

Fundamentals of Applied Data Science with R	
	Classification & Clustering
	Individual Assignment 2 Report
Name	Ahmed Shehata Mahmoud AboMoustafa
E-Mail	aabom018@uOttawa.ca

Part 1: Classification

- 1) - Check severity of class imbalance
 - Check missing values
 - Remove Missing values
 - Drop customerID columns from dataset
 - Convert categorical variables

```
# Check severity of class imbalance
round(prop.table(table(churn_df$Churn)), 2)

# Check missing values
anyNA(churn_df)
sum(is.na(churn_df))

# find columns with NA
list_na <- colnames(churn_df)[apply(churn_df, 2, anyNA) ]
list_na

# Remove Missing values
clean_churn <- churn_df %>%
  na.omit()

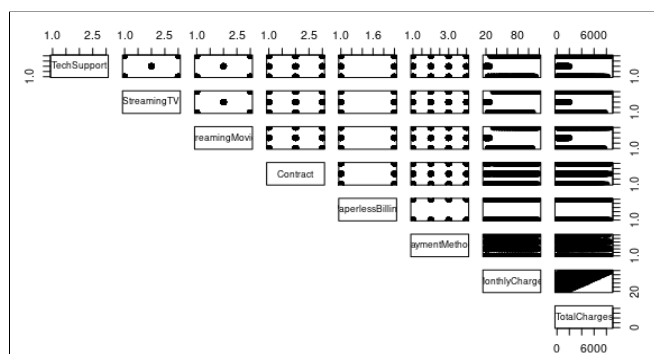
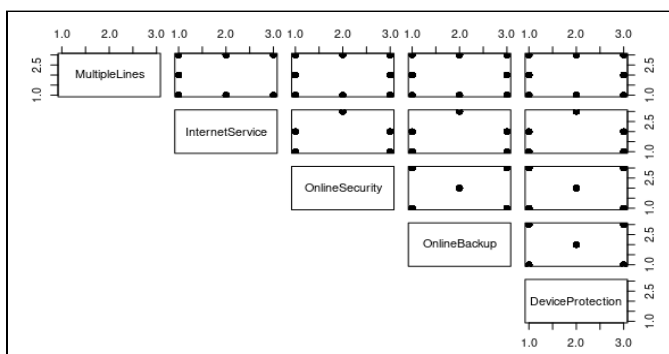
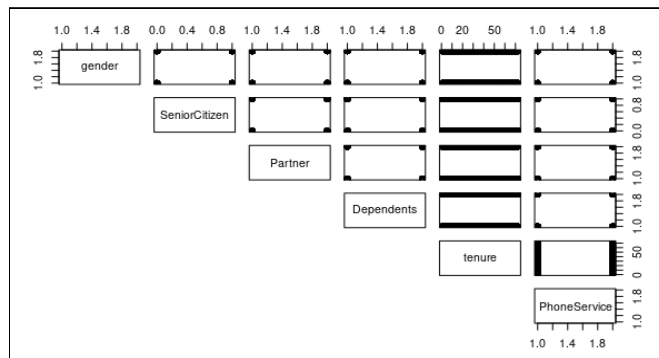
anyNA(clean_churn)
sum(is.na(clean_churn))
```

```
No  Yes
0.73 0.27
> # Check missing values
> anyNA(churn_df)
[1] TRUE
> sum(is.na(churn_df))
[1] 11
> # find columns with NA
> list_na <- colnames(churn_df)[apply(churn_df, 2, anyNA) ]
> list_na
[1] "TotalCharges"
> # Remove Missing values
> clean_churn <- churn_df %>%
+   na.omit()
> anyNA(clean_churn)
[1] FALSE
> sum(is.na(clean_churn))
[1] 0
```

2) - Scatter plot Matrix

Here we see that there are binary features that have only 2 values and there is a high correlation between MonthlyCharges and TotalCharges.

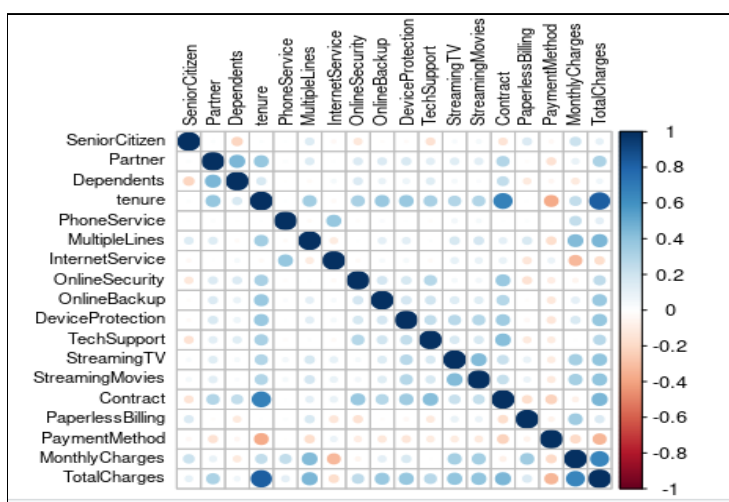
```
### Scatter Matrix
pairs(churn_data_df[,1:6], pch = 19, lower.panel = NULL)
pairs(churn_data_df[,7:11], pch = 19, lower.panel = NULL)
pairs(churn_data_df[,12:19], pch = 19, lower.panel = NULL)
```



- Correlation Matrix

- Here we see the correlation between attributes and there is a high positive correlation between TotalCharges and Tenure, MonthlyCharges and TotalCharges.

```
corrplot(cor(churn_data_df[,2:19]),
  method = "circle",
  type = "full",
  diag = TRUE,
  tl.col = "black",
  bg = "white",
  title = "",
  col = NULL,
  tl.cex = 0.7,
  cl.ratio = 0.2)
```



3) 1. - Here, I split the data to 80% train and 20% test

- Convert train and test to dataframe
- After that, I build first decision tree

Data Splitting	DT Model
<pre> # 3.1 80% of the sample size smp_size <- floor(0.80 * nrow(clean_churn)) ## set the seed to make your partition reproducible set.seed(123) train_ind <- sample(seq_len(nrow(clean_churn)), size = smp_size) train1 <- clean_churn[train_ind,] # 5625 row test1 <- clean_churn[-train_ind,] # 1407 row # convert matrix to dataframe for model train_df1 <- as.data.frame(train1) test_df1 <- as.data.frame(test1) </pre>	

3. After that, I evaluate the performance using confusion matrix and ROC

Accuracy: 0.8088	AUC: 0.690	Precision: 0.9367	Recall: 0.8280
------------------	------------	-------------------	----------------

Confusion Matrix	ROC Curve

4. From This evaluation, we conclude that the decision tree model is not good enough in prediction of the true negative points as the recall is 0.82, and overall accuracy is not high, so the model needs some hyperparameter tuning.

