DETERMINATION OF PROPER GROUPING APPROACHES FOR LARGE DATA SETS BY APPLYING CLUSTER ANALYSIS METHODS IN MATLAB

by
Gomarage Umesha Madushan
SC/2019/10724

Supervisor: Prof. Leslie Jayasekara

Project report submitted to the
Department of Mathematics
in partial fulfillment of the requirement for the
Level II Industrial Mathematics course unit (IMT2b2β)
of the

Bachelor of Science (General) Degree

in the
Faculty of Science
University of Ruhuna
Matara
Sri Lanka.

December 2022

ACKNOWLEDGMENTS

I am very grateful to my supervisor **Prof. Leslie Jayasekara** for his advice and guidance. Your advice and helping us get through mathematically difficult spots is greatly appreciated. I am also very thankful my group Members **Vasala Bandara** and **Gimhani Dewasingha** for the support and dedication given to make this project a success. Without all of us, this project would not have been possible. My heartfelt thanks to everyone who helped.

ABSTRACT

Classification or grouping is very important in a statistical study and helps to make the study easier. one of those grouping method is Clustering. Basically, a cluster is a group of individuals or objects that share a characteristic. its easy to analyse data after clustering. But clustering large amounts of data using human labor is still a very less effective process. We see that as a problem with this process. Therefore, it will be more effective if the data can be entered into a computer in an easy way and clustered by the computer to get the required output. Therefore, it is very important to design a computer program that can perform the clustering. particularly in this project, identification of problems arising in the implementation of the clustering method, Finding solutions to the identified problems and finding ways to apply them, Prepare a program using Matlab to solve the problems using the solutions found in this way were the Objectives. Matlab is a advanced and effective computer program used to program mathematical concepts. The problem described above can be avoided if a program is developed for this requirement using the Matlab program. In this project Euclidean Distances were used to Cluster Data. Hierarchical clustering method under cluster analysis was used here.In this method, the data is arranged according to a certain level of importance. also we were able to add 6 methods of clustering into this Program. we have been able to do this project successfully to some extent.

Keywords: Clustering , Matlab , Hierarchy , Euclidean Distance

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

1.1 Background of study

1.1.1 Cluster Analysis

The term ”cluster analysis” is a catch-all term for a broad range of statistical techniques that all seek to identify clusters, or groups of objects, in a sample of objects. Contrary to discriminant analysis, a group structure need not be known a prior in cluster analysis, which is crucial.

1.1.2 A Cluster

Basically, a cluster is a group of individuals or objects that share a characteristic. Cluster is described as a collection of items with comparable traits and properties in statistics and data science. Numerous fields employ these clusters. Machine learning and big data analysis are two of them.

1.1.3 Clustering

Clustering is the term used to describe the above-mentioned process of creating clusters. There are many different clustering techniques.

- Centroid-based Clustering
- Density-based Clustering
- Distribution-based Clustering

- Hierarchical Clustering

Using these methods big data can be grouped. then its gets easy to analyze. this is called cluster analysis.

1.1.4 Centroid-based Clustering

Each cluster in centroid-based clustering is represented by a center vector. The clusters are filled with objects in a way that minimizes the squared distance between each object and the central vector.

1.1.5 Density-based Clustering

By creating clusters based on the notion that a cluster in the data space is a continuous zone of high point density, separated from other clusters by continuous regions of low density, it organizes the provided collection of data into categories.

1.1.6 Distribution-based Clustering

We will fit the data according to the likelihood that it may belong to the same distribution in this clustering model. The grouping performed could be Gaussian or normal. When we have a limited number of distributions and all future data is fitted into one of them in order to optimize the distribution of data, the Gaussian distribution is more prominent.

1.1.7 Hierarchical Clustering

Data are grouped into groups in a tree structure in a hierarchical clustering method. Every data point is first treated as a separate cluster in a hierarchical clustering process. The goal of hierarchical clustering is to create a hierarchy of nested clusters. This hierarchy is graphically represented by a figure known as a dendrogram, which is an inverted tree that explains the order in which elements are combined or groups are divided.

In this project Euclidean Distances were used to Cluster Data

1.1.8 Euclidean n-Space

Using pairs of numbers, we can pinpoint any location in a coordinate system. To express a point in geometry, we need two numbers in 2D and three numbers in 3D. Most likely, a lot of people are unable to communicate effectively outside of 3D. What can we do if we want to express a point in four, five, or more dimensions of space? For example, a point in a 4-dimensional space is represented by a quadruple of numbers (2,4,3,1), and higher dimensions work similarly. So, in a nn-dimensional space, we can express a nn-tuple of numbers.

1.1.9 Euclidean Distance

The term " distance" refers to the shortest path between two sites. This distance metric is used by the majority of machine learning algorithms, such as K-Means, to gauge how similar two observations are.

