EAS 509

Statistical Learning II

CLUSTERING

PROJECT - I

Prepared by:

Shravani Soma

Prathamesh Sunil Kakade

Sriinitha Chinnapatlola

PROBLEM STATEMENT:

As pulmonary diseases are very common and can affect people belonging to all age groups, it is essential to identify the underlying characteristics of the patients having pulmonary disease. This is done by implementing various clustering techniques that help us to gain meaningful insights from the clusters or groups formed.

Nature of the dataset (Exasens dataset): The dataset has been taken from UCI Machine Learning Repository. The dataset consists of multivariable data and features consists of both numeric and categorical data and consists of 9 attributes and 399 rows**.** The "**Exasens"** dataset has demographic information of 4 groups of saliva samples diagnosed as follows:

(1) COPD: Patients (Outpatients and hospitalized) with COPD without acute respiratory infections.

(2) Asthma: Patients (Outpatients and hospitalized) with asthma without acute respiratory infections.

(3) Infected: Patients having respiratory infections but have no COPD or asthma.

(4) HC: Healthy controls without COPD, asthma, or any respiratory infection.

The attributes of the dataset are:

Attribute Name	**Type**
Diagnosis	character
ID	character
Gender	Integer (1=male, 0=female)
Smoking Status	Integer (1=Non-smoker, 2=Ex-smoker, 3=Active smoker)
Saliva Permittivity a) Imaginary part	Numeric (Min(Î")=Absolute minimum value, Avg.(Î")=Average)
Saliva Permittivity b)Real part	Integer (Min(Î")=Absolute minimum value, Avg.(Î")=Average)
Age	integer

DATA PREPROCESSING:

1) Replacing NA in place of missing values:

```r
# Set working directory to the folder containing the CSV file
setwd("C:\\Users\\Sriinitha Reddy\\Downloads")

# Read CSV file into a data frame
data <- read.csv("Exasens.csv", header=TRUE,na.strings = c("", "NA", "N/A"))
```

Fig: Reading the CSV file and specifying the missing values

We initially start by setting the current working directory to the directory where we have our dataset – Exasens. Using the read.csv we read our CSV file and the "**na.strings**" argument is used to specify the characters that are to be treated as missing values.

2) Replacing NA values with the mean:

```r
# View the first 10 rows of a data frame called my_data
head(data, n = 10)
```

	Diagnosis <chr>	ID <chr>	Imaginary.Part <chr>	X <chr>	Real.Part <chr>	X.1 <chr>	Gender <int>	Age <int>	Smoking <int>
1	COPD	301-4	-320.61	-300.5635307	-495.26	-464.1719907	1	77	2
2	COPD	302-3	-325.39	-314.7503595	-473.73	-469.2631404	0	72	2
3	COPD	303-3	-323	-317.4360556	-476.12	-471.8976667	1	73	3
4	COPD	304-4	-327.78	-317.3996698	-473.73	-468.856388	1	76	2
5	COPD	305-4	-325.39	-316.1557853	-478.52	-472.8697828	0	65	2
6	COPD	306-3	-327.78	-318.6775535	-507.23	-469.0241943	1	60	2
7	COPD	307-3	-330.18	-320.6174777	-473.73	-467.3618538	1	76	2
8	COPD	308	*NA*	*NA*	*NA*	*NA*	1	77	2
9	COPD	309-4	-320.61	-307.5995856	-476.12	-470.1816328	1	74	2
10	COPD	310-4	-315.82	-300.104765	-473.73	-466.3786343	1	67	2

1-10 of 10 rows

Fig: First 10 rows of the dataset before replacing the missing values.

```r
for (column in colnames(data)) {
  num_missing <- sum(is.na(data[[column]]))
  print(paste0("Number of missing values in ", column, ": ", num_missing))
}
```

```
[1] "Number of missing values in Diagnosis: 0"
[1] "Number of missing values in ID: 0"
[1] "Number of missing values in Imaginary.Part: 299"
[1] "Number of missing values in X: 299"
[1] "Number of missing values in Real.Part: 299"
[1] "Number of missing values in X.1: 299"
[1] "Number of missing values in Gender: 0"
[1] "Number of missing values in Age: 0"
[1] "Number of missing values in Smoking: 0"
```

Fig: Number of missing values present in each column of the dataset.

```r
# Convert the Imaginary.Part column to numeric class
data$Imaginary.Part <- as.numeric(data$Imaginary.Part)
data$Real.Part <- as.numeric(data$Real.Part)
data$X <- as.numeric(data$X)
data$X.1 <- as.numeric(data$X.1)
# Calculate the mean of the column and replace missing values with the mean

mean_img <- mean(data$Imaginary.Part, na.rm = TRUE)
data$Imaginary.Part[is.na(data$Imaginary.Part)] <- mean_img
mean_real <- mean(data$Imaginary.Part, na.rm = TRUE)
data$Real.Part[is.na(data$Real.Part)] <- mean_real
mean_X1 <- mean(data$X.1, na.rm = TRUE)
data$X.1[is.na(data$X.1)] <- mean_X1
mean_X <- mean(data$X, na.rm = TRUE)
data$X[is.na(data$X)] <- mean_X
```

```r
head(data, n = 10)
```

	Diagnosis <chr>	ID <chr>	Imaginary.Part <dbl>	X <dbl>	Real.Part <dbl>	X.1 <dbl>	Gender <int>	Age <int>	Smoking <int>
1	COPD	301-4	-320.6100	-300.5635	-495.2600	-464.1720	1	77	2
2	COPD	302-3	-325.3900	-314.7504	-473.7300	-469.2631	0	72	2
3	COPD	303-3	-323.0000	-317.4361	-476.1200	-471.8977	1	73	3
4	COPD	304-4	-327.7800	-317.3997	-473.7300	-468.8564	1	76	2
5	COPD	305-4	-325.3900	-316.1558	-478.5200	-472.8698	0	65	2
6	COPD	306-3	-327.7800	-318.6776	-507.2300	-469.0242	1	60	2
7	COPD	307-3	-330.1800	-320.6175	-473.7300	-467.3619	1	76	2
8	COPD	308	-314.9418	-304.7797	-314.9418	-458.7017	1	77	2
9	COPD	309-4	-320.6100	-307.5996	-476.1200	-470.1816	1	74	2
10	COPD	310-4	-315.8200	-300.1048	-473.7300	-466.3786	1	67	2

1-10 of 10 rows

Fig: First 10 rows of the dataset after replacing the missing values.

