DTI 5126: Fundamentals of Data Science (Summer 2023 - Assignment 2)

Part 1: Classification (A. Feature Engineering)

1. count total missing values in each column of data frame

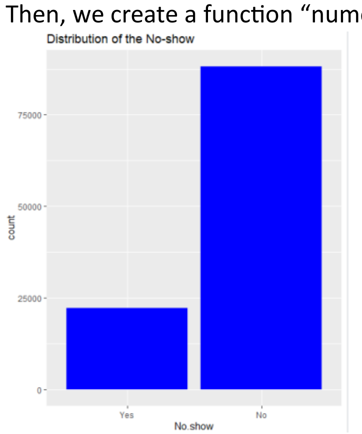
```
> sapply(data, function(x) sum(is.na(x)))
      Gender   ScheduledDay AppointmentDay            Age  Neighbourhood    Scholarship   Hipertension       Diabetes
           0              0              0              3              0              0              0              0
  Alcoholism        Handcap   SMS_received        No.show
           0              0              0              0
```

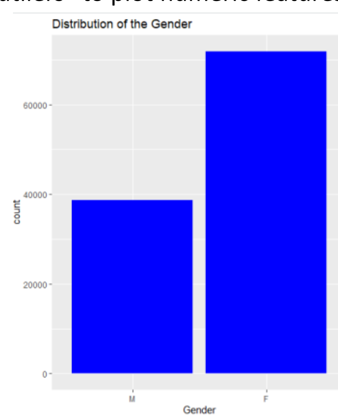
```
> # Remove rows with missing values
> cleaned_df <- na.omit(data)
> # check if missing values are removed
> sum(is.na(cleaned_df)) # ) 0: indicates that there is no missing values
[1] 0
```

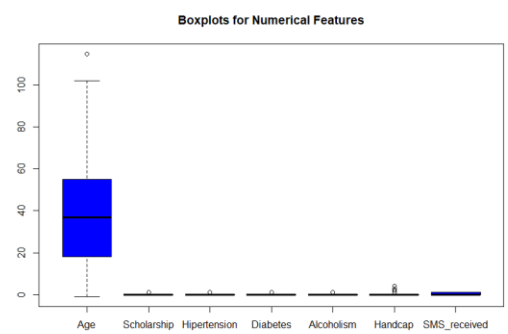
2. Frist, Display types for each attribute

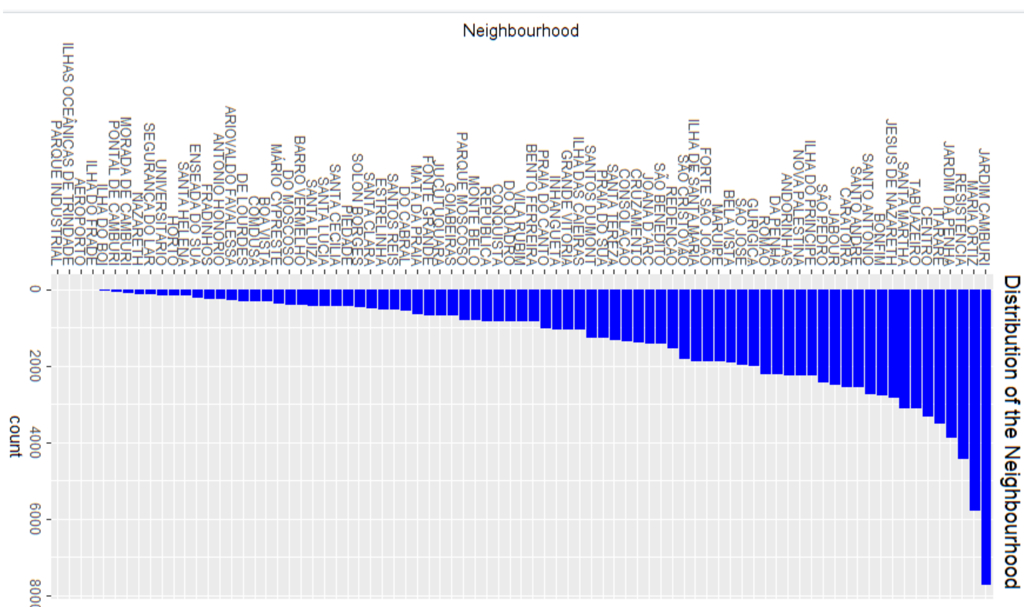
```
sapply(cleaned_df, class)
      Gender   ScheduledDay AppointmentDay            Age  Neighbourhood    Scholarship   Hipertension       Diabetes
 "character"    "character"    "character"      "integer"    "character"      "integer"      "integer"      "integer"
  Alcoholism        Handcap   SMS_received        No.show
   "integer"      "integer"      "integer"    "character"
```