1.1.10 Applications of Cluster analysis

- There are trillions of web pages on the World Wide Web, and a search engine query can provide thousands of pages as a result. These search results can be grouped into a limited number of clusters using clustering.
- Cluster analysis can be used to comprehend the patterns of climate change on earth.
- Cluster analysis can help discover the various subcategories of a disease or condition because these differences are common.
- Businesses gather a lot of data on their clients, both present and potential. Customers can be divided into a few groups using clustering.

1.2 Objectives

- Identification of problems arising in the implementation of the clustering method.
- Finding solutions to the identified problems and finding ways to apply them.
- Prepare a program using Matlab to solve the problems using the solutions found in this way.
- Proficiency in Matlab while programming
- In the end, work to solve the identified problems as successfully as possible.

CHAPTER 2
PROBLEM STATEMENT

Statistics is a very important subject in dealing with data in almost all the world's activities. Even a very large amount of data can be analyzed using statistics. But when we take fields like machine learning and big data, the amount of data in them is not an amount of data that can be easily analyzed like this. But in this case, some convenience can be obtained in data analysis by separating the data into distinct groups.

A number of different techniques are used to group this in statistics. Clustering can be mentioned as one of them. Here, data with the same characteristics are clustered together. After clustering like this, study is easier than before. Clusters also have a variety of uses. Therefore, it can also be described as a set of processed data to some extent.

But clustering large amounts of data using human labor is still a very less effective process. We see that as a problem with this process. Therefore, it will be more effective if the data can be entered into a computer in an easy way and clustered by the computer to get the required output. Therefore, it is very important to design a computer program that can perform the clustering.

Matlab is a computer program used to program mathematical concepts. It is a very advanced and effective program. The problem described above can be avoided if a program is developed for this requirement using the Matlab program.

CHAPTER 3
METHODOLOGY

3.1 Initial Data sets and attributes

Data is divided into groupings via cluster analysis that are relevant, practical, or both. If creating meaningful groups is the goal, the clusters should reflect the inherent structure of data. However, in other situations, cluster analysis serves just as a helpful starting point for further tasks, such data memorization and analysis.

Each data set in the original set has an n-number of attributes. As previously mentioned, all data are delivered as a set. Each characteristic of the observed sample or population serves as a clustering data point. for instance,

Hartigan (1975a, p. 28) compared the crime rates per 100,000 population for various cities. The data are in Table 14.1 (taken from the 1970 U.S.Statistical Abstract).

City	Murder	Rape	Robbery	Assault	Burglary	Larceny	Auto Theft
Atlanta	16.5	24.8	106	147	1112	905	494
Boston	4.2	13.3	122	90	982	669	954
Chicago	11.6	24.7	340	242	808	609	645
Dallas	18.1	34.2	184	293	1668	901	602
Denver	6.9	41.5	173	191	1534	1368	780
Detroit	13	35.7	477	220	1566	1183	788

Table 3.1: Example (1.1) (Euclidean Space Data)

Due to above example each row is a data set and all or some values in this set will be considered as a attribute while clustering.due to this example, Attributes of

Atlanta are **{ 16.5, 24.8, 106, 147, 1112, 905, 494 }**each value is in separate Dimension. due to this example,**Murder, Rape, Robbery, Assault, Burglary, Larceny, Auto Theft** are the dimensions. Therefore this particular example has 7 dimensions or can be said, Euclidean 7-Space.

when these data represents in a euclidean space each City becomes a vector which has coordinates composed of its attributes. in other hand vector will be composed of 7 coordinates in 7 dimensions.

Since here row matrices will be indicated as (x,y,...)' and column matrices will be indicated as (x,y,...)

For better understanding, If A vector with only 2 coordinates exists on a 2 dimensional space.

$$\underline{\boldsymbol{A}} = (\boldsymbol{a_1}, \boldsymbol{a_2})' = \boldsymbol{a_1}\underline{\boldsymbol{i}} + \boldsymbol{a_2}\underline{\boldsymbol{j}}$$

But if vector in n-dimensional space.

$$\underline{\boldsymbol{B}} = (\boldsymbol{b_1}, \boldsymbol{b_2}, ..., \boldsymbol{b_n})' = \boldsymbol{b_1}\underline{\boldsymbol{v_1}} + \boldsymbol{b_2}\underline{\boldsymbol{v_2}} + ...\boldsymbol{b_n}\underline{\boldsymbol{v_n}}$$

as shown in above there are n number of coordinates represents the vector in n-dimensions space.