Here as the dataset consists of 399 rows, there were missing values present in 299 rows. Hence, removing the rows that consist of missing values is not optimal. Instead, the missing values can be replaced either with the mean or median of each of the columns.

3) Converting Real part and Imaginary part column values to float:

```r
# Convert the values in Imaginary part, Real part to float type
data$Imaginary.Part <- as.numeric(data$Imaginary.Part)
data$Real.Part <- as.numeric(data$Real.Part)
data$X <- as.numeric(data$X)
data$X.1 <- as.numeric(data$X.1)

head(data, n = 10)
```

	Diagnosis <chr>	ID <chr>	Imaginary.Part <dbl>	X <dbl>	Real.Part <dbl>	X.1 <dbl>	Gender <int>	Age <int>	Smoking <int>
1	COPD	301-4	-320.6100	-300.5635	-495.2600	-464.1720	1	77	2
2	COPD	302-3	-325.3900	-314.7504	-473.7300	-469.2631	0	72	2
3	COPD	303-3	-323.0000	-317.4361	-476.1200	-471.8977	1	73	3
4	COPD	304-4	-327.7800	-317.3997	-473.7300	-468.8564	1	76	2
5	COPD	305-4	-325.3900	-316.1558	-478.5200	-472.8698	0	65	2
6	COPD	306-3	-327.7800	-318.6776	-507.2300	-469.0242	1	60	2
7	COPD	307-3	-330.1800	-320.6175	-473.7300	-467.3619	1	76	2
8	COPD	308	-314.9418	-304.7797	-314.9418	-458.7017	1	77	2
9	COPD	309-4	-320.6100	-307.5996	-476.1200	-470.1816	1	74	2
10	COPD	310-4	-315.8200	-300.1048	-473.7300	-466.3786	1	67	2

1-10 of 10 rows

Fig: Dataset after converting data in real, imaginary part, X.1, X columns to type float .

We initially convert the data in the columns Imaginary part, Real part, X.1 and X to float to later convert those to absolute values.

4) Converting the float values to absolute values:

```r
# Take the absolute value of the Real Part and Imaginary Part columns
data <- data %>% mutate(
  `Real.Part` = abs(`Real.Part`),
  `Imaginary.Part` = abs(`Imaginary.Part`),
  `X` = abs(`X`),
  `X.1` = abs(`X.1`)

)
head(data, n = 10)
```

	Diagnosis <chr>	ID <chr>	Imaginary.Part <dbl>	X <dbl>	Real.Part <dbl>	X.1 <dbl>	Gender <int>	Age <int>	Smoking <int>
1	COPD	301-4	320.6100	300.5635	495.2600	464.1720	1	77	2
2	COPD	302-3	325.3900	314.7504	473.7300	469.2631	0	72	2
3	COPD	303-3	323.0000	317.4361	476.1200	471.8977	1	73	3
4	COPD	304-4	327.7800	317.3997	473.7300	468.8564	1	76	2
5	COPD	305-4	325.3900	316.1558	478.5200	472.8698	0	65	2
6	COPD	306-3	327.7800	318.6776	507.2300	469.0242	1	60	2
7	COPD	307-3	330.1800	320.6175	473.7300	467.3619	1	76	2
8	COPD	308	314.9418	304.7797	314.9418	458.7017	1	77	2
9	COPD	309-4	320.6100	307.5996	476.1200	470.1816	1	74	2
10	COPD	310-4	315.8200	300.1048	473.7300	466.3786	1	67	2

1-10 of 10 rows

Fig: Dataset after converting the float values to absolute values.

To make the mathematical computations easier and to extract the essential features, at this step we preprocess the data to convert the float values present in columns Imaginary Part, Real Part, X.1 and X to absolute values.

5) Removing ID column:

```r
```{r}
cleaned_data <- select(data, -ID)
head(data, n = 10)
```
```

	Diagnosis <chr>	ID <chr>	Imaginary.Part <dbl>	X <dbl>	Real.Part <dbl>	X.1 <dbl>	Gender <int>	Age <int>	Smoking <int>
1	COPD	301-4	320.6100	300.5635	495.2600	464.1720	1	77	2
2	COPD	302-3	325.3900	314.7504	473.7300	469.2631	0	72	2
3	COPD	303-3	323.0000	317.4361	476.1200	471.8977	1	73	3
4	COPD	304-4	327.7800	317.3997	473.7300	468.8564	1	76	2
5	COPD	305-4	325.3900	316.1558	478.5200	472.8698	0	65	2
6	COPD	306-3	327.7800	318.6776	507.2300	469.0242	1	60	2
7	COPD	307-3	330.1800	320.6175	473.7300	467.3619	1	76	2
8	COPD	308	314.9418	304.7797	314.9418	458.7017	1	77	2
9	COPD	309-4	320.6100	307.5996	476.1200	470.1816	1	74	2
10	COPD	310-4	315.8200	300.1048	473.7300	466.3786	1	67	2

1-10 of 10 rows

Fig: Columns in the dataset after removing the ID column.

In general, one of the ways to do feature extraction is by plotting the heatmap and depending on the correlation between each of the features, we select them. But out of the 9 columns, we can say that the "**ID**" column consists of patient ID and it in no way contributes to a patient being diagnosed any of the disease. Hence, we remove the "ID" column.