Then, we create a function “numeric _outliers” to plot numeric features, and we also plot categorical features









3. Count the frequency of negative Age feature observations, and remove them

```
> sum(cleaned_df$Age < 0)
[1] 1
> cleaned_df <- cleaned_df[cleaned_df$Age >= 0, ]
> sum(cleaned_df$Age < 0)
[1] 0
```

4. We created AwaitingTime column according to the equation: AwaitingTime= ScheduledDay – AppointmentDay

```
> sum(cleaned_df$AwaitingTime < 0)
[1] 38566
> cleaned_df$AwaitingTime <- abs(cleaned_df$AwaitingTime)
> sum(cleaned_df$AwaitingTime < 0)
[1] 0
```

5. display the first 6 row to ensure that categorical values are converted to numeric

```
> head(cleaned_df)
  Gender        ScheduledDay AppointmentDay Age Neighbourhood Scholarship Hipertension Diabetes Alcoholism Handcap SMS_received No.show
1      2 2016-04-29 18:38:08     2016-04-29  62            78           0            1        0          0       0            0       2
2      1 2016-04-29 16:08:27     2016-04-29  56            78           0            0        0          0       0            0       2
3      2 2016-04-29 16:19:04     2016-04-29  62            32           0            0        0          0       0            0       2
4      2 2016-04-29 17:29:31     2016-04-29   8             6           0            0        0          0       0            0       2
5      2 2016-04-29 16:07:23     2016-04-29  56            78           0            1        1          0       0            0       2
6      2 2016-04-27 08:36:51     2016-04-29  76            38           0            1        0          0       0            0       2
```

6. Separate the date features into date components

```
> cleaned_df$ScheduledDay <- as.POSIXct(cleaned_df$ScheduledDay, format="%Y-%m-%dT%H:%M:%SZ")
> cleaned_df$AppointmentDay <- as.POSIXct(cleaned_df$AppointmentDay, format="%Y-%m-%dT%H:%M:%SZ")
> cleaned_df$ScheduledDay_Year <- format(cleaned_df$ScheduledDay, "%Y")
> cleaned_df$ScheduledDay_Month <- format(cleaned_df$ScheduledDay, "%m")
> cleaned_df$ScheduledDay_Day <- format(cleaned_df$ScheduledDay, "%d")
> cleaned_df$AppointmentDay_Year <- format(cleaned_df$AppointmentDay, "%Y")
> cleaned_df$AppointmentDay_Month <- format(cleaned_df$AppointmentDay, "%m")
> cleaned_df$AppointmentDay_Day <- format(cleaned_df$AppointmentDay, "%d")
> # display random samples of data to show the date components
> sample_df <- cleaned_df %>% sample_n(10)
> sample_df
```