- 4) The DT here was first split based on contract feature, and it divided to two branches:
- If it's higher than or equal to 2 contracts, it goes to check the next split.
 - And if it's below 2 contracts, DT classify it as 1
- Then it split by OnlineSecurity, which is the next feature for splitting and it also divided into 2 branches, etc.
- Rules:**

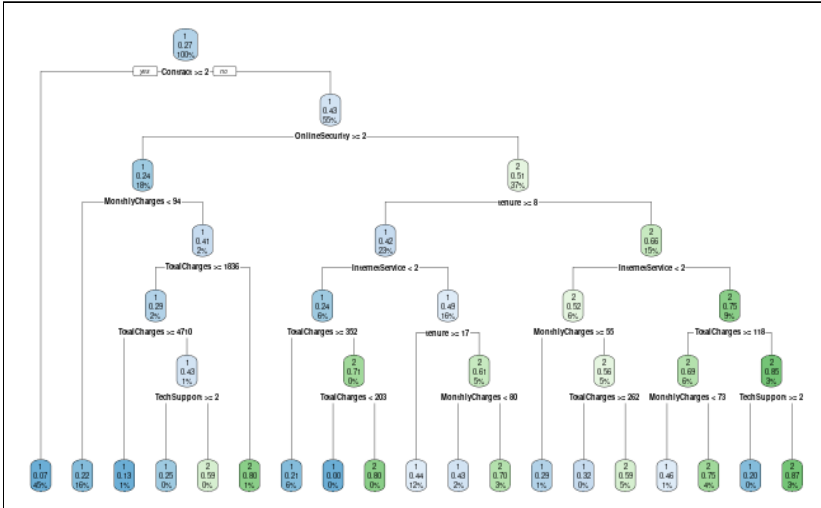
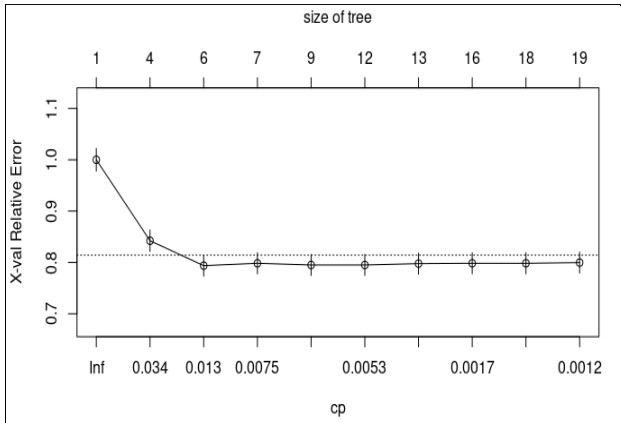
Rules		label
Rule 1	Contact ≥ 2	1
Rule 2	Contact < 2 & Online Security ≥ 2	1
Rule 3	Contact < 2 & Online Security < 2 & tenure < 8 & InternetService < 2	2
Rule 3	Contact < 2 & Online Security < 2 & tenure < 8 & InternetService ≥ 2 & TotalCharges ≥ 103	2

5) Improve decision tree

- Way 01 to improve decision tree accuracy

1. I choose the most important features then I build the model with max depth:6, min split:2, splitting using **gini** and pruning using $cp = 0.001$

Select Features	DT Model
<pre>##### Way 01 to improve decision tree accuracy important_var_df <- clean_churn[, c('tenure', 'Contract', 'TechSupport', 'OnlineSecurity', 'MonthlyCharges', 'OnlineBackup', 'InternetService', 'TotalCharges', 'Churn')]</pre>	<pre># specify method as class since we are dealing with classification DT_model2 <- rpart(Churn ~ ., data = train_df2, method = "class", parms = list(split = 'gini'), control = rpart.control(cp = 0.001, maxdepth = 6, minsplit = 2))</pre>

DT	Error
	<p>The Error decreased by increasing the tree size until a specific point, there is no improvement.</p> 

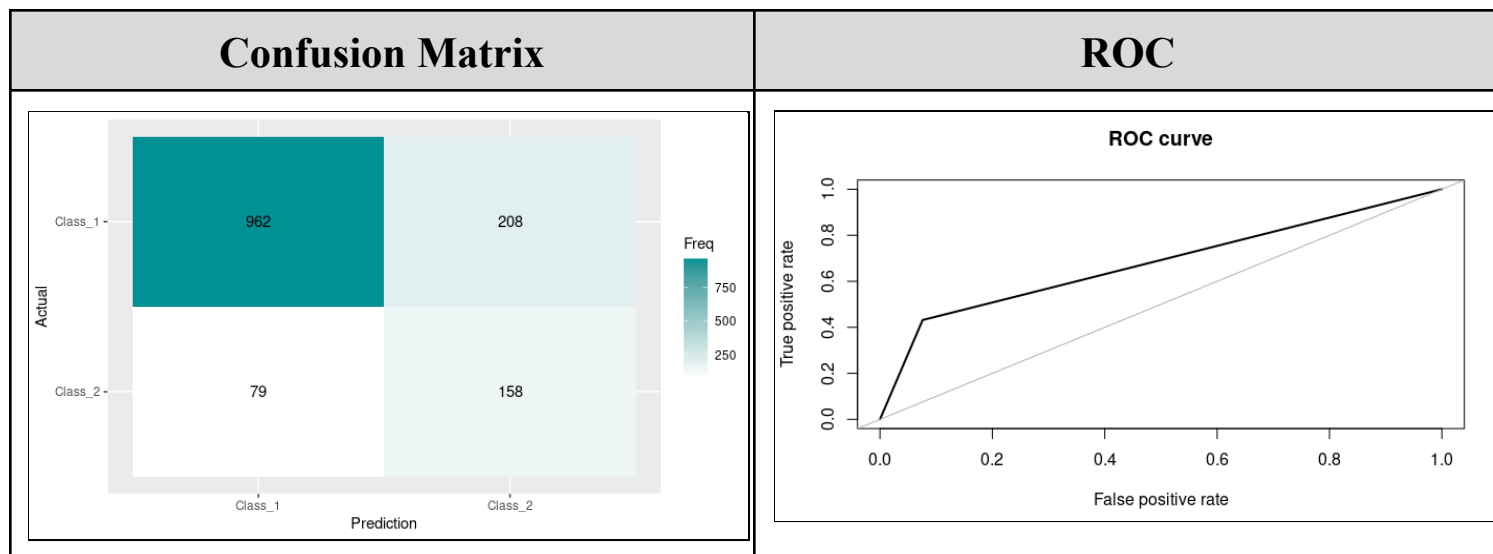
3. After that, I evaluate the performance using confusion matrix and ROC