6) Encoding Categorical data:

```r
```{r}
heatmap_df = cleaned_data
heatmap_df$Diagnosis <- as.numeric(factor(heatmap_df$Diagnosis))
head(heatmap_df, n = 10)
```
```

	Diagnosis <dbl>	Imaginary.Part <dbl>	X <dbl>	Real.Part <dbl>	X.1 <dbl>	Gender <int>	Age <int>	Smoking <int>
1	2	320.6100	300.5635	495.2600	464.1720	1	77	2
2	2	325.3900	314.7504	473.7300	469.2631	0	72	2
3	2	323.0000	317.4361	476.1200	471.8977	1	73	3
4	2	327.7800	317.3997	473.7300	468.8564	1	76	2
5	2	325.3900	316.1558	478.5200	472.8698	0	65	2
6	2	327.7800	318.6776	507.2300	469.0242	1	60	2
7	2	330.1800	320.6175	473.7300	467.3619	1	76	2
8	2	314.9418	304.7797	314.9418	458.7017	1	77	2
9	2	320.6100	307.5996	476.1200	470.1816	1	74	2
10	2	315.8200	300.1048	473.7300	466.3786	1	67	2

1-10 of 10 rows

Fig: Dataset after performing encoding on the categorical columns.

Heatmap is plotted for a dataset that consists of numeric data. But in our dataset, we have a column **"Diagnosis"** which consists of **categorical data**. Hence, before we plot the heatmap we perform encoding on the "Diagnosis" column as it is categorical and not continuous in nature.

7) Scaling:

```r
library(stats)

features <- heatmap_df[,c("Diagnosis", "Imaginary.Part", "Real.Part", "Gender", "Age", "Smoking")]

# Standardize(scaling) the features
features <- scale(features)
features <- as.data.frame(features)
head(features, n = 10)
```

	Diagnosis <dbl>	Imaginary.Part <dbl>	Real.Part <dbl>	Gender <dbl>	Age <dbl>	Smoking <dbl>
1	-0.5886019	0.39985540	1.9324622	1.2270497	1.5222284	0.3680883
2	-0.5886019	0.73705395	1.6367642	-0.8129204	1.2529329	0.3680883
3	-0.5886019	0.56845468	1.6695890	1.2270497	1.3067920	1.7154941
4	-0.5886019	0.90565323	1.6367642	1.2270497	1.4683693	0.3680883
5	-0.5886019	0.73705395	1.7025511	-0.8129204	0.8759192	0.3680883
6	-0.5886019	0.90565323	2.0968609	1.2270497	0.6066236	0.3680883
7	-0.5886019	1.07495794	1.6367642	1.2270497	1.4683693	0.3680883
8	-0.5886019	0.00000000	-0.5440697	1.2270497	1.5222284	0.3680883
9	-0.5886019	0.39985540	1.6695890	1.2270497	1.3606511	0.3680883
10	-0.5886019	0.06195142	1.6367642	1.2270497	0.9836374	0.3680883

1-10 of 10 rows

Fig: Dataset after Scaling.

After identifying the relevant features, we scale the data. Scaling the data is important as to avoid the domination of one feature over the other. As different columns have values in different ranges, it is essential to scale them so that they are equal significance.

DATA VISUALIZATION:

(1) Diagnosis Frequency:

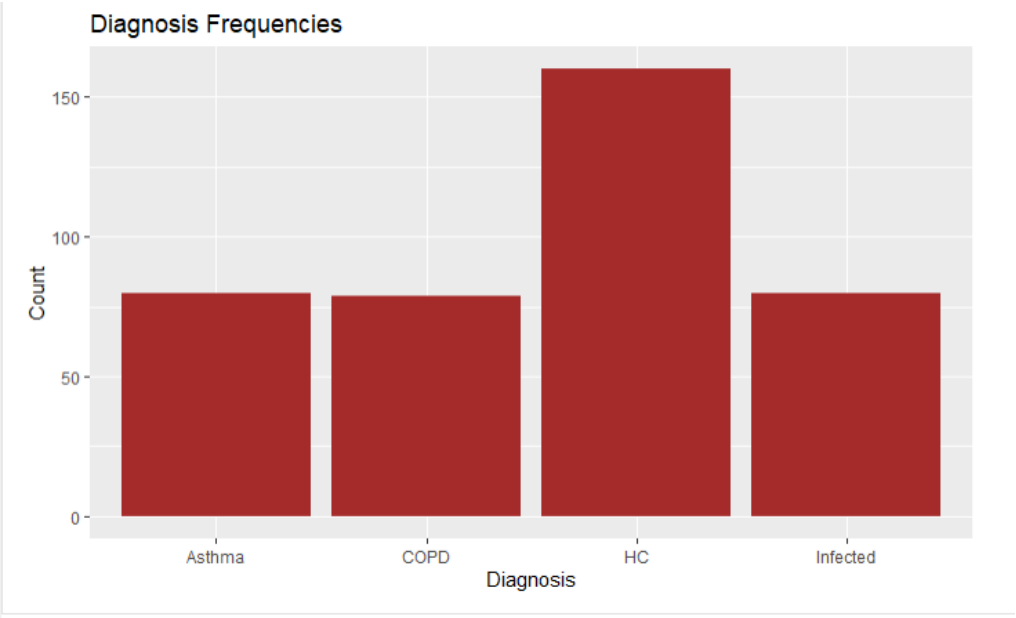

Fig: Bar plot representing the count of each type of Diagnosis

Observations:

From the above bar plot, we can observe that the number of patients diagnosed with HC are the highest among all the 4. The number of patients not infected with any diseases or suffering from asthma or COPD are almost equal in number with a very little variation.