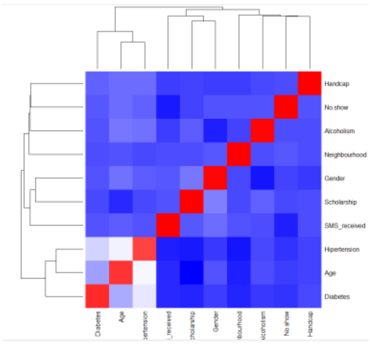
```
   Gender        ScheduledDay AppointmentDay Age Neighbourhood Scholarship Hipertension Diabetes Alcoholism Handcap SMS_received No.show AwaitingTime
1       1 2016-05-02 07:27:28     2016-05-04  73            60           0            1        1          0       0            0       2  145952 secs
2       1 2016-05-20 16:26:54     2016-05-24   7            58           0            0        0          0       0            1       2  286386 secs
3       1 2016-05-13 08:49:58     2016-05-13  52            44           0            0        0          1       0            0       2   31798 secs
4       2 2016-06-01 07:44:41     2016-06-01  13            80           0            0        0          0       0            0       2   27881 secs
5       2 2016-05-18 08:17:21     2016-05-19  43            77           0            0        1          0       0            0       1   56559 secs
6       2 2016-06-03 06:35:53     2016-06-07  62            46           0            1        1          0       0            1       2  321847 secs
7       2 2016-05-19 13:47:57     2016-05-24  73            80           0            0        0          0       0            1       2  382323 secs
8       1 2016-04-06 17:08:02     2016-05-06  65            48           0            0        1          0       0            1       2 2530318 secs
9       2 2016-04-11 13:29:08     2016-05-13   1            31           0            0        0          0       0            0       1 2716252 secs
10      1 2016-06-07 15:53:47     2016-06-07  15            54           0            0        0          0       0            0       2   57227 secs
   ScheduledDay_Year ScheduledDay_Month ScheduledDay_Day AppointmentDay_Year AppointmentDay_Month AppointmentDay_Day
1               2016                 05               02                2016                   05                 04
2               2016                 05               20                2016                   05                 24
3               2016                 05               13                2016                   05                 13
4               2016                 06               01                2016                   06                 01
5               2016                 05               18                2016                   05                 19
6               2016                 06               03                2016                   06                 07
7               2016                 05               19                2016                   05                 24
8               2016                 04               06                2016                   05                 06
9               2016                 04               11                2016                   05                 13
10              2016                 06               07                2016                   06                 07
```

7. Rescale the age feature with a normalizing (e.g., min_max normalization) function and display head(Age)

```
> cleaned_df$Age <- (cleaned_df$Age - min(cleaned_df$Age)) / (max(cleaned_df$Age) - min(cleaned_df$Age))
```

```
> head(cleaned_df$Age)
[1] 0.53913043 0.48695652 0.53913043 0.06956522 0.48695652 0.66086957
```

8. Conduct variability comparison between features using a correlation matrix & drop correlated features




```
> # find attributes that are highly corrected (ideally > 0.5)
> highlyCorrelated <- findCorrelation(cor_matrix, cutoff=0.5)
> print(highlyCorrelated)
[1] 5
> hc = sort(highlyCorrelated)
> hc
[1] 5
> reduced_Data = cleaned_df[ ,-c(hc)]
```

B. Model Development I

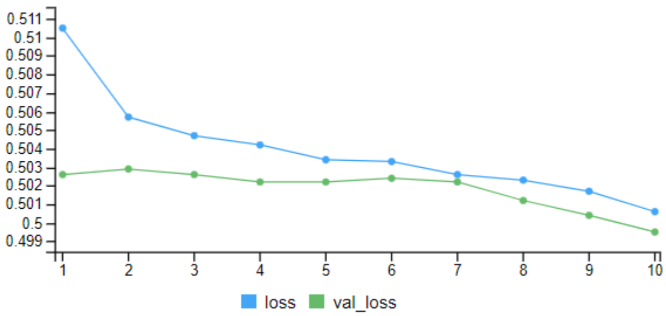
```
> print(paste("Accuracy of the SVM model on the testing dataset is:", accuracy_svm))
[1] "Accuracy of the SVM model on the testing dataset is: 0.786981007728716"
> accuracy_dt <- sum(dt_predict == actual_labels) / length(actual_labels)
> # Print the accuracy of the model on the testing dataset
> print(paste("Accuracy of the Decision Tree model on the testing dataset is:", accuracy_dt))
[1] "Accuracy of the Decision Tree model on the testing dataset is: 0.79830447546582"
```