Accuracy: 0.796

AUC: 0.678

Precision: 0.9241

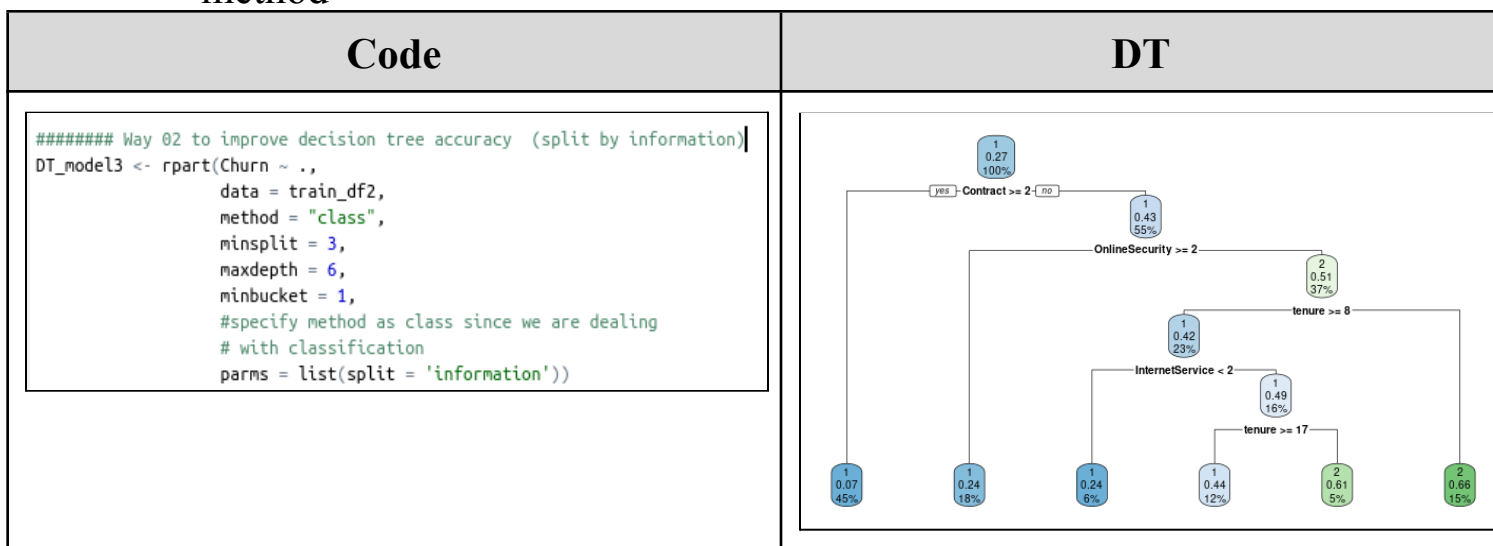
Recall: 0.8222



From that we can conclude that max depth, splitting with gini and pruning after splitting have no effect here, as the accuracy decreased

- Way 02 to improve decision tree accuracy

1. I build this DT model based on all features and use **information** as splitting method



2. After that, I evaluate the performance using confusion matrix and ROC

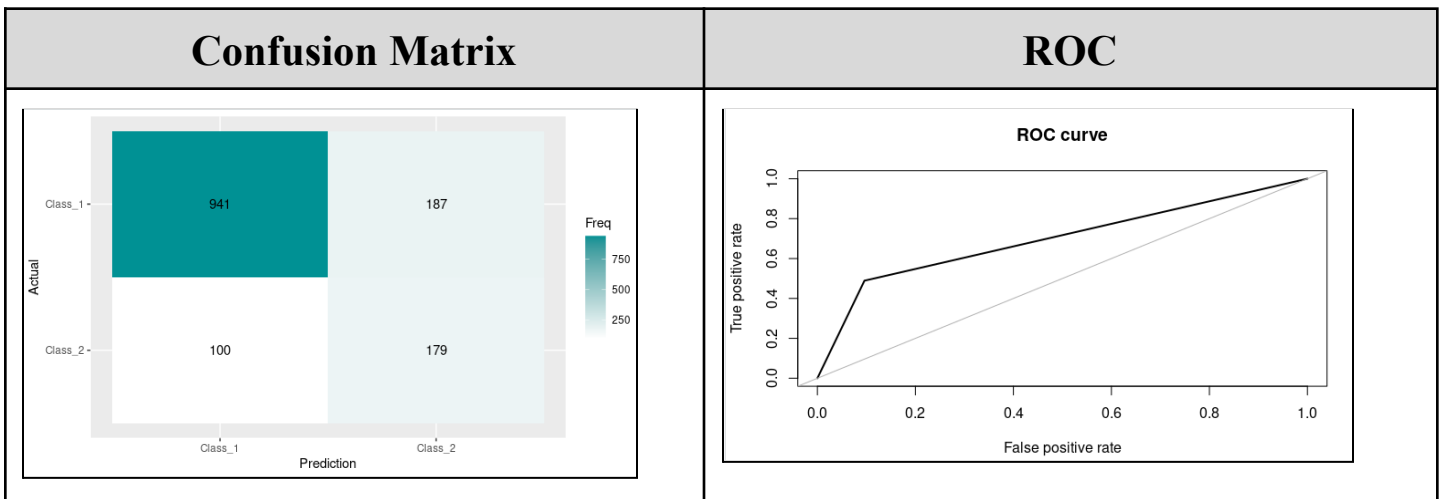
Accuracy: 0.796

AUC: 0.697

Precision: 0.9039

Recall: 0.8342

- We find that precision decreases and recall increases, so this model predicts lower true negatives than the latest one.
- But overall, not a bigger improvement when using information as a splitting method.



6) XGBoost

Convert the training and testing sets into Matrix and convert y label as a factor

```
# 6.1 Convert the training and testing sets into Matrix|
X_train = as.matrix(train_df1 %>% select(-Churn))
# y_train = as.matrix(train_df1$Churn)
y_train <- factor(train_df1$Churn, labels = c("2", "1"))

X_test = as.matrix(test_df1 %>% select(-Churn))
y_test = factor(test_df1$Churn, labels = c("2", "1"))
```

- Specify cross-validation method and number of folds to 10 with repeats to 3
- Specify grid search technique to search for best parameters for the model

```
# 6.2 Specify cross-validation method and number of folds
xgb_trcontrol = trainControl( method = "cv",
                              number = 10,
                              repeats = 3,
                              allowParallel = TRUE,
                              verboseIter = FALSE,
                              returnData = FALSE,
                              sampling = "smote")

# 6.3 grid space to search for the best hyperparameters
xgb_grid <- expand.grid(gamma = c(1, 0),
                       nrounds = c(100, 200), # n_estimators
                       max_depth = c(5, 10, 15, 20, 25),
                       colsample_bytree = seq(0.5, 0.9, length.out = 5),
                       eta = c(0.1, 0.01, 0.001, 0.0001),
                       min_child_weight = 1,
                       subsample = 1)
```

- Train your model using **xgbtree** method and find the best parameters

```
# 6.4 train your model
set.seed(0)
xgb_model = train(x= X_train,
                  y= y_train,
                  trControl = xgb_trcontrol,
                  tuneGrid = xgb_grid,
                  method = "xgbTree")
```