3.2 Calculating Euclidean Distance

Now, The euclidean Distance can be calculated using below equation for vector with coordinates $\underline{\boldsymbol{p}} = (\underline{\boldsymbol{p_1}}, \underline{\boldsymbol{p_2}}), \underline{\boldsymbol{q}} = (\underline{\boldsymbol{q_1}}, \underline{\boldsymbol{q_2}})$

$$\boldsymbol{d} = \sqrt{((\boldsymbol{q_1} - \boldsymbol{p_1})^2 + (\boldsymbol{q_2} - \boldsymbol{p_2})^2)}$$

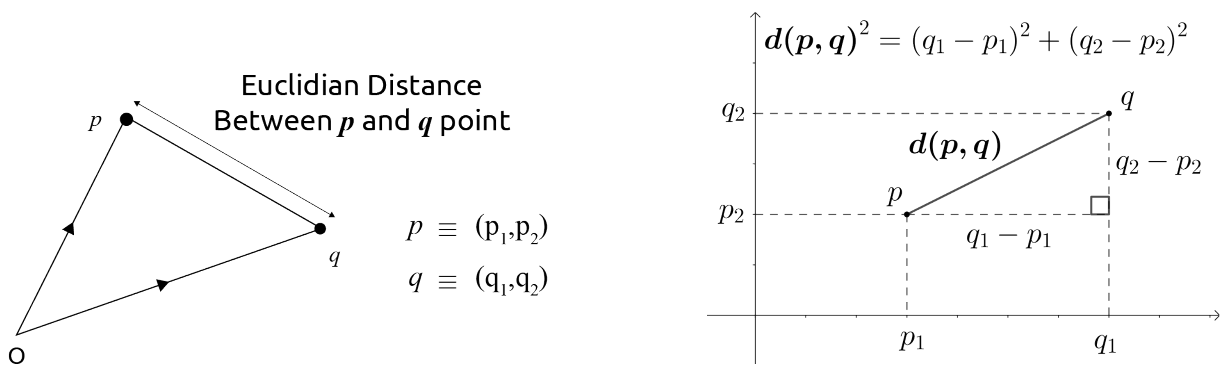

Figure 3.1: Euclidean Distance of two Vectors

Let there are 2 vectors in 2 dimensional space

$$\underline{\boldsymbol{A}} = (\boldsymbol{a_1}, \boldsymbol{a_2})' = \boldsymbol{a_1}\underline{\boldsymbol{i}} + \boldsymbol{a_2}\underline{\boldsymbol{j}} \;\; , \;\; \underline{\boldsymbol{B}} = (\boldsymbol{b_1}, \boldsymbol{b_2},)' = \boldsymbol{b_1}\underline{\boldsymbol{i}} + \boldsymbol{b_2}\underline{\boldsymbol{j}}$$

The euclidean Distance will be

$$\boldsymbol{d} = \sqrt{((\boldsymbol{a_1} - \boldsymbol{b_1})^2 + (\boldsymbol{a_2} - \boldsymbol{b_2})^2)}$$

For 2 vectors in n-dimensional space

$$\underline{\boldsymbol{A}} = (\boldsymbol{a_1}, \boldsymbol{a_2}, ..., \boldsymbol{a_n})' = \ , \ \underline{\boldsymbol{B}} = (\boldsymbol{b_1}, \boldsymbol{b_2}, ..., \boldsymbol{b_n})'$$

d $=\sqrt{(\sum_{i=1}^{n}(\boldsymbol{a_i} - \boldsymbol{b_i})^2)}$ This also equal to d $=\sqrt{(\underline{\boldsymbol{A}} - \underline{\boldsymbol{B}})'(\underline{\boldsymbol{A}} - \underline{\boldsymbol{B}})}$

If $\boldsymbol{D} = \boldsymbol{d_{ij}}$ taken as a matrix, $\boldsymbol{D} = \begin{bmatrix} 0 & .. & d_{1j} & .. & d_{1n} \\ . & . & & & . \\ d_{i1} & & 0 & & d_{in} \\ . & & & . & . \\ d_{n1} & .. & d_{nj} & .. & 0 \end{bmatrix}$

for example, suppose three items have the following bi-variate measurements $(\boldsymbol{y_1}, \boldsymbol{y_2})$: **(2, 5), (4, 2), (7, 9)**.They can be shown like this

Item	y_1	y_2
1	2	5
2	4	2
3	7	9

where **Item 1 = (2,5) , Item 2 = (4,2) , Item 3 = (7,9)**

the matrix $\boldsymbol{D} = (\boldsymbol{d_{ij}})$ for these items

$$\boldsymbol{D} = \begin{bmatrix} 0.0 & 3.6 & 6.4 \\ 3.6 & 0.0 & 7.6 \\ 6.4 & 7.6 & 0.0 \end{bmatrix}$$

Calculations

$$(\boldsymbol{d_{12}}) = (\boldsymbol{d_{21}}) = \sqrt{\begin{bmatrix}(\mathbf{2-4}) & (\mathbf{5-2})\end{bmatrix}\begin{bmatrix}(\mathbf{2-4})\\(\mathbf{5-2})\end{bmatrix}} = \sqrt{\begin{bmatrix}(\mathbf{-2}) & (\mathbf{3})\end{bmatrix}\begin{bmatrix}(\mathbf{-2})\\(\mathbf{3})\end{bmatrix}} = 3.6$$