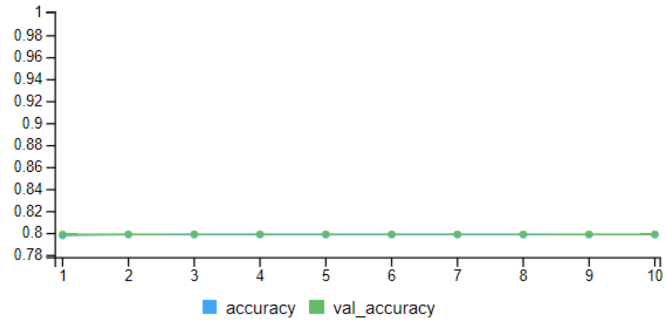
```
> svm_predict <- predict(svm_model, newdata = test)
> print("Confusion Matrix for SVM")
[1] "Confusion Matrix for SVM"
> confusionMatrix(data = svm_predict, reference = test$No.show)
Confusion Matrix and Statistics

          Reference
Prediction   Yes    No
       Yes    33    66
       No   6696 26586

               Accuracy : 0.7974
                 95% CI : (0.7931, 0.8017)
    No Information Rate : 0.7984
    P-Value [Acc > NIR] : 0.6766

                  Kappa : 0.0038

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

            Sensitivity : 0.0049041
            Specificity : 0.9975236
         Pos Pred Value : 0.3333333
         Neg Pred Value : 0.7988102
             Prevalence : 0.2015817
         Detection Rate : 0.0009886
   Detection Prevalence : 0.0029658
      Balanced Accuracy : 0.5012139

       'Positive' Class : Yes
```

```
[1] "Confusion Matrix"
> confusionMatrix(data = dt_predict, reference = test$No.show)
Confusion Matrix and Statistics

          Reference
Prediction   Yes    No
       Yes     0     0
       No   6729 26652

               Accuracy : 0.7984
                 95% CI : (0.7941, 0.8027)
    No Information Rate : 0.7984
    P-Value [Acc > NIR] : 0.5033

                  Kappa : 0

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

            Sensitivity : 0.0000
            Specificity : 1.0000
         Pos Pred Value :    NaN
         Neg Pred Value : 0.7984
             Prevalence : 0.2016
         Detection Rate : 0.0000
   Detection Prevalence : 0.0000
      Balanced Accuracy : 0.5000

       'Positive' Class : Yes
```

C. Model Development II

```
> loss_and_metrics[2]
 accuracy
0.7983045
```





D. Model Evaluation & Comparison

```
detect_accuracy <- function(model, test_data) {

  # Predict the labels of No-show using the trained model
  predicted_probs <- predict(model, x_input_test)
  predicted_labels <- ifelse(predicted_probs > 0.5, 1, 0)

  # Compare predicted labels with actual labels to calculate accuracy
  actual_labels <- test_data$No.show
  accuracy <- sum(predicted_labels == actual_labels) / length(actual_labels)
  return(accuracy)
}
```

1)

2) Tune the model using GridSearchCV

```
# Perform Grid Search for SVM
library(caret)
svm_grid <- expand.grid(C = c(0.1, 1, 10), gamma = c(0.1, 1, 10))
svm_tuned <- train(No.show ~ ., data = train, method = "svmRadial", tuneGrid = svm_grid)
svm_tuned_predicted <- predict(svm_tuned, test)
svm_tuned_accuracy <- calculateAccuracy(test$No.show, svm_tuned_predicted)

# Perform Grid Search for Decision Tree
dt_grid <- expand.grid(cp = seq(0.01, 0.5, by = 0.01))
dt_tuned <- train(No.show ~ ., data = train, method = "rpart", tuneGrid = dt_grid)
dt_tuned_predicted <- predict(dt_tuned, test)
dt_tuned_accuracy <- calculateAccuracy(test$No.show, dt_tuned_predicted)

# Print tuned accuracies
print(paste("Tuned SVM Accuracy:", svm_tuned_accuracy))
print(paste("Tuned Decision Tree Accuracy:", dt_tuned_accuracy))
```