Best parameters

nrounds	max_depth	eta	gamma	colsample_bytree	min_child_weight	subsample
23	100	10	1e-04	0	0.6	1

- After that, I evaluate the performance using confusion matrix and ROC

Accuracy: 0.823

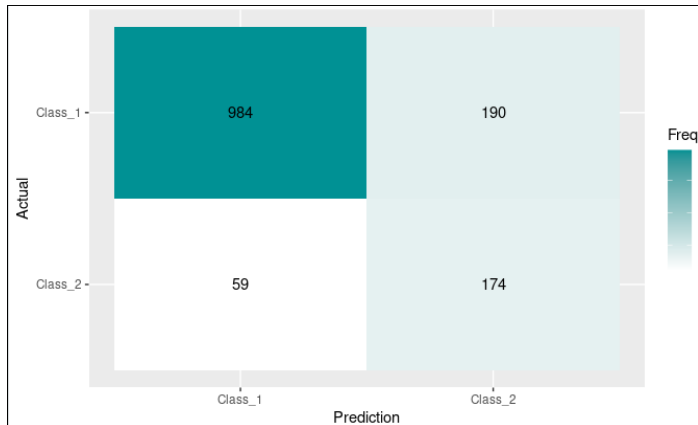
AUC: 0.711

Precision: 0.9434

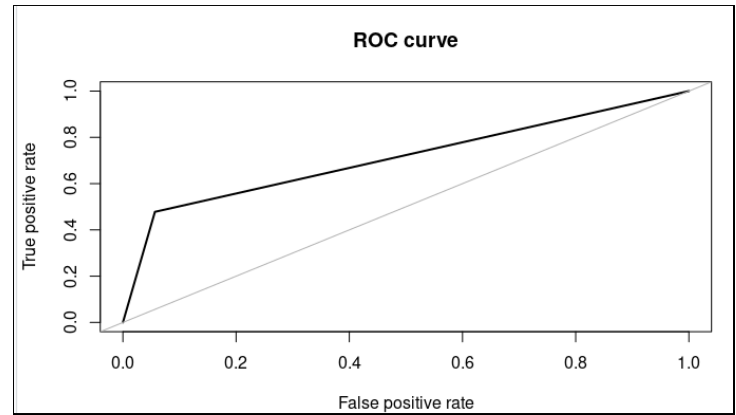
Recall: 0.8382

We find that the XGBoost is better than decision tree as the accuracy metrics increases

Confusion Matrix



ROC



7) Neural Network

1. Neural 1

- Build NN based on important feature dataset and split it to train and test dataset, then I scale the x train and x test
- Then Build NN using activation function **relu**, dropout **0.2**, optimizer **rmsprop** then, I train my NN based on scaled data

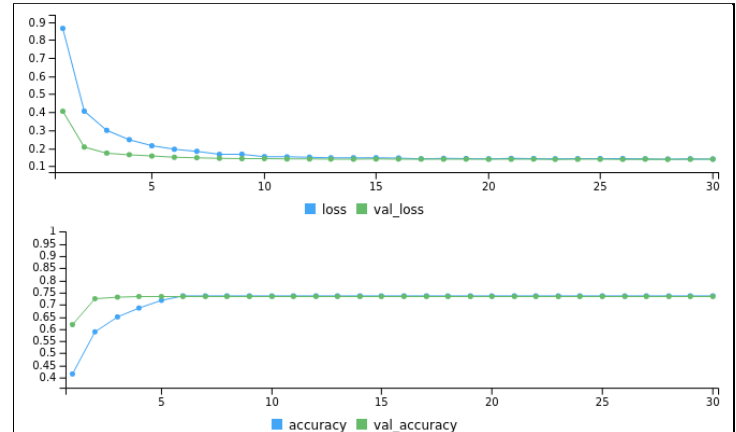
Model

```
# 7.4 Model Creation
model1 <- keras_model_sequential()
model1 %>%
  layer_dense(units = 5, activation = 'relu', input_shape = c(8)) %>%
  layer_dropout(rate=0.2) %>%
  layer_dense(units = 1)

# 7.5 Model Compilation
model1 %>% compile(loss = 'mse',
  optimizer = 'rmsprop',
  metrics = 'accuracy')

# 7.6 Model Training
nn_model1 <- model1 %>%
  fit(X_train1,
    y_train1,
    epochs = 30,
    batch_size = 32,
    validation_split = 0.2)
```