(2) Age Distribution of participants:

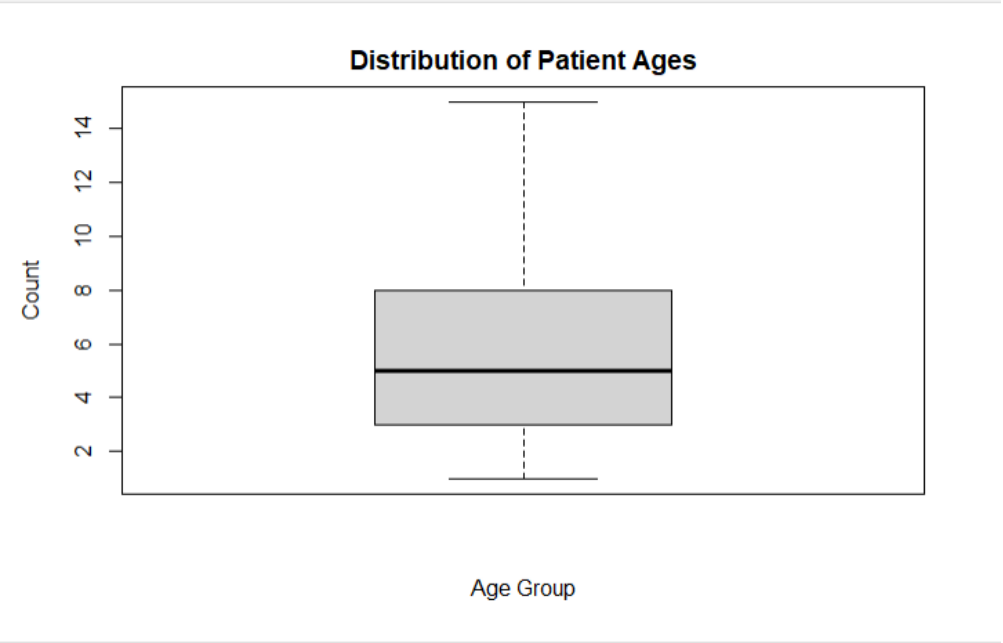

Fig: Box plot representing the age distribution of the patients.

Observation:

Box plots help us in determining the presence of outliers in our data. From the above box plot of age, we can infer that there are no outliers and our data is not skewed in terms of age.

(3) Heatmap:

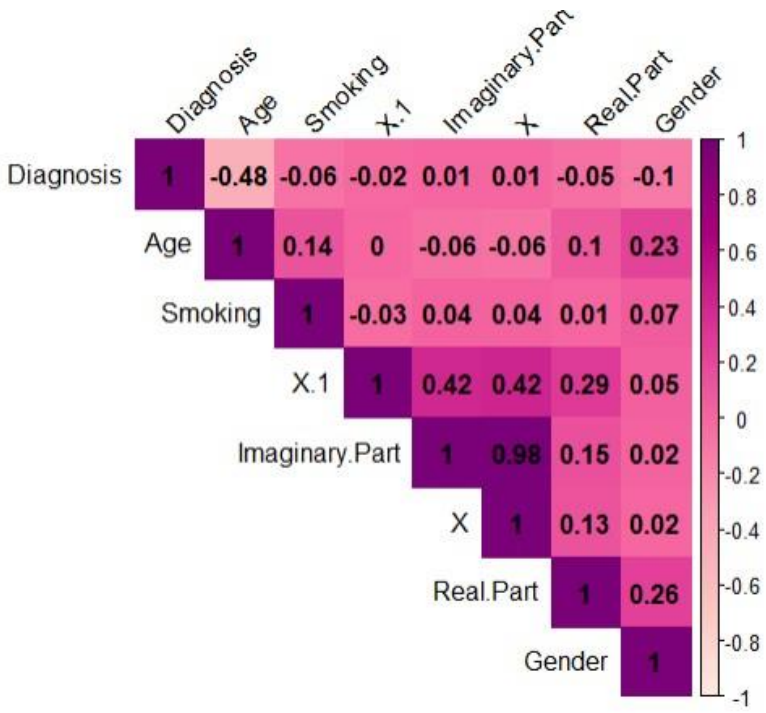

Fig: Heatmap of all the features in the dataset.

Heatmap visually represents the correlation between various features in the dataset. From the heatmap we can conclude that, the features “X.1” and “X” are **highly correlated** to the columns “Imaginary part” and “Real Part”. Hence, we drop the X.1, and X columns from our features and perform clustering on the rest of the features.

CLUSTERING:

Clustering is an unsupervised machine learning that is not aware of the classes of the data but tries to find patterns within the datapoints and group them based on their similarities. Here, we are implementing two types of clustering algorithms:

(1) K-Means Clustering:

This algorithm works by initializing K-centroids and assigning each datapoint to the closest centroid based on the Euclidean distance. This process is done until there is no change in the value of the centroids.

Elbow Method: As we pass the parameter K in K-means clustering algorithm, we use the elbow method to determine the optimal K value. In this method, we calculate the clustering score over a given range of values of k and identify the K where the curve bends and that is chosen value of K.

```
253 ```{r}
254 library(cluster)
255 library(ggplot2)
256
257 # Compute the within-cluster sum of squares for different values of k
258 wss <- (nrow(heatmap_df)-1)*sum(apply(data,2,var))
259 for (i in 2:10) wss[i] <- sum(kmeans(heatmap_df, centers=i)$withinss)
260
261 # Plot the elbow curve
262 ggplot(data.frame(x=1:8, y=wss[1:8]), aes(x=x, y=y)) + geom_line() + geom_point() + scale_x_continuous(breaks=1:8)
    +labs(x="Number of Clusters", y="WSS") +ggtitle("Elbow Curve")
263 ```
```