3) The performance of the SVM, Decision tree and Deep Neural Network classifier on the dataset

```
# Evaluate the performance of the SVM classifier
svm_confusion <- confusionMatrix(data = svm_predict, reference = test$No.show)
svm_accuracy <- svm_confusion$overall["Accuracy"]
svm_sensitivity <- svm_confusion$byClass["Sensitivity"]
svm_specificity <- svm_confusion$byClass["Specificity"]

# Evaluate the performance of the decision tree classifier
dt_confusion <- confusionMatrix(data = dt_predict, reference = test$No.show)
dt_accuracy <- dt_confusion$overall["Accuracy"]
dt_sensitivity <- dt_confusion$byClass["Sensitivity"]
dt_specificity <- dt_confusion$byClass["Specificity"]

# Evaluate the performance of the deep neural network classifier
dnn_predicted_probs <- predict(model, x_input_test)
dnn_predicted <- ifelse(dnn_predicted_probs > 0.5, 1, 0)
dnn_confusion <- confusionMatrix(dnn_predicted, test$No.show)
dnn_accuracy <- dnn_confusion$overall["Accuracy"]
dnn_sensitivity <- dnn_confusion$byClass["Sensitivity"]
dnn_specificity <- dnn_confusion$byClass["Specificity"]

# Print the evaluation results
print("Accuracy:")
print(paste("SVM:", svm_accuracy))
print(paste("Decision Tree:", dt_accuracy))
print(paste("Deep Neural Network:", dnn_accuracy))
cat("\n")
print("Sensitivity:")
print(paste("SVM:", svm_sensitivity))
print(paste("Decision Tree:", dt_sensitivity))
print(paste("Deep Neural Network:", dnn_sensitivity))
cat("\n")
print("Specificity:")
print(paste("SVM:", svm_specificity))
print(paste("Decision Tree:", dt_specificity))
print(paste("Deep Neural Network:", dnn_specificity))
```

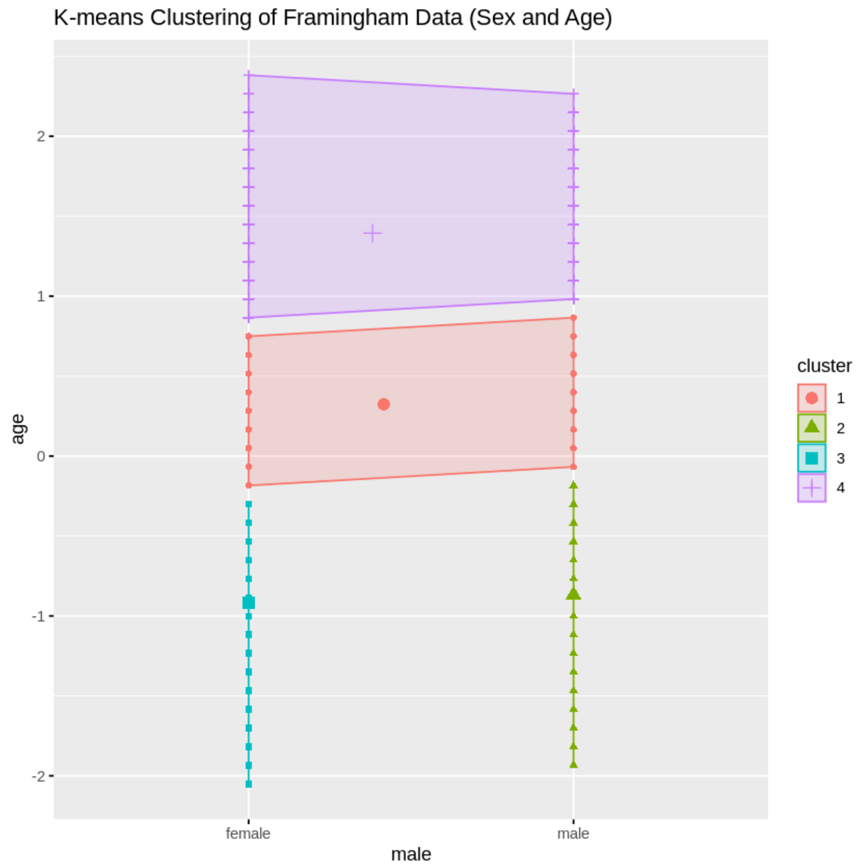
4) Carry out a ROC analysis to compare the performance of the SVM model with the Decision Tree model. Plot the ROC graph of the models.

```
# Predict the class probabilities for SVM and decision tree models
svm_probabilities <- predict(svm_model, test, type = "prob")[, 2]
dt_probabilities <- predict(dt_model, test, type = "prob")[, 2]

# Create prediction objects for SVM and decision tree models
svm_prediction <- prediction(svm_probabilities, test$No.show)
dt_prediction <- prediction(dt_probabilities, test$No.show)

# Calculate the performance metrics for SVM and decision tree models
svm_performance <- performance(svm_prediction, "tpr", "fpr")
dt_performance <- performance(dt_prediction, "tpr", "fpr")

# Plot the ROC curves for SVM and decision tree models
plot(svm_performance, col = "blue", lwd = 2, main = "ROC Curve Comparison")
plot(dt_performance, col = "red", lwd = 2, add = TRUE)

# Add a legend to the plot
legend("bottomright", legend = c("SVM", "Decision Tree"), col = c("blue", "red"), lwd = 2)
```