Training



- After that, I evaluate the performance using confusion matrix and ROC

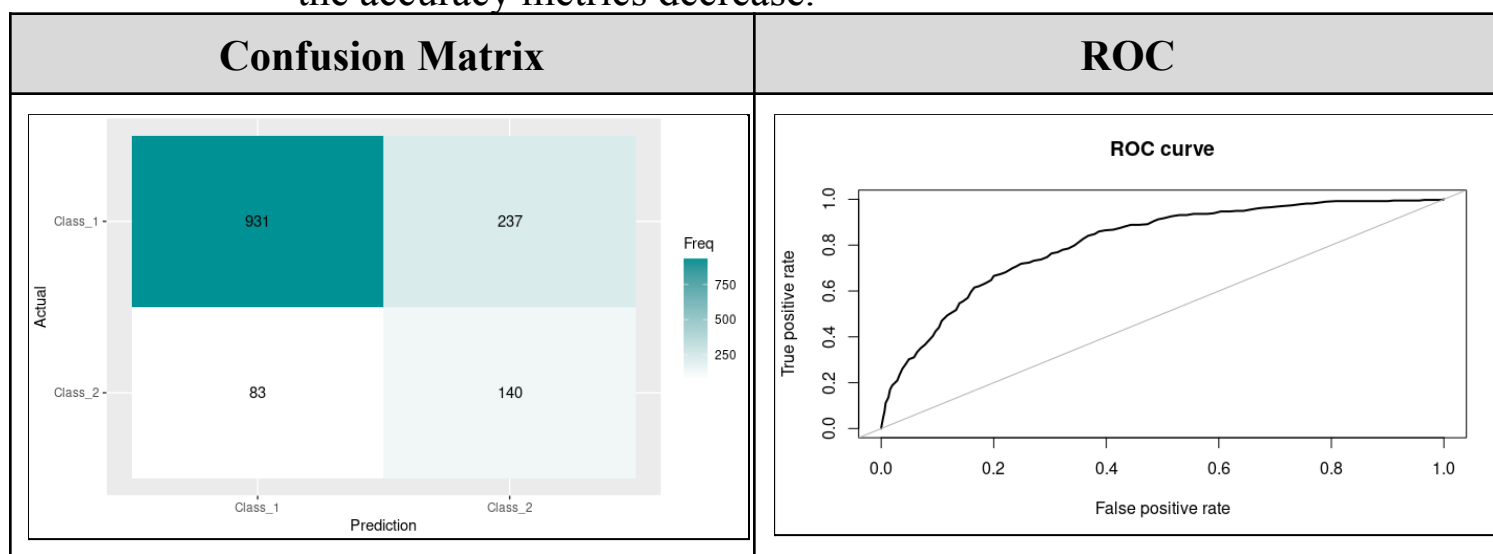
Accuracy: 0.7699

AUC: 0.810

Precision: 0.9181

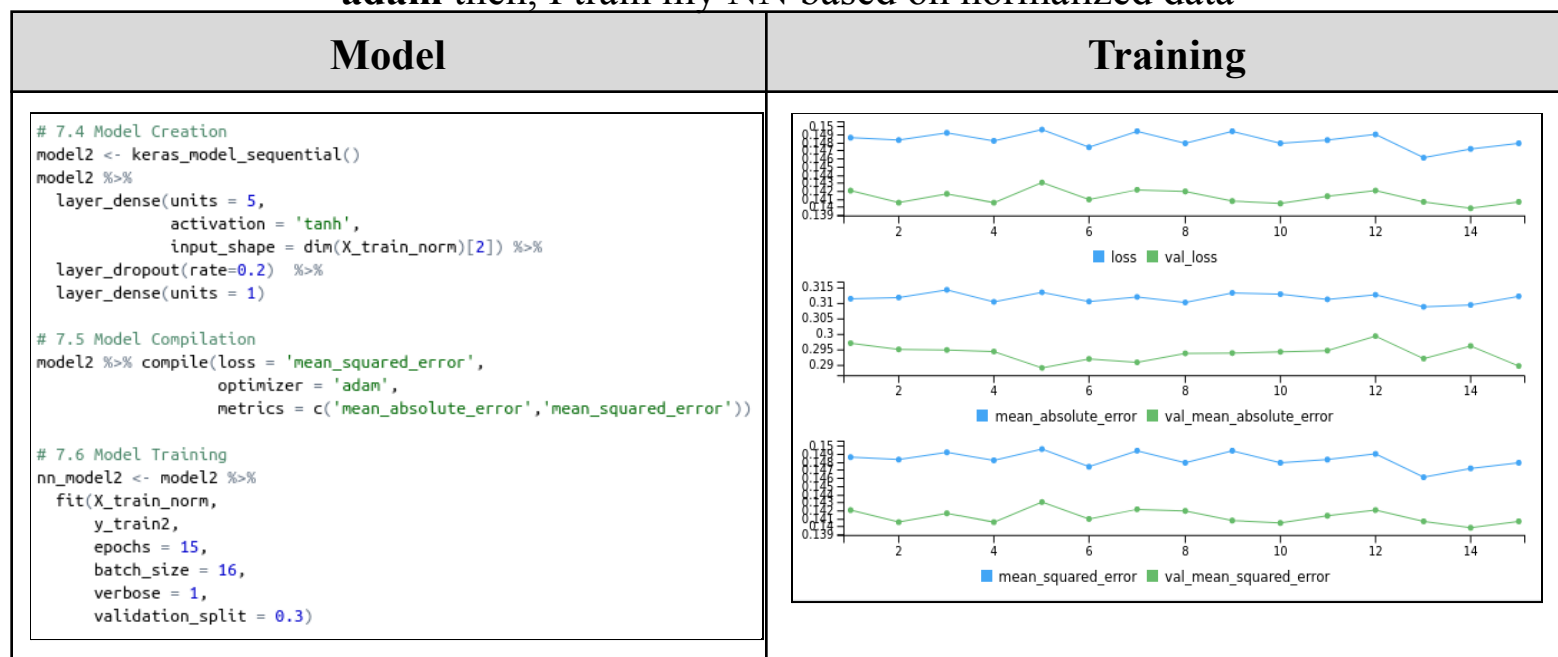
Recall: 0.7971

- We find that the NN is worse than the decision tree and XGboost as the accuracy metrics decrease.



2. Neural 2

- Build NN based on Clean Churn dataset and split it to train and test dataset, then I normalize the x train and x test
- Then Build NN using activation function **tanh**, dropout **0.2**, optimizer **adam** then, I train my NN based on normalized data



- After that, I evaluate the performance using confusion matrix and ROC

Accuracy: 0.7913

AUC: 0.830

Precision: 0.9099

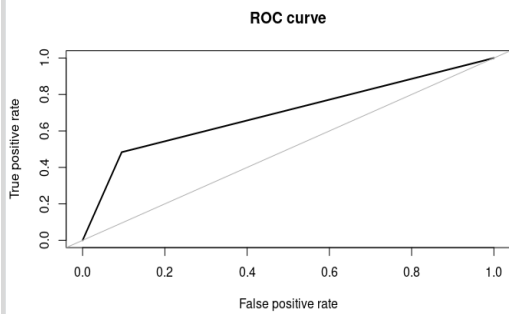
Recall: 0.8241

- We find that the second NN is better than the first as the accuracy metrics increase according to recall as the dataset increases in the train phase which affects the evaluation.

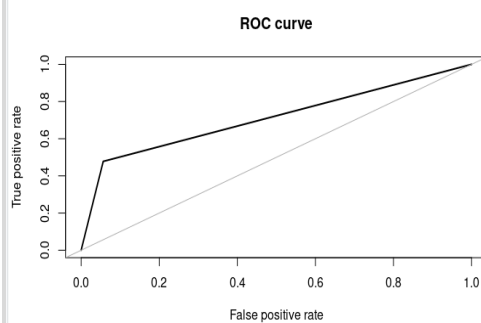
8) ROC Curve of Our Models

From our models we see that best model according to accuracy is **XGboost** and according to ROC and AUC is **Second NN** which is 0.830

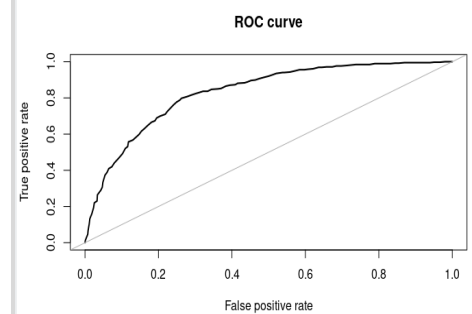
DT ROC



XGBoost ROC



NN ROC



Part 2: Clustering

- 1) Read the dataset and perform some preprocessing on it as check null values, convert target values to 0 and 1, and change feature names to meaningful names

```
# Read the Data
sc_df <- read.csv("Datasets/Shopping_Customers.csv",header=TRUE)
View(sc_df)
str(sc_df)

any(is.na(sc_df))

sc_df$Gender <- ifelse(sc_df$Gender=="Male", 1, 0)
sc_df$Gender <- as.integer(sc_df$Gender)
# Rename columns 4 and 5
names(sc_df)[4] <- 'Annual_Income'
names(sc_df)[5] <- 'Spending_Score'
```

