=2 and plotting the result of k-means. Cluster1 has 115 points and cluster 2 has 85 points. Save the result of clusters in data frame. Plotting the data with clusters by two method. The first method by function (clusplot) and second method in the code file.

```
#Apply K-Means Clustering Algorithm on the data set
set.seed(100)
Cluster_kmean <- kmeans(Data1, 2, nstart = 20)

#Tabulate the cross distribution
table(Cluster_kmean$cluster)


  1    2
115   85
```



4) Plotting heat map to find the most correlated attribute by two different ways (the second way in the code file).The first way (heatmap function):

```
#heat map
x <- Data1
x <- as.matrix(x)
heatmap(x)
```

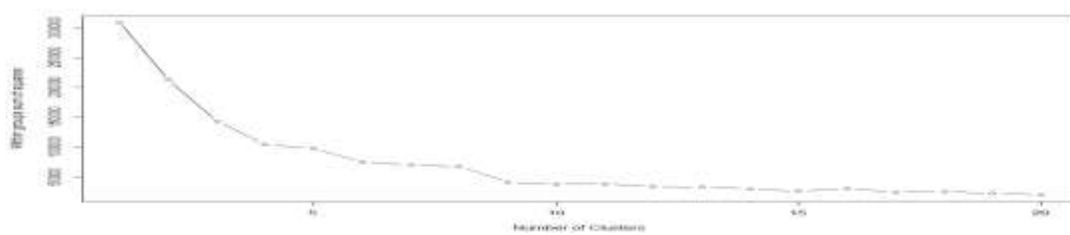


The Age and spending.score..1.100. are related.

5) To obtain the best k in (k-means) by elbow method and Silhouette Coefficient method.

Elbow method and plot within groups sum of squares with number of cluster:

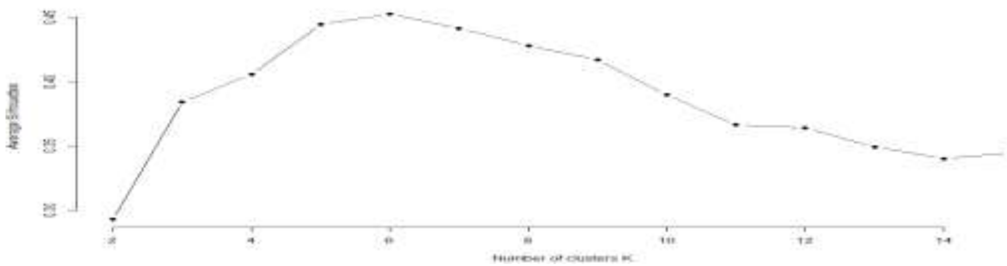
```
# Elbow Curve
wss <- (nrow(Data1)-1)*sum(apply(Data1,2,var))
for (i in 2:20) {
  wss[i] <- sum(kmeans(Data[,3:5],centers=i)$withinss)
}
plot(1:20, wss, type="b", xlab="Number of Clusters",
     ylab="Within groups sum of squares")
```



The best k in elbow method (k=9) because the within groups sum of squares is constant or little less.

Silhouette Coefficient method by Average silhouette and plot Average silhouette with number of cluster:

```
#silhouette function
library(cluster)
avg_sil <- function(k) {
  km.res <- kmeans(Data1, centers = k, nstart = 25)
  ss <- silhouette(km.res$cluster, dist(Data1))
 mean(ss[, 3])
}
# Compute and plot Average silhouette for k = 2 to k = 15
k.values <- 2:15
avg_sil_values <- map_dbl(k.values, avg_sil)

plot(k.values, avg_sil_values,
     type = "b", pch = 19, frame = FALSE,
     xlab = "Number of clusters K",
     ylab = "Average Silhouettes")
```

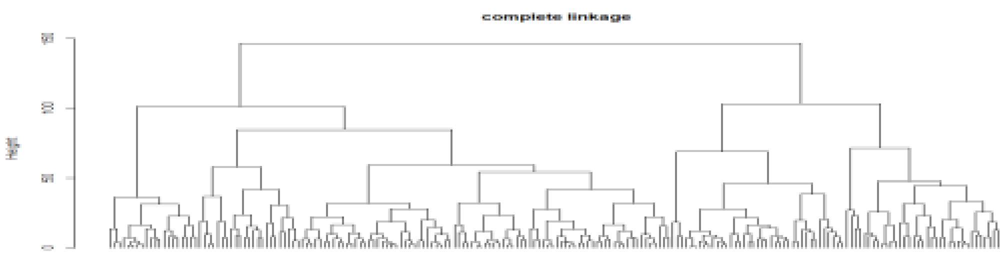


The best k in Silhouette Coefficient method (k=6) because the high of average of silhouette when k=6 (best case).