$$(\boldsymbol{d_{13}}) = (\boldsymbol{d_{31}}) = \sqrt{\begin{bmatrix}(\mathbf{2-7}) & (\mathbf{5-9})\end{bmatrix}\begin{bmatrix}(\mathbf{2-7})\\(\mathbf{5-9})\end{bmatrix}} = \sqrt{\begin{bmatrix}(\mathbf{-5}) & (\mathbf{-4})\end{bmatrix}\begin{bmatrix}(\mathbf{-5})\\(\mathbf{-4})\end{bmatrix}} = 6.4$$

$$(\boldsymbol{d_{23}}) = (\boldsymbol{d_{32}}) = \sqrt{\begin{bmatrix}(\mathbf{4-7}) & (\mathbf{2-9})\end{bmatrix}\begin{bmatrix}(\mathbf{4-7})\\(\mathbf{2-9})\end{bmatrix}} = \sqrt{\begin{bmatrix}(\mathbf{-3}) & (\mathbf{-7})\end{bmatrix}\begin{bmatrix}(\mathbf{-3})\\(\mathbf{-7})\end{bmatrix}} = 7.6$$

$$(\boldsymbol{d_{11}}) = \sqrt{\begin{bmatrix}(\mathbf{2-2}) & (\mathbf{5-5})\end{bmatrix}\begin{bmatrix}(\mathbf{2-2})\\(\mathbf{5-5})\end{bmatrix}} = \sqrt{\begin{bmatrix}(\mathbf{0}) & (\mathbf{0})\end{bmatrix}\begin{bmatrix}(\mathbf{0})\\(\mathbf{0})\end{bmatrix}} = 0$$

$$(\boldsymbol{d_{22}}) = \sqrt{\begin{bmatrix}(\mathbf{4-4}) & (\mathbf{2-2})\end{bmatrix}\begin{bmatrix}(\mathbf{4-4})\\(\mathbf{2-2})\end{bmatrix}} = \sqrt{\begin{bmatrix}(\mathbf{0}) & (\mathbf{0})\end{bmatrix}\begin{bmatrix}(\mathbf{0})\\(\mathbf{0})\end{bmatrix}} = 0$$

$$(\boldsymbol{d_{33}}) = \sqrt{\begin{bmatrix}(\mathbf{7-7}) & (\mathbf{9-9})\end{bmatrix}\begin{bmatrix}(\mathbf{7-7})\\(\mathbf{9-9})\end{bmatrix}} = \sqrt{\begin{bmatrix}(\mathbf{0}) & (\mathbf{0})\end{bmatrix}\begin{bmatrix}(\mathbf{0})\\(\mathbf{0})\end{bmatrix}} = 0$$

Below table shows the result when euclidean distance method was applied to **Example 1**

Atlanta	0	536.6	516.4	590.2	693.6	716.2
Boston	536.6	0	447.4	833.1	915	881.1
Chicago	516.4	447.4	0	924	1073.4	971.5
Dallas	590.2	833.1	924	0	527.7	464.5
Denver	693.6	915	1073.4	527.7	0	358.7
Detroit	716.2	881.1	971.5	464.5	358.7	0

Table 3.2: Example (1.2) (Euclidean Distance Matrix)

3.3 Clustering Methods

3.3.1 Single Linkage method(Nearest Neighbor)

In the single linkage method, the distance between two **clusters A and B** is defined as the **minimum** distance between **a point in A and a point in B**:

$$\mathbf{D(A,\ B)} = \min \boldsymbol{\{dy_i, y_j\}}, \text{ for } \boldsymbol{y_i} \text{ in } \mathbf{A} \text{ and } \boldsymbol{y_j} \text{ in } \mathbf{B}$$

where $\boldsymbol{d(y_i, y_j)}$ is the Euclidean distance between the vectors $\boldsymbol{y_i}$ and $\boldsymbol{y_j}$.

Step1

The distance $\mathbf{D(A,\ B)}$ is found for every pair of clusters

Step2

Two clusters with smallest distance are merged. Smallest value of every pair will be the only remaining one after merging.

Step3

The number of clusters is therefore reduced by 1.

The process is repeated for the following stage after merging two clusters. All cluster pair distances are once more calculated, and the pair with the smallest distance is combined into a single cluster. Until there is just one cluster left, this operation will be repeated.

Example 1 Continued..

The smallest distance is **358.7** between **Denver** and **Detroit**, and therefore these two cities are joined at the first step to form **C1 = {Denver, Detroit}**. In the next step, the distance matrix is calculated for **Atlanta, Boston, Chicago, Dallas,** and **C1**

Atlanta	0	536.7	516.4	590.2	693.6
Boston	536.6	0	447.4	833.1	881.1
Chicago	516.4	447.4	0	924.0	971.5
Dallas	590.2	833.1	924.0	0	464.5
C1	693.6	881.1	971.5	464.5	0

Table 3.3: Example (1.3) Reduced matrix 1(Single)

The smallest distance is **447.4** between **Boston** and **Chicago**. Therefore **C2 = {Boston, Chicago}**. At the next step, the distance matrix is calculated for **Atlanta, Dallas, C1**, and **C2**:

Atlanta	0	516.4	590.2	693.6
C2	516.4	0	833.1	881.1
Dallas	590.2	833.1	0	464.5
C1	693.6	881.1	464.5	0

Table 3.4: Example (1.4) Reduced matrix 2(Single)

The smallest distance is **464.5** between **Dallas** and **C1**, so that **C3 = Dallas, C1**. The distance matrix for **Atlanta, C2,** and **C3** is given by

Atlanta	0	516.4	590.2
C2	516.4	0	833.1
C3	590.2	833.1	0

Table 3.5: Example (1.5) Reduced matrix 3(Single)

The smallest distance is **516.4**, which defines **C4 = Atlanta, C2**. The distance matrix for **C3** and **C4** is

C4	0	590.2
C3	590.2	0

Table 3.6: Example (1.6) Reduced matrix 4(Single)

Dendrogram of Single linkage method is illustrated below

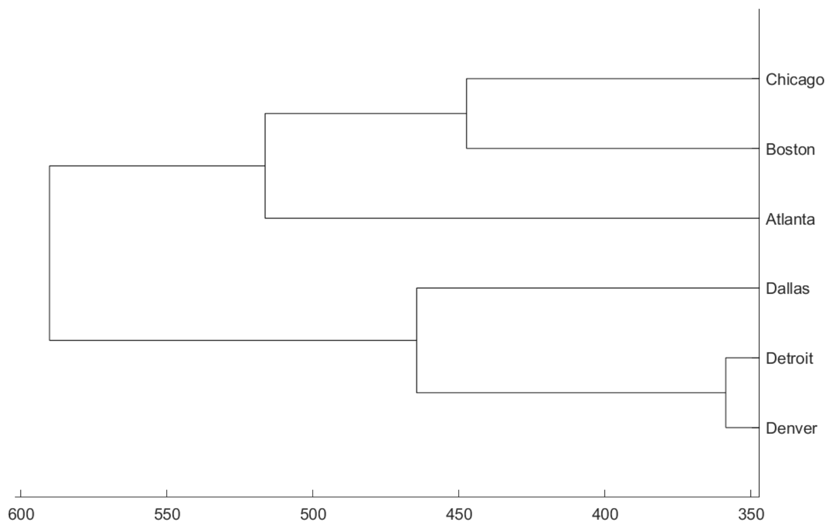

Figure 3.2: Single Linkage Dendrogram

3.3.2 Complete Linkage method(Farthest Neighbor)

In the complete linkage approach, the distance between two **clusters A and B** is defined as the **maximum** distance between a point in A and a point in B:

$$\mathbf{D(A,\ B)} = \max\ \{\boldsymbol{dy_i, y_j}\},\ \text{for } \boldsymbol{y_i} \text{ in } \mathbf{A} \text{ and } \boldsymbol{y_j} \text{ in } \mathbf{B}$$

where $\boldsymbol{d(y_i, y_j)}$ is the Euclidean distance between the vectors $\boldsymbol{y_i}$ and $\boldsymbol{y_j}$.

Step1

The distance $\mathbf{D(A,\ B)}$ is found for every pair of clusters

Step2

Two clusters with smallest distance are merged. largest value of every pair will be the only remaining one after merging.

Step3

The number of clusters is therefore reduced by 1.

The process is repeated for the following stage after merging two clusters. All cluster pair distances are once more calculated, and the pair with the smallest distance is combined into a single cluster. Until there is just one cluster left, this operation will be repeated.

Example 1 Continued..

The smallest distance is **358.7** between **Denver** and **Detroit**, and therefore these two cities are joined at the first step to form $\mathbf{C1 = \{Denver,\ Detroit\}}$. In the next step, the distance matrix is calculated for **Atlanta, Boston, Chicago, Dallas,** and **C1**

Atlanta	0	536.6	516.4	590.2	716.2
Boston	536.6	0	447.4	833.1	915.0
Chicago	516.4	447.4	0	924.0	1073.4
Dallas	590.2	833.1	924.0	0	527.7
C1	716.2	915.0	1073.4	527.7	0

Table 3.7: Example (1.7) Reduced matrix 1(Complete)

The smallest distance is **447.4** between **Boston** and **Chicago**. Therefore **C2 = {Boston, Chicago}**. At the next step, the distance matrix is calculated for **Atlanta, Dallas, C1**, and **C2**:

Atlanta	0	536.6	590.2	716.2
C2	536.6	0	833.1	915.0
Dallas	590.2	833.1	0	527.7
C1	716.2	1073.4	527.7	0

Table 3.8: Example (1.8) Reduced matrix 2(Complete)

The smallest distance is **527.7** between **Dallas** and **C1**, so that **C3 = Dallas, C1**. The distance matrix for **Atlanta, C2,** and **C3** is given by

Atlanta	0	536.6	716.2
C2	536.6	0	1073.4
C3	716.2	1073.4	0

Table 3.9: Example (1.9) Reduced matrix 3(Complete)