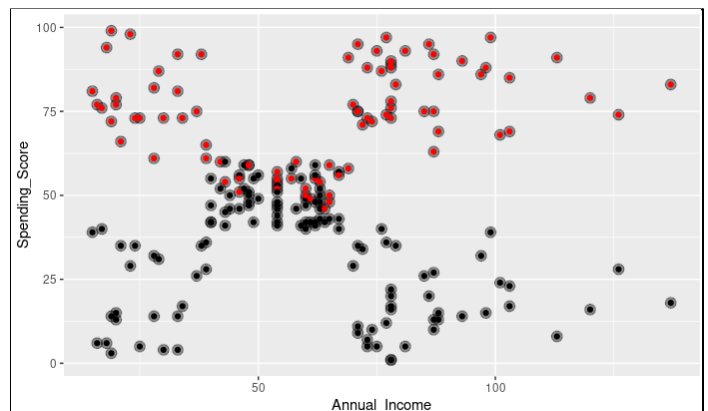
- 2) I Build K-means features from 3 to 5 because 1 and 2 do not affect

K-means

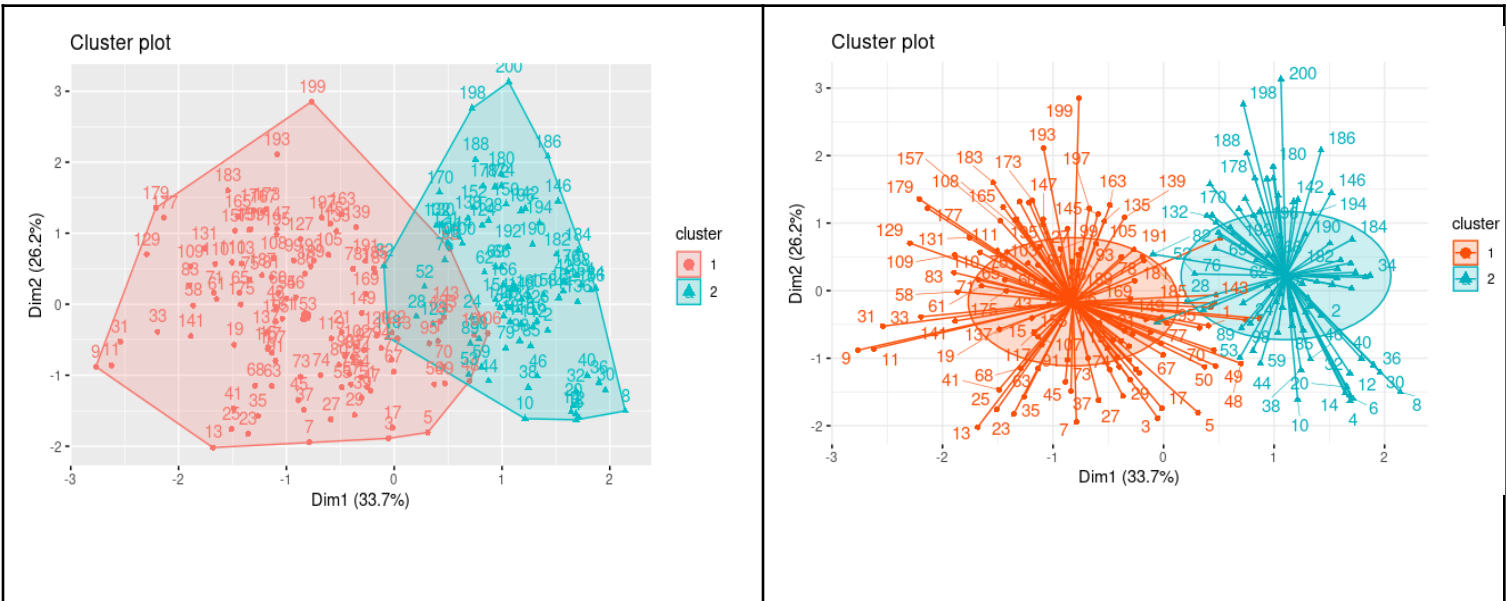
```
# Run k-means on sc_df
km <- kmeans(sc_df[,3:5], 2, nstart = 1)

km$cluster <- factor(km$cluster)
ggplot(sc_df, aes(Annual_Income, Spending_Score)) +
  geom_point(alpha = 0.4, size = 3.5) + geom_point(col = km$cluster) +
  scale_color_manual(values = c('black', 'red'))
x <- sc_df[,2:5]
x <- as.matrix(x)
```