6) Applying hierarchical clustering (single and complete linkage) on the dataset using Euclidean-distance.

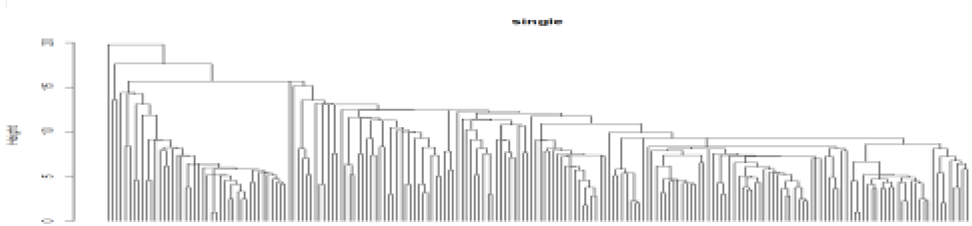
Complete linkage:

```
#apply the hierarchical clustering algorithm and plot dendrogram(complete linkage)
clusters <- hclust(dist(Data1))
plot(clusters,hang = -1,labels = FALSE,main=" complete linkage")
```



Single:

```
#apply the hierarchical clustering algorithm and plot dendrogram (single)
clusters1 <- hclust(dist(Data1),method = "single")
plot(clusters1,hang = -1,labels = FALSE,main = "single")
```



After applying hierarchical clustering (single and complete linkage) on the dataset the result of single is more different complete linkage because the complete linkage of hierarchical clustering choose the maximum distance between their individual points but the single of hierarchical clustering choose the minimum distance between their individual points.

Part 2: Use hierarchical agglomerative clustering with single linkage using Euclidean distance.

Euclidean distance = $((x_1\text{-}x_2)^2 + (y_1\text{-}y_2)^2)^{1/2}$

1) Source data and calculate Euclidean distance.

cluster	10	20	40	80	85	121	160	168	195
10	0	10	30	70	75	111	150	158	185
20	10	0	20	60	65	101	140	148	175
40	30	20	0	40	45	81	120	128	155
80	70	60	40	0	5	41	80	86	115
85	75	65	45	5	0	36	75	83	110
121	111	101	81	41	36	0	39	47	74
160	150	140	120	80	75	39	0	8	35
168	158	148	128	86	83	47	8	0	27
195	185	175	155	115	110	74	35	27	0

2) First step in single linkage **(choose the small distance and combine the two cluster in one cell in every iteration) compute (80-85). Small distance = 5**.

cluster	10	20	40	80-85	121	160	168	195
10	0	10	30	70	111	150	158	185
20	10	0	20	60	101	140	148	175
40	30	20	0	40	81	120	128	155
80-85	70	60	40	0	36	75	83	110
121	111	101	81	36	0	39	47	74
160	150	140	120	75	39	0	8	35
168	158	148	128	83	47	8	0	27
195	185	175	155	110	74	35	27	0

3) Second step in single linkage **compute (160-168). Small distance = 8**.

cluster	10	20	40	80-85	121	160-168	195
10	0	10	30	70	111	150	185
20	10	0	20	60	101	140	175
40	30	20	0	40	81	120	155
80-85	70	60	40	0	36	75	110
121	111	101	81	36	0	39	74
160-168	150	140	120	75	39	0	27
195	185	175	155	110	74	27	0

4) Third step in single linkage **compute (10-20). Small distance = 10**.

cluster	10-20	40	80-85	121	160-168	195

10-20	0	20	60	101	140	175
40	20	0	40	81	120	155
80-85	60	40	0	36	75	110
121	101	81	36	0	39	74
160-168	140	120	75	39	0	27
195	175	155	110	74	27	0

5) Fourth step in single linkage **compute (10-20-40). Small distance = 20**.

cluster	10-20-40	80-85	121	160-168	195
10-20-40	0	40	81	120	155
80-85	40	0	36	75	110
121	81	36	0	39	74
160-168	120	75	39	0	27
195	155	110	74	27	0

6) Fifth step in single linkage **compute (160-168-195). Small distance = 27**.

cluster	10-20-40	80-85	121	160-168-195
10-20-40	0	40	81	120
80-85	40	0	36	75
121	81	36	0	39
160-168-195	120	75	39	0

7) Sixth step in single linkage **compute (80-85-121). Small distance = 36**.