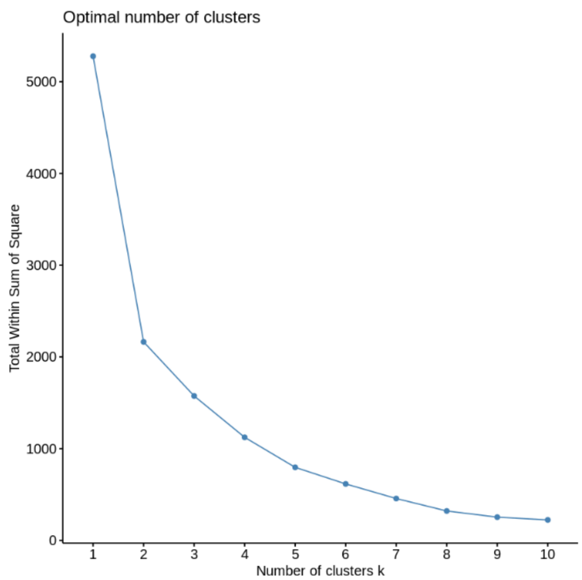
Part 2: Unsupervised Learning

A. K-Means Clustering:

(1) Perform k-means clustering on the selected attributes, specifying k = 4 clusters and plot.



(2) Apply the elbow method to determine the best k and plot.

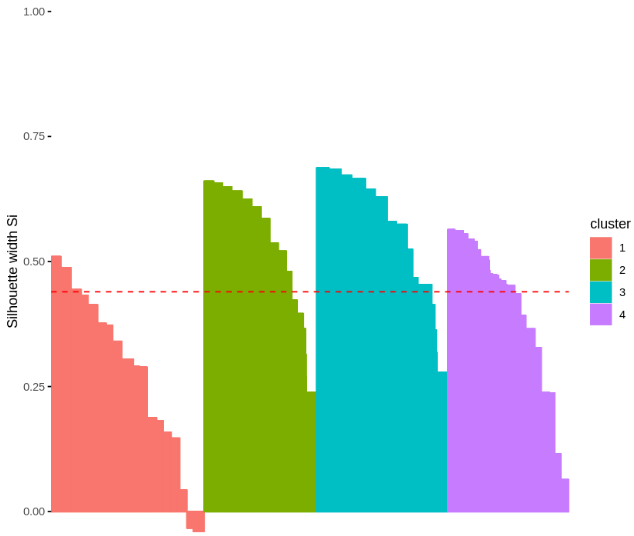


(3) Evaluate the quality of the clusters using the Silhouette Coefficient method.

```
  cluster size ave.sil.width
1       1 1253          0.28
2       2  921          0.54
3       3 1075          0.57
4       4  991          0.40
```