Result

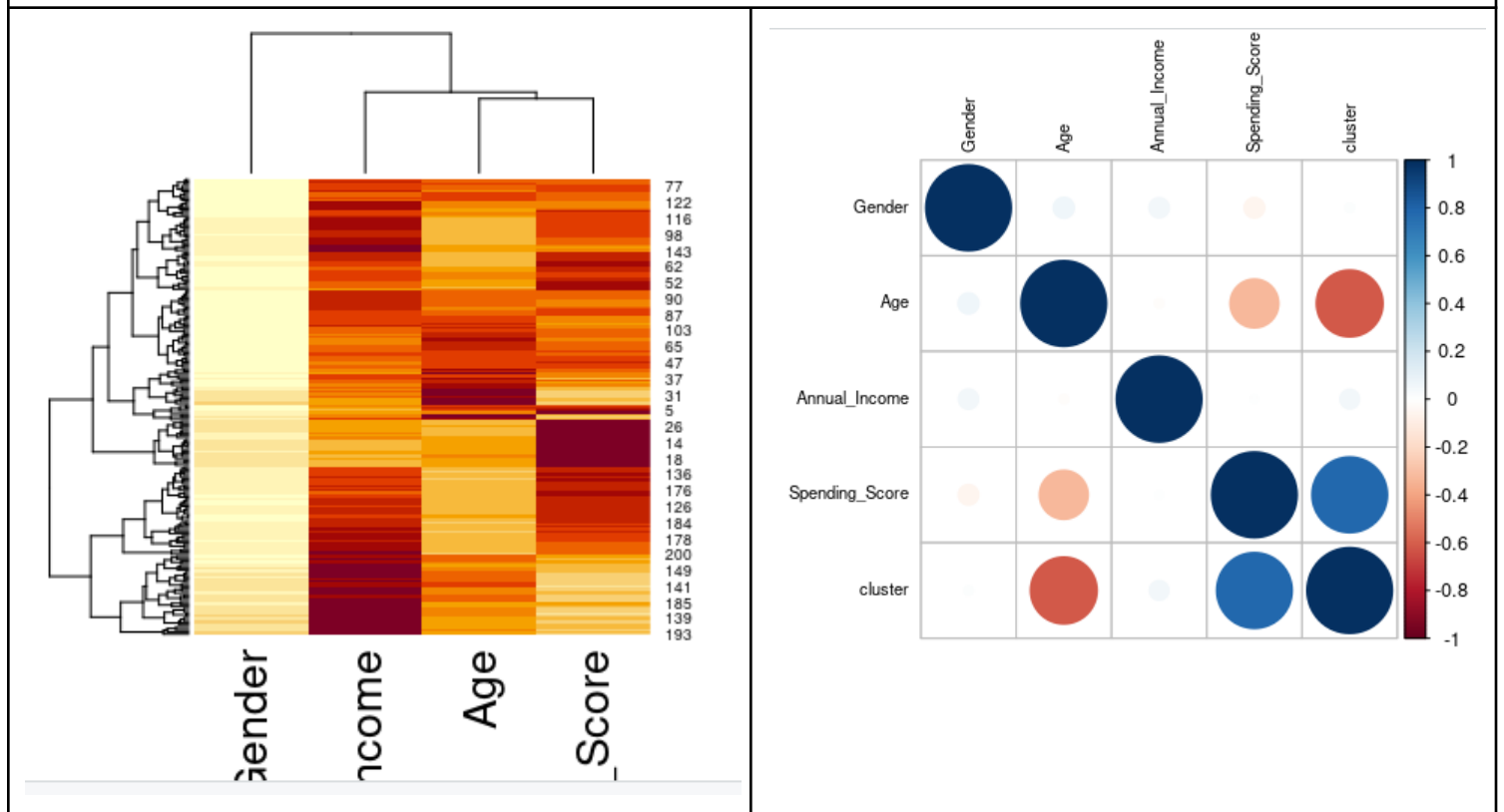


Here, We see that there are small overlapping points between the two clusters



Then, To determine the most correlated attributes to the cluster I use heatmap on the features with predicted clusters

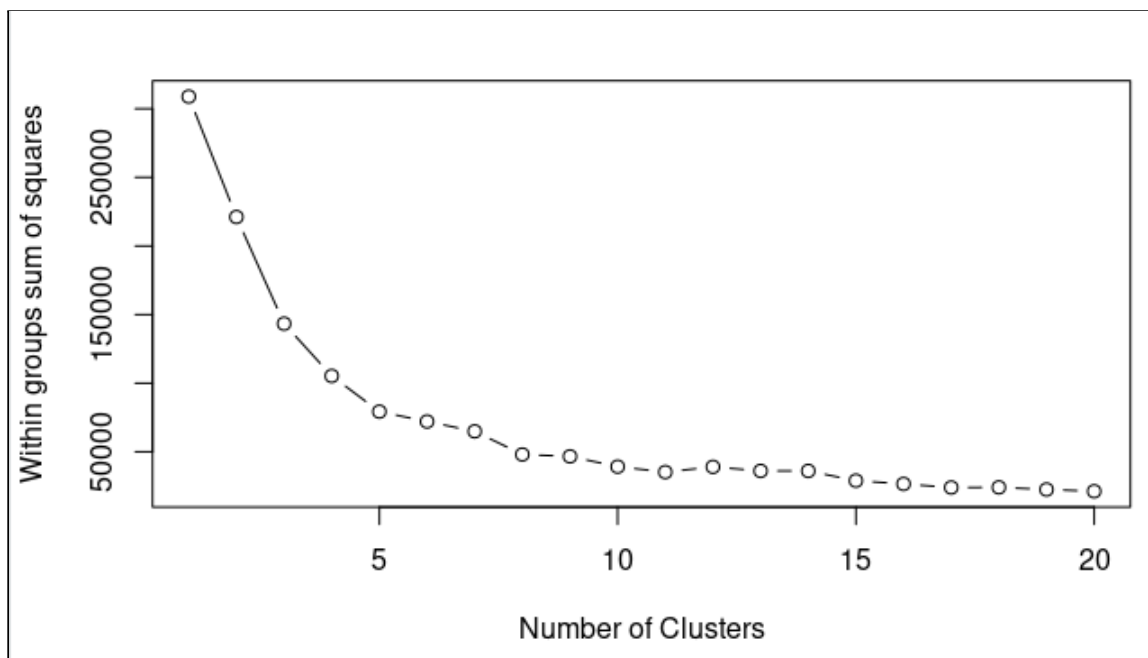
Here, we see that the most correlated attribute is **Spending Score and age**



Then I save the predicted cluster labels on the dataset

CustomerID	Gender	Age	Annual_Income	Spending_Score	cluster
1	1	19	15	39	1
2	1	21	15	81	2
3	0	20	16	6	1
4	0	23	16	77	2
5	0	31	17	40	1
6	0	22	17	76	2
7	0	35	18	6	1
8	0	23	18	94	2
9	1	64	19	3	1
10	0	30	19	72	2
11	1	67	19	14	1
12	0	35	19	99	2
13	0	58	20	15	1

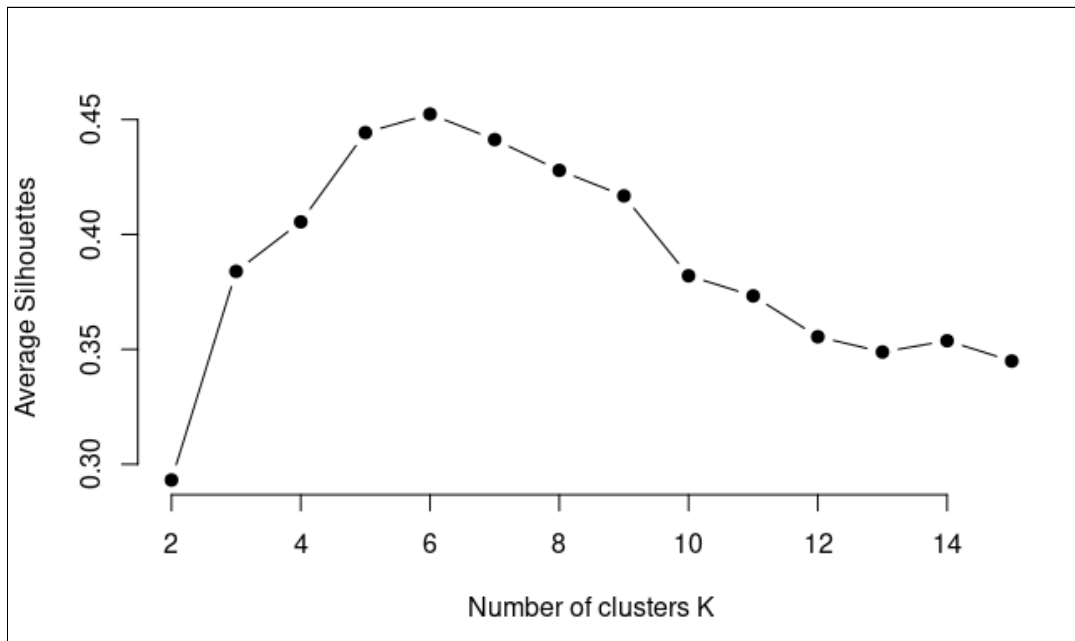
3) Then I apply the elbow method to determine the best k



Here, we see that the best cluster number is **5**

4) Then, I evaluate the quality of the clusters using the Silhouette Coefficient method.