The smallest distance is **536.6**, which defines **C4 = Atlanta, C2**. The distance matrix for **C3** and **C4** is

C4	0	1073.4
C3	1073.4	0

Table 3.10: Example (1.10) Reduced matrix 4(Complete)

Dendrogram of Complete linkage method is illustrated below

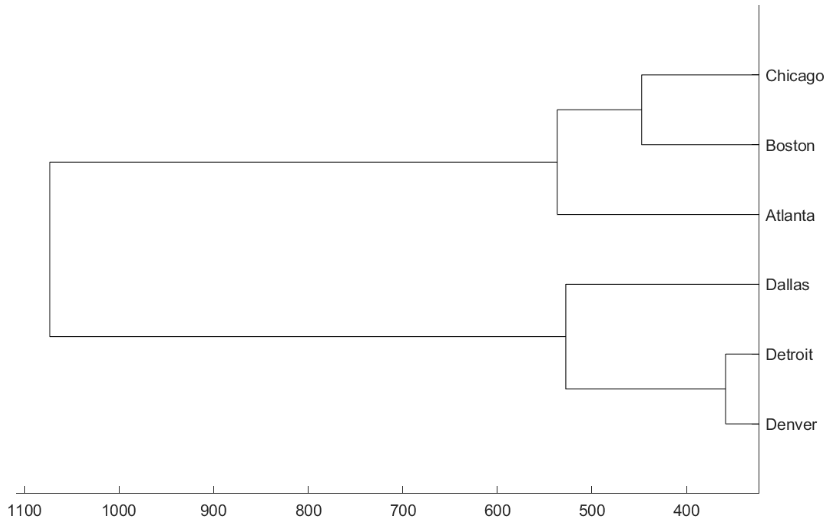

Figure 3.3: Complete Linkage Dendrogram

3.3.3 Average Linkage method

The average of the $\boldsymbol{n_A n_B}$ distances between the $\boldsymbol{n_A}$ points in **A** and the $\boldsymbol{n_B}$ points in **B** is used to establish the distance between **two clusters A and B** in the average linkage approach:

$$\boldsymbol{D(A,B) = \frac{1}{n_A n_B} \sum_{i=1}^{n_A} \sum_{j=1}^{n_B} d(y_i, y_j)}$$

where the total includes every $\boldsymbol{y_i}$ in **A** and every $\boldsymbol{y_j}$ in **B**. We combine the two clusters with the least distance between them at each step using above equation.

Dendrogram of Average Linkage method is illustrated below

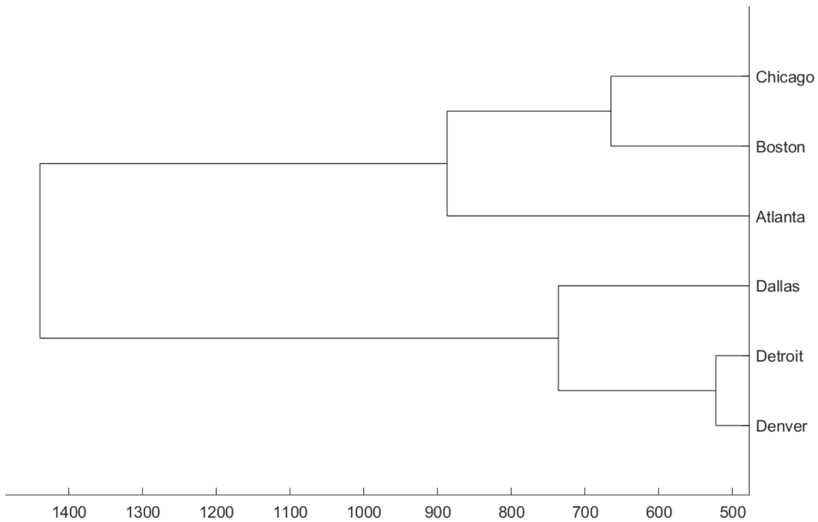

Figure 3.4: Average Linkage Dendrogram

3.3.4 Ward's method

The within-cluster (squared) distances and between-cluster (squared) distances are used in Ward's approach, commonly known as the incremental sum of squares method (Ward 1963, Wishart 1969a). If cluster AB is the cluster created by joining clusters A and B, then the items from the cluster mean vectors' within-cluster distances are

$$\boldsymbol{SSE_A = \sum_{i=1}^{nA} (y_i - \overline{y}_A)'(y_i - \overline{y}_A),}$$

$$\boldsymbol{SSE_B = \sum_{i=1}^{nB} (y_i - \overline{y}_B)'(y_i - \overline{y}_B),}$$

$$\boldsymbol{SSE_{AB} = \sum_{i=1}^{nAB} (y_i - \overline{y}_{AB})'(y_i - \overline{y}_{AB})}$$