Fig: Code to plot Elbow Curve to find optimal K values for K-means clustering

ELBOW CURVE

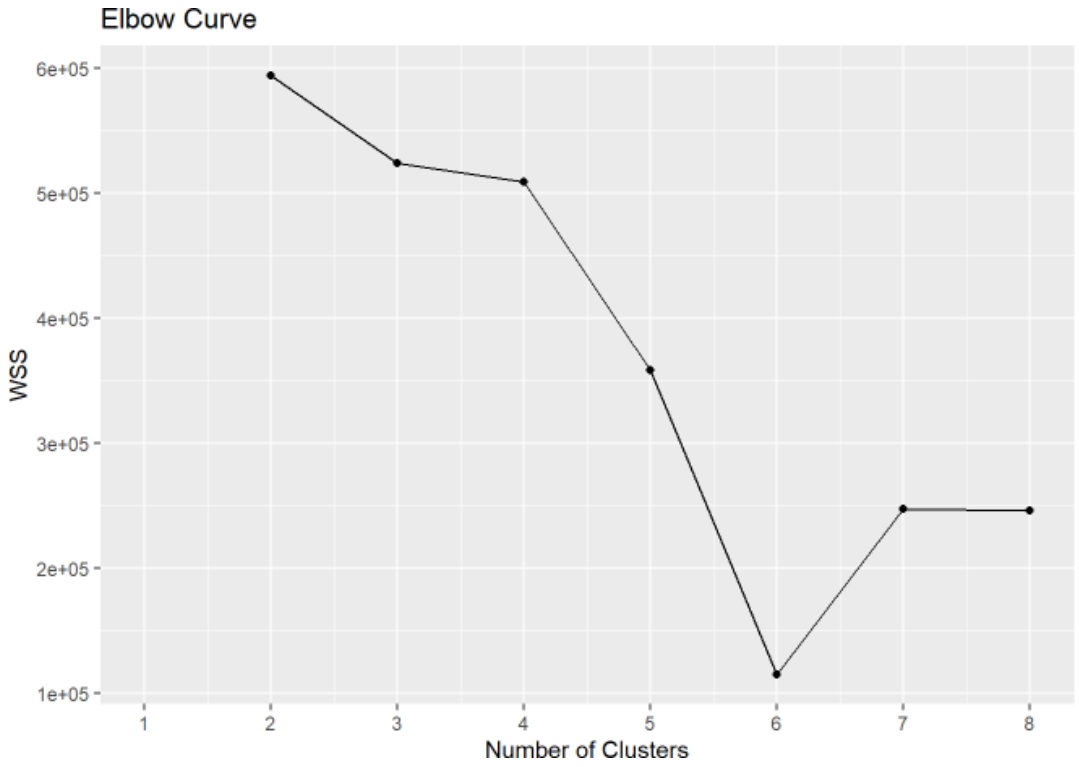

Fig: Elbow Curve (K – means Clustering)

From the above elbow curve, we can infer that the curve bends at k=3. We have plotted the curve for 8 clusters as the dataset is not very large. On analyzing we can conclude that the optimal K-value for our dataset is 3.

Fitting the model:

```r
# Perform k-means clustering
k_best = 3
kmeans_model <- kmeans(features, k_best)
```

Fig: code to fit the data to the model

Results/Summary:

```r
# View the clustering results
print(kmeans_model)
```

```
K-means clustering with 3 clusters of sizes 10, 160, 229

Cluster means:
    Diagnosis Imaginary.Part  Real.Part     Gender        Age    Smoking
1 -0.09932658     -5.8580700  0.6790762  0.4110616  0.4989054 -0.1708740
2 -0.95555845      0.2386319  0.2234628  0.2835635  0.7991699  0.3007180
3  0.67197650      0.0890812 -0.1857852 -0.2160733 -0.5801583 -0.2026469

Clustering vector:
  [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 2 2 2 2 2 2 2 2 1 1 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [54] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
[107] 3 2 3 3 3 3 3 3 2 2 3 3 1 3 3 3 1 3 3 3 3 3 3 3 3 1 1 3 1 3 3 3 2 3 3 3 2 2 2 2 3 3 2 3 2 3 2 3 3 3 3 3 3
[160] 3 3 3 3 3 3 3 3 2 3 3 3 3 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
[213] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 3 2 3 2 2 2 2 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2
[266] 2 2 2 2 2 2 2 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 2 2 2 2 2 2 2
[319] 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
[372] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3

Within cluster sum of squares by cluster:
[1]  58.12748 638.43469 836.06944
 (between_SS / total_SS =  35.8 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"    "size"
[8] "iter"         "ifault"
```

Fig: K-means clustering results

Result Visualization:

```r
# Plotting the clusters
library(ggfortify)
autoplot(kmeans_model,features,frame=TRUE)+
  scale_fill_manual(values = c("red", "green", "blue")) +
  labs(fill = "Cluster")
```

Fig: Code to plot the scatter plot to visualize clusters

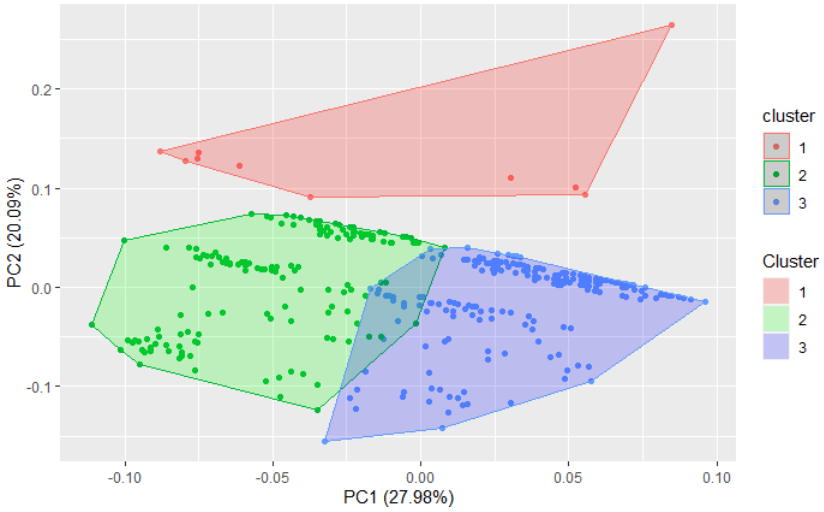

Fig: Scatterplot representing 3 clusters

Hyperparameter Tuning (K):

Here, we can tune the K-value to find how the datapoints are clustered based on different values of K. Here are scatterplots obtained by varying the hyper parameter k.