- The Silhouette graph here show increasing in the consistency of the clusters until 6 cluster then the consistency start to decreases, and in our case with 2 clusters, the average silhouette is **0.30** which is accepted

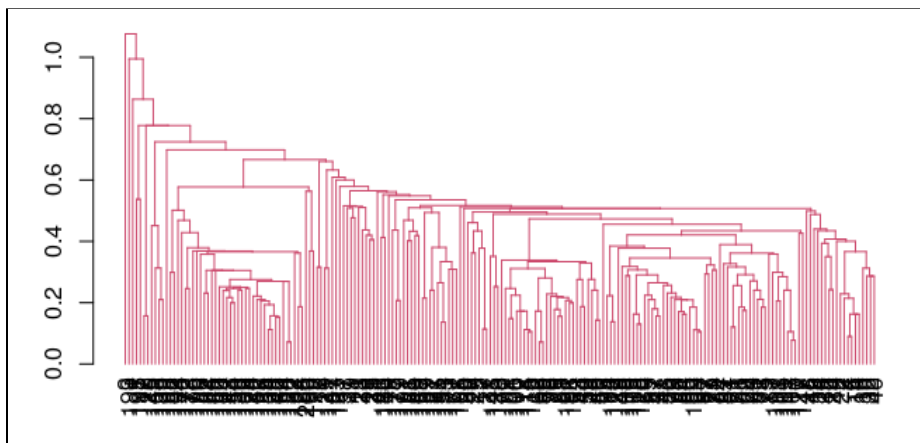


5) Here, I apply hierarchical clustering (single linkage) to the dataset using Euclidean-based distance

- First, I scale the dataset
- Then, I calculate the distance using euclidean distance
- After that, I build hierarchical clustering based on single linkage and plot the result to interpret the model.

```
d <- dist(SC_Scaled, method = "euclidean")
hs <- hclust(d, method = 'single')

#Plot the single dendrogram
com_dend_obj1 <- as.dendrogram(hs)
com_col_dend1 <- color_branches(com_dend_obj1, h = 4)
plot(com_col_dend1)
```



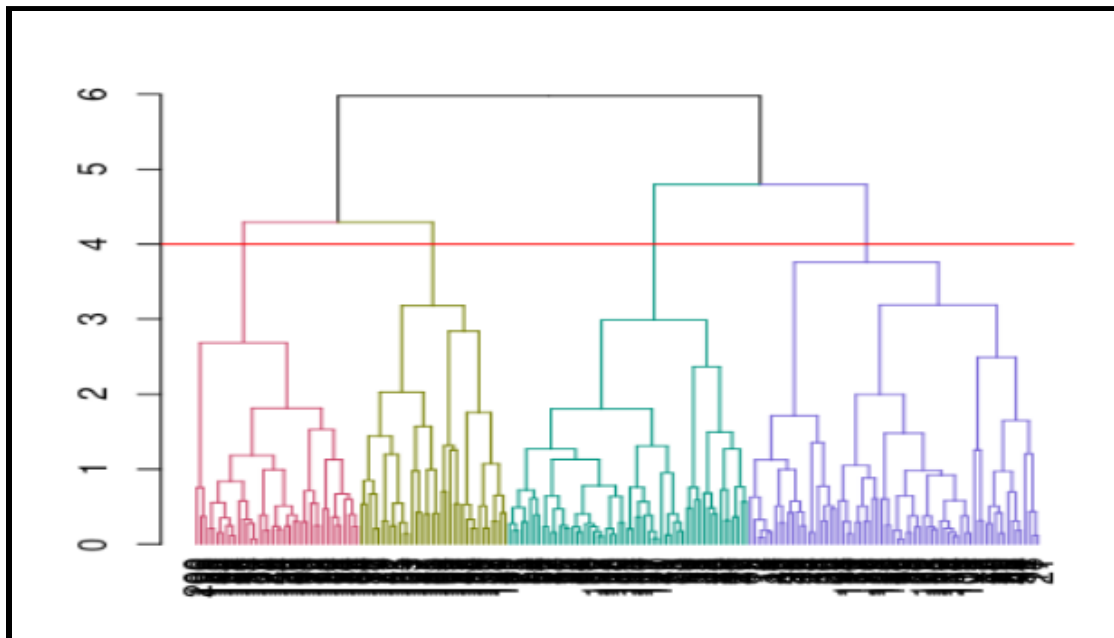
- Then, I build hierarchical clustering based on complete linkage and plot the result to interpret the model

```
hc = hclust(dist(SC_Scaled), method = 'complete')

#Plot the complete dendrogram
# plot(hc, hang = -1, cex = 0.6)

plot(hc)
com_dend_obj <- as.dendrogram(hc)
com_col_dend <- color_branches(com_dend_obj, h = 4)
plot(com_col_dend)
abline(h = 4, col = 'red')
```

Here, I make red line to cut on 4 clusters



- **In Single Linkage**, the distance between two clusters is the minimum distance between members of the two clusters
- **In Complete Linkage**, the distance between two clusters is the maximum distance between members of the two clusters
- And from the two graphs, the results are different from each other as the cluster structure is different as the method of calc distance in each of them chooses different values.

Part 3: Clustering problem

- STEP #1: Euclidean Distance

	10	20	40	80	85	121	160	168	195
10	0	10	30	70	75	111	150	158	185
20		0	20	60	65	101	140	148	175
40			0	40	45	81	120	128	155
80				0	5	41	80	88	115
85					0	36	75	83	110
121						0	39	47	74
160							0	8	35
168								0	27
195									0

- STEP #2:

1. Min distance is 5 Between (85, 80)

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

2. Min distance is 8 Between (168, 160)

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

3. Min distance is 10 Between (10, 20)

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

4. Min distance is 20 Between [(10, 20), 40]

	[(10, 20), 40]	80,85	121	160, 168	195
[(10, 20), 40]	0	40	81	120	155
80,85		0	36	75	110
121			0	39	74
160, 168				0	27
195					0

5. Min distance is 27 Between [(160, 168), 195]

	[(10, 20), 40]	80,85	121	[(160, 168), 195]
[(10, 20), 40]	0	40	81	120
(80, 85)		0	36	75
121			0	39
[(160, 168), 195]				0

6. Min distance is 36 Between [(80, 85), 121]

	[(10, 20), 40]	[(80,85),121]	[(160, 168), 195]
[(10, 20), 40]	0	40	120
[(80, 85), 121]		0	39
[(160, 168), 195]			0

7. Min distance is 39 Between [(80, 85), 121], [(160, 168), 195]

	[(10, 20), 40]	[(80, 85), 121], [(160, 168), 195]
[(10, 20), 40]	0	40
[(80, 85), 121], [(160, 168), 195]		0

Dendrogram