$\boldsymbol{n_A}$, $\boldsymbol{n_B}$, and $\boldsymbol{n_{AB} = n_A + n_B}$ are the corresponding numbers of points in **A**, **B**, and **AB**. Where $\boldsymbol{y_{AB} = (n_A y_A + n_B y_B)/(n_A + n_B)}$, as in

$$\boldsymbol{\overline{y}_{AB} = \frac{n_A \overline{y}_A + n_B \overline{y}_B}{n_A + n_B}}$$

$\boldsymbol{SSE_A}$, $\boldsymbol{SSE_B}$, and $\boldsymbol{SSE_{AB}}$ are used to represent these sums of distances since they are the same as within-cluster sums of squares. The two clusters **A** and **B** that reduce the rise in **SSE**, which is defined as,

$$\boldsymbol{I_{AB} = SSE_{AB} - (SSE_A + SSE_B)}$$

are joined by Ward's approach.

Dendrogram of Ward's method is illustrated below

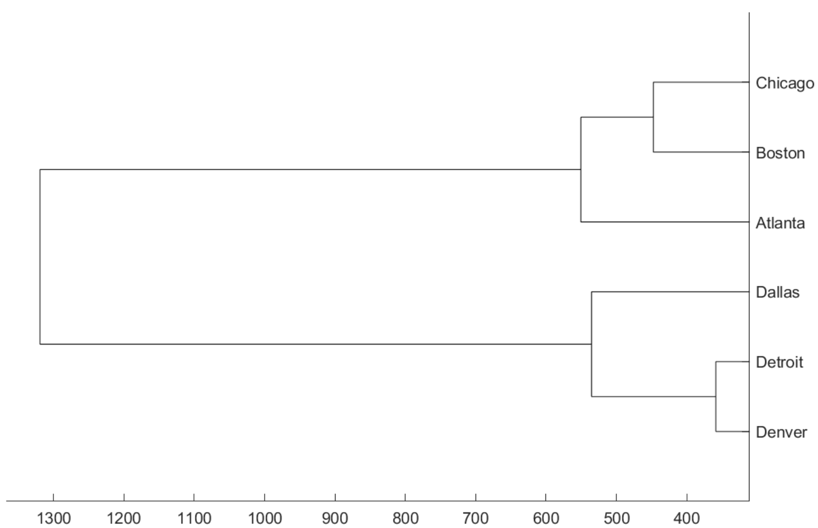

Figure 3.5: Ward's Method Dendrogram

3.3.5 Centroid method

The centroid method defines the distance between two clusters A and B as the Euclidean distance between the two clusters' mean vectors (also known as centroids)

$$\boldsymbol{D(A,B) = d(y_A, y_B)}$$

where $\overline{\boldsymbol{y}}_{\boldsymbol{A}}$ and $\overline{\boldsymbol{y}}_{\boldsymbol{B}}$ are the mean vectors for the observation vectors in **A** and **B**, respectively, and d(yA, yB) is defined in

$$\text{d} = \sqrt{\left(\sum_{i=1}^{n} (\boldsymbol{a_i} - \boldsymbol{b_i})^2\right)} \quad \text{This also equal to} \quad \text{d} = \sqrt{(\underline{\boldsymbol{A}} - \underline{\boldsymbol{B}})'(\underline{\boldsymbol{A}} - \underline{\boldsymbol{B}})}$$

We define $\overline{\boldsymbol{y}}_{\boldsymbol{A}}$ and $\overline{\boldsymbol{y}}_{\boldsymbol{B}}$ as usual, that is,$\overline{\boldsymbol{y}}_{\boldsymbol{A}} = \left(\sum_{i=1}^{n_A} (\boldsymbol{y_i}/\boldsymbol{n_A})\right)$. At each phase, the two clusters with the shortest distance between centroids are combined.