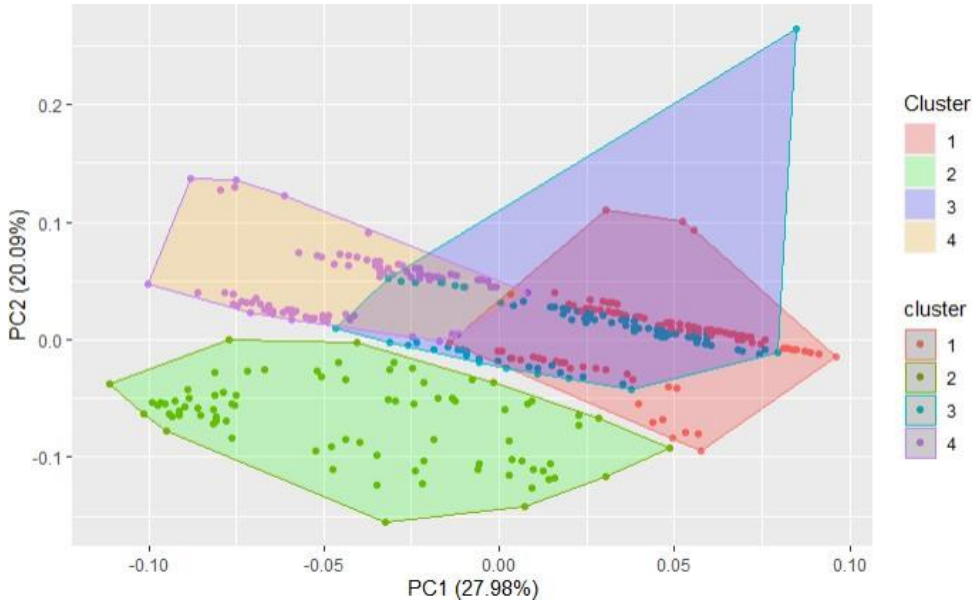

Fig: Scatterplot representing 4 clusters

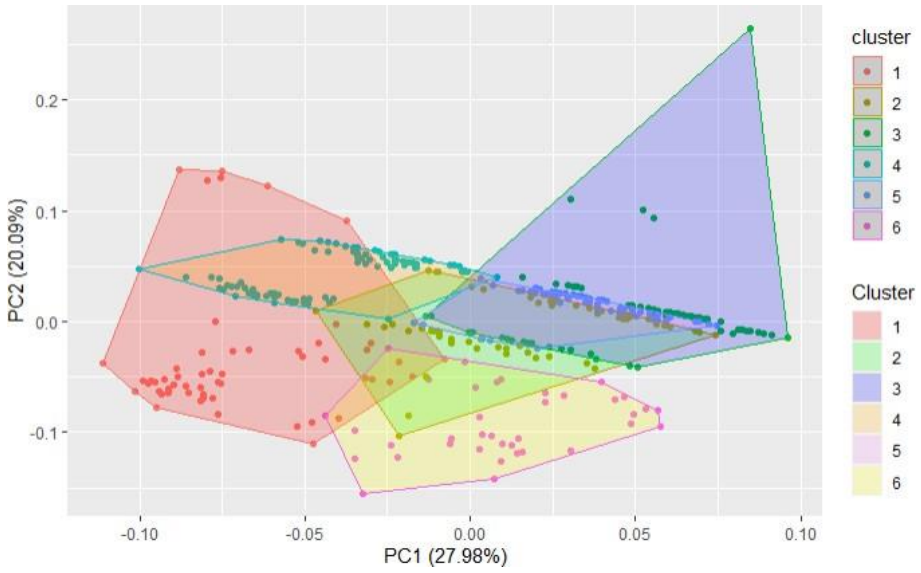

Fig: Scatterplot representing 6 clusters

From the above two scatter plots we can conclude that as the dataset is not very huge, increasing the number of clustering (hyper tuning the parameter – k) results in no proper formation of clusters.

Evaluation Metric:

Silhouette Score: Silhouette score is one of the evaluation metrics to evaluate the results of a clustering algorithm. The silhouette score ranges from -1 to 1 where -1 indicates that the data point is more like the neighboring clusters than the assigned cluster and 1 indicates that the datapoint is like the assigned cluster.

```r
knn_score <- silhouette(kmeans_model$cluster, dist(heatmap_df))

# Calculate the average Silhouette score
avg_score <- mean(knn_score[, 3])

# Print the average Silhouette score
print(paste("Average Silhouette score for the K-means clustering is:", avg_score))
```

```
[1] "Average Silhouette score for the K-means clustering is: 0.0719031886791418"
```

Fig: Average Silhouette Score for K-means clustering

Average Silhouette score (K – Means): The Silhouette score obtained for K-means clustering is 0.07 which is close to 0 and that means that data points are not clearly assigned to the clusters. This indicates that the result is not well defined, or the data point may belong to another cluster.

(2) Hierarchical Clustering:

Hierarchical clustering works by iteratively combining pairs of data points or existing clusters based on their similarity, creating a hierarchical tree-like structures called dendrograms.

```r
distance <- dist(heatmap_df, method = "euclidean")
complete_h <- hclust(distance, method = "complete")
single_h <- hclust(distance,method = "single")
avg_h <- hclust(distance,method = "average")
centroid_h <- hclust(distance,method="centroid")
```

Fig: Code that computes the distance matrix and hclust() to create dendrogram for each of the linkage method

Linkage methods:

There are 4 linkage methods we can implement in Hierarchical Clustering, and they are implemented as follows:

```r
# Dendrogram - method - Complete
cutree(complete_h,4)
plot(complete_h, hang = -1)


# Dendrogram - method - Single
cutree(single_h,4)
plot(single_h, hang = -1)

# Dendrogram - method -Average
cutree(avg_h,4)
plot(avg_h, hang = -1)

# Dendrogram - method - Centroid
cutree(centroid_h,4)
plot(centroid_h, hang = -1)
```

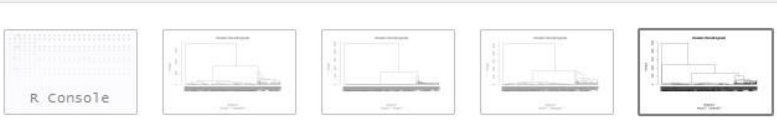

Fig: Code to plot dendrograms for each of the linkage methods

There are 4 linkage methods we can implement in Hierarchical Clustering, and they are implemented as follows:

(1) Complete method: This method calculates maximum distance between all the pair of points in any of the two clusters.