Clusters silhouette plot
Average silhouette width: 0.44

Clusters silhouette plot
Average silhouette width: 0.44



B. Hierarchical Clustering:

1) Use hierarchical agglomerative clustering. Draw a dendrogramز

	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	88	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	88	83	47	8	0	27
195	185	175	155	115	110	74	35	27	0

The minimum value is 5 in value 80,85

	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

The minimum value is 8 in value 160,168

	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

The minimum value is 10 in value 10,20

	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	63	0	39	74
160-168	140	120	75	39	0	27
195	175	155	110	74	27	0

The minimum value is 20 in value (10-20),40

	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

<u>The minimum value is 27in value (160-168),195</u>

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

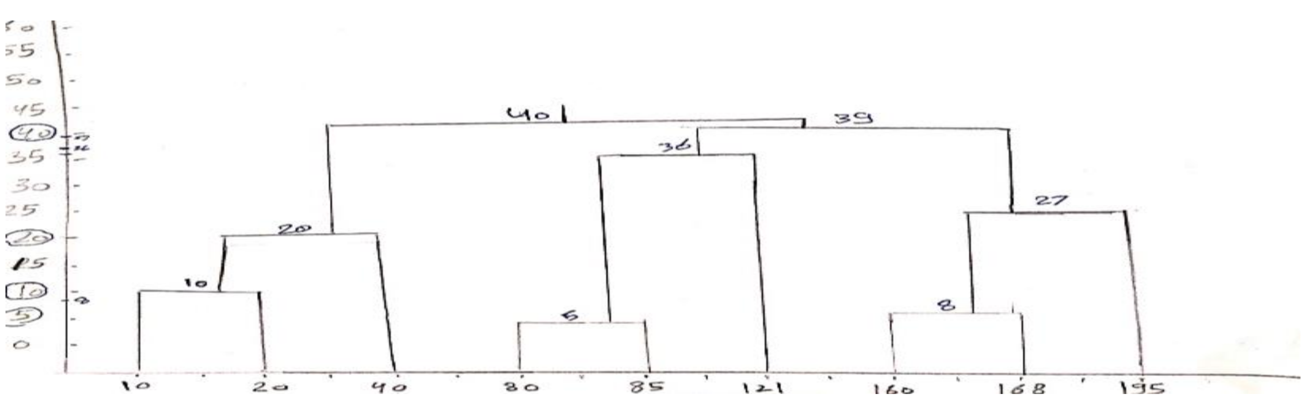
<u>The minimum value is 36 in value (80-85),121</u>

	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

<u>The minimum value is 39 in value (80-85-121), (160-168-195)</u>

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

<u>The minimum value is 40 in the clusters (80-85-121-160-168-195), (10-20-40)</u>



(2) Repeat part (a) using hierarchical agglomerative clustering with complete linkage

	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	88	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	88	83	47	8	0	27
195	185	175	155	115	110	74	35	27	0

<u>The minimum value is 5 in value 80,85</u>

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

<u>The minimum value is 8 in value 160,168</u>

	10	20	40	85	121	160-168	195
10	0	10	30	75	111	158	185
20	10	0	20	65	101	148	175
40	30	20	0	45	81	128	155
85	75	65	45	0	41	88	115
121	111	101	81	41	0	47	74
160-168	158	148	128	88	47	0	35

195	185	175	155	115	74	35	0

<u>The minimum value is 10 in value 10,20</u>

	10-20	40	85	121	168	195
10-20	0	30	75	111	158	185
40	30	0	45	81	128	155
85	75	45	0	41	88	115
121	111	81	41	0	47	74
168	158	128	88	47	0	35
195	185	155	115	74	35	0