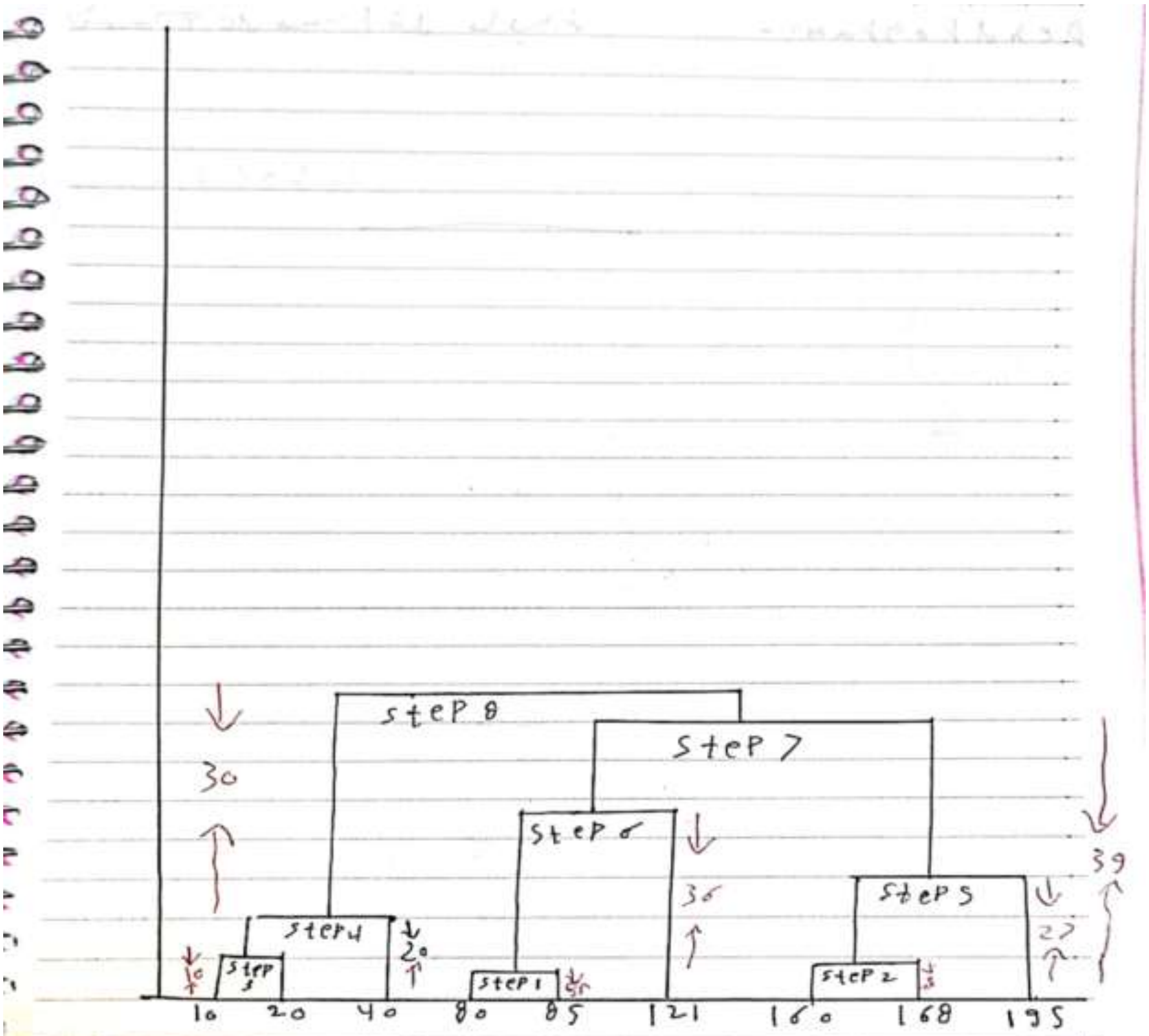
cluster	10-20-40	80-85-121	160-168-195
10-20-40	0	40	120
80-85-121	40	0	39
160-168-195	120	39	0

8) Seventh step in single linkage **compute (80-85-121-160-168-195). Small distance = 39**.

cluster	10-20-40	80-85-121-160-168-195
10-20-40	0	40
80-85-121-160-168-195	40	0

Last iteration all become connect in dendrogram. Finally data was combined to two cluster.

Dendrogram : draw the cluster in the X axis with the hight in th Y axis from Useing hierarchical agglomerative (single linkage).



Conclusion

In conclusion, this report could be summarized by Applying clean the data and more models (decision tree, XGboost, neural network, k-means, hierarchical clustering) in classification and clustering in different data by using R.   .