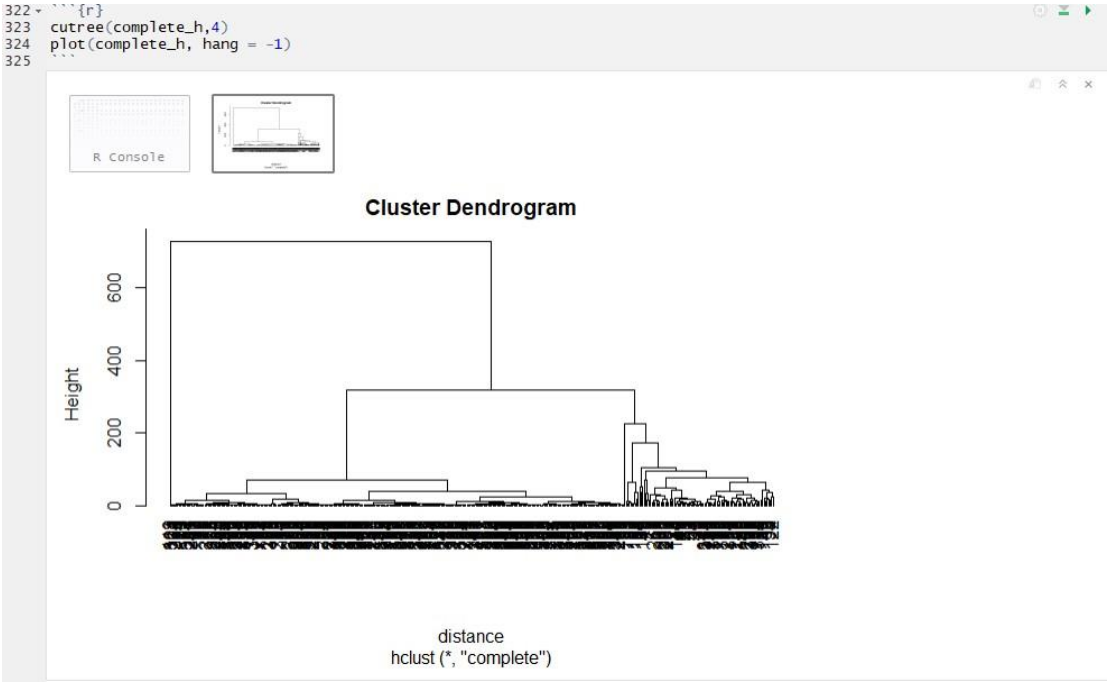

Fig: Dendrogram of hierarchical clustering using complete method

(2) Single method: This method calculates minimum distance between all the pair of points in any of the two clusters.

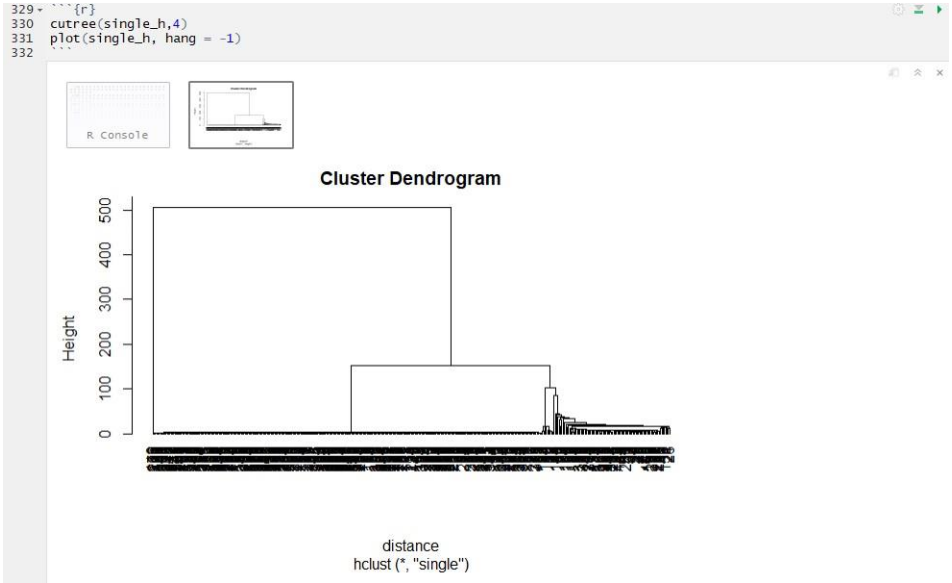

Fig: Dendrogram of hierarchical clustering using single method

(3) Average Method: This method calculates the average distance between all the pair of points in any of the two clusters.

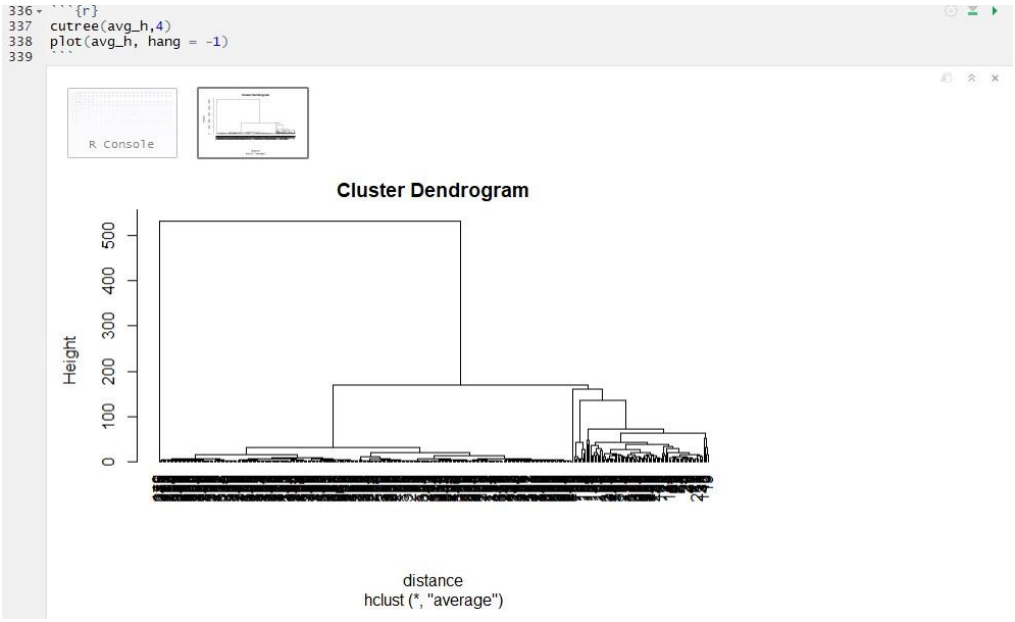

Fig: Dendrogram of hierarchical clustering using Average method

(4) Centroid Method: This method calculates the distance between the centroids in any of the two clusters.