<u>The minimum value is 30 in value (10,20),40</u>

	10-20-40	85	121	168	195
10-20-40	0	75	111	158	185
85	75	0	41	88	115
121	111	41	0	47	74
168	158	88	47	0	35
195	185	115	74	35	0

<u>The minimum value is 35 in value 168,195</u>

	10-20-40	85	121	168-195
10-20-40	0	75	111	185
85	75	0	41	115
121	111	41	0	74
168-195	185	115	74	0

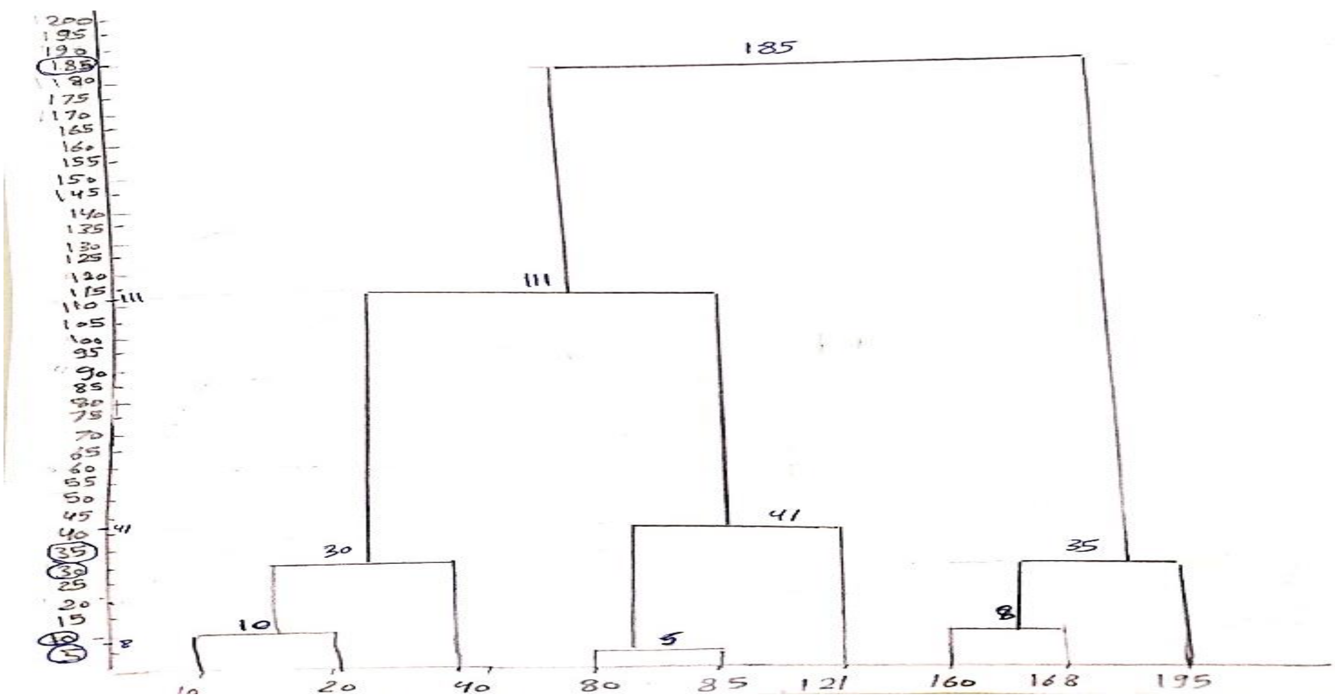
<u>The minimum value is 41in value 85,121</u>

	10-20-40	85-121	168-195
10-20-40	0	111	185
85-121	111	0	115
168-195	185	115	0

<u>The minimum value is 111in value (85,121), (10-20-40)</u>

	10-20-40-85-121	168-195
10-20-40-85-121	0	185
168-195	185	0

<u>The minimum value is 185</u>



Group 3:

- **Amira Abu Isaa**
- **Aya Metwallly**