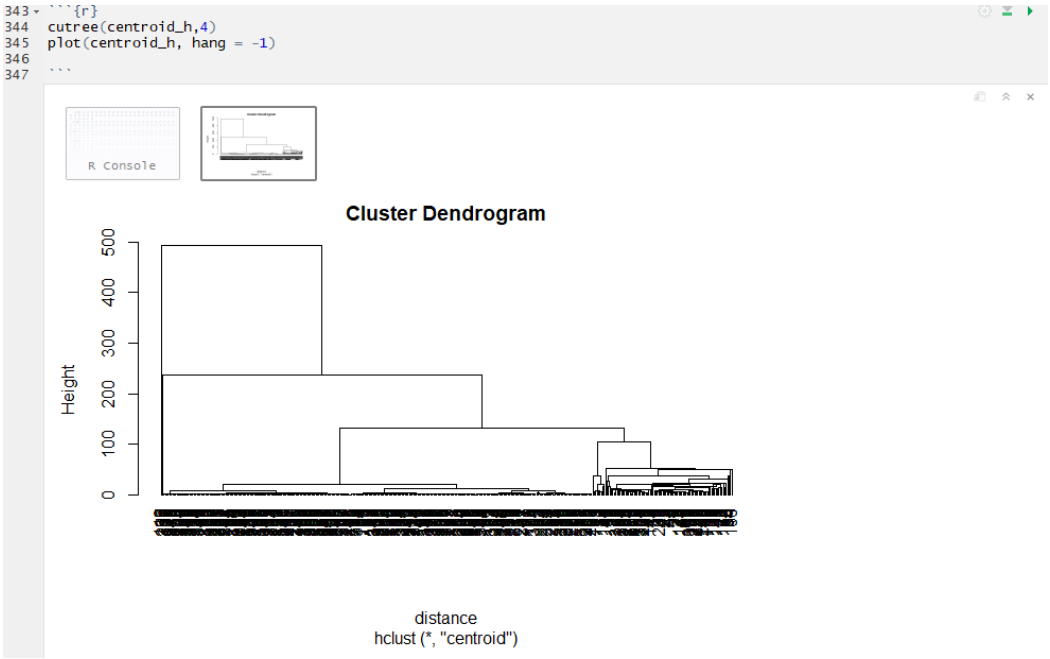

Fig: Dendrogram of hierarchical clustering using centroid method

Evaluation metrics:

Average Silhouette score (Complete Method):

```r
# The number of clusters we choose here are 3 for hierarchical clustering
complete_score <- silhouette(cutree(complete_h, k = 3), dist(heatmap_df))
avg_score_complete <- mean(complete_score[, 3])
print(paste("Average Silhouette score using complete method is:",avg_score_complete))
```

```
[1] "Average Silhouette score using complete method is: 0.826030750636403"
```

Fig: Average Silhouette Score using complete method

Silhouette Score: The average silhouette score obtained using the **complete method** is 0.82 which is close to one. From this we can conclude that most of the datapoints are similar to the cluster they are clustered into which means that the clusters are formed properly.

Average Silhouette score (Single Method):

```r
single_score <- silhouette(cutree(single_h, k = 3), dist(heatmap_df))
avg_score_single <- mean(single_score[, 3])
print(paste("Average Silhouette score using single method is:",avg_score_single))
```

```
[1] "Average Silhouette score using single method is: 0.826030750636403"
```

Fig: Average Silhouette Score using Single method

Silhouette Score: The average silhouette score obtained using the **single method** is 0.83 which is close to one. From this we can conclude that most of the datapoints are similar to the cluster they are clustered into which means that the clusters are formed properly.

Average Silhouette score (Average Method):

```r
average_score <- silhouette(cutree(avg_h, k = 3), dist(heatmap_df))
avg_score_method <- mean(average_score[, 3])
print(paste("Average Silhouette score using average method is:",avg_score_method))
```

```
[1] "Average Silhouette score using average method is: 0.826030750636403"
```

Fig: Average Silhouette Score using Average method

Silhouette Score: The average silhouette score obtained using the **average method** is 0.83 which is close to one. From this we can conclude that most of the datapoints are similar to the cluster they are clustered into which means that the clusters are formed properly.

Average Silhouette score (Average Method):

```r
centroid_score <- silhouette(cutree(centroid_h, k = 3), dist(heatmap_df))
avg_score_centroid <- mean(centroid_score[, 3])
print(paste("Average Silhouette score using centroid method is:",avg_score_centroid))
```

```
[1] "Average Silhouette score using centroid method is: 0.642083063167706"
```

Fig: Average Silhouette Score using Centroid method

Silhouette Score: The average silhouette score obtained using the **Centroid method** is 0.64 which is close to one. From this we can conclude that most of the datapoints are similar to the cluster they are clustered into which means that the clusters are formed properly.

CONCLUSION:

After performing various clustering algorithms on the Exasens dataset, we conclude that the hierarchical clustering using the complete, single and average methods gave best performance as the evaluation metric (i.e average silhouette score) obtained in all the three linkage methods is 0.83.

REFERENCES:

1) R: Documentation (r-project.org)

2) Function reference • ggplot2 (tidyverse.org)

3) dplyr package - RDocumentation

4) corrplot function - RDocumentation

5) cluster package - RDocumentation