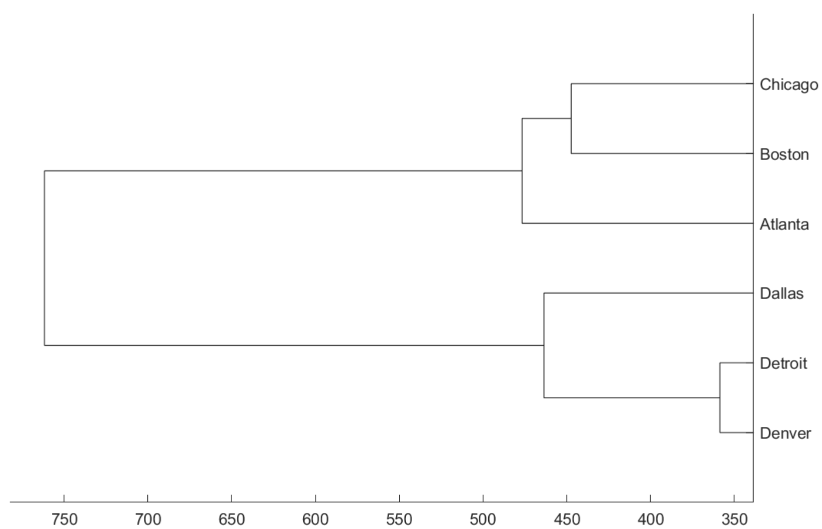

Figure 3.6: Centroid Method Dendrogram

3.3.6 Median method

If **two clusters A** and **B** are joined using the centroid method, and **A** has more items than **B**, the new centroid $\boldsymbol{y_{AB}} = (\boldsymbol{n_A y_A} + \boldsymbol{n_B y_B})/(\boldsymbol{n_A} + \boldsymbol{n_B})$ may be significantly closer to $\boldsymbol{y_A}$ than to $\boldsymbol{y_B}$. To avoid weighting the mean vectors based on cluster size, we can compute fresh distances to additional clusters using the median (midpoint) of the line connecting **A** and **B**:

$$\boldsymbol{m_{AB}} = \frac{\mathbf{1}}{\mathbf{2}}(\overline{\boldsymbol{y}}_{\boldsymbol{A}} + \overline{\boldsymbol{y}}_{\boldsymbol{B}}).$$

It is worth noting that the median is not the standard statistical median.

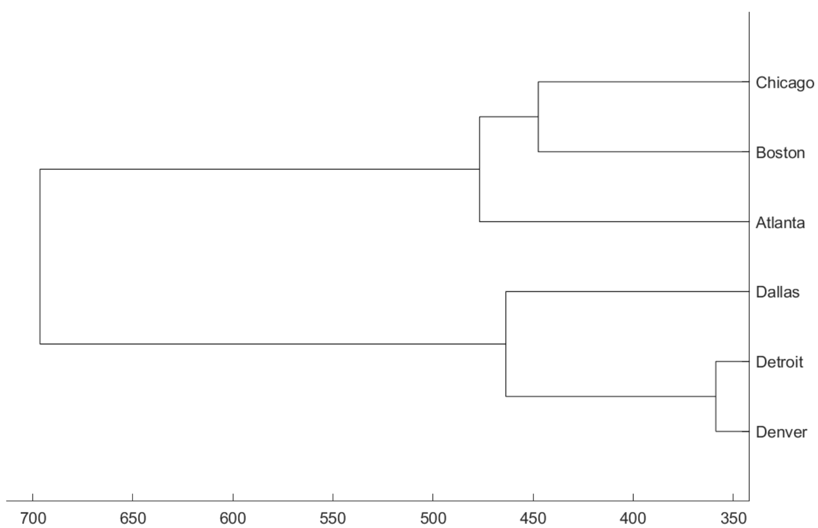

Figure 3.7: Median Method Dendrogram

CHAPTER 4
DISCUSSION

Classification or grouping is very important in a statistical study and helps to make the study easier. The importance of this is felt very well in the study of big data. The project developed a program for this classification of rather large data using Matlab. Hierarchical clustering method under cluster analysis was used here. In this method, the data is arranged according to a certain level of importance.

- Single linkage method
- Complete linkage method
- Average Linkage method
- Wards method
- Median method
- Centroid method

are included in this program. At the start of this program, the user can select the desired method from among the options. Second, the input can be given as a Excel File(.xlsx). Thus given inputs are processed and given as outputs Excel File(.xlsx) and Image File(.png). Here, the matrix is given as the Excel File and the dendrogram as the image. The names of the output files are made up of the method name and the date and time. Therefore, the output files obtained by this program will not be confused with each other. At the end of each clustering method, the user is asked if the program should be run again. This program can be used for all clustering methods again and again depending on the user's command.

Below are the images taken when the program was executed.

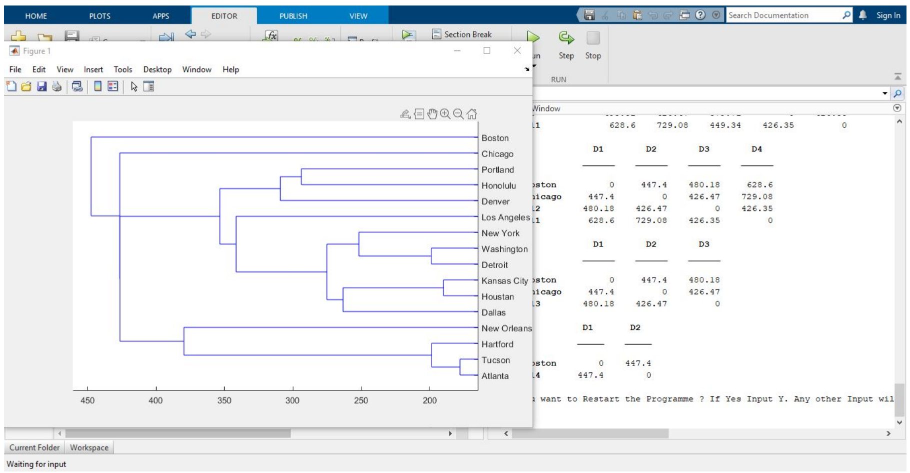

Figure 4.1: Screenshot (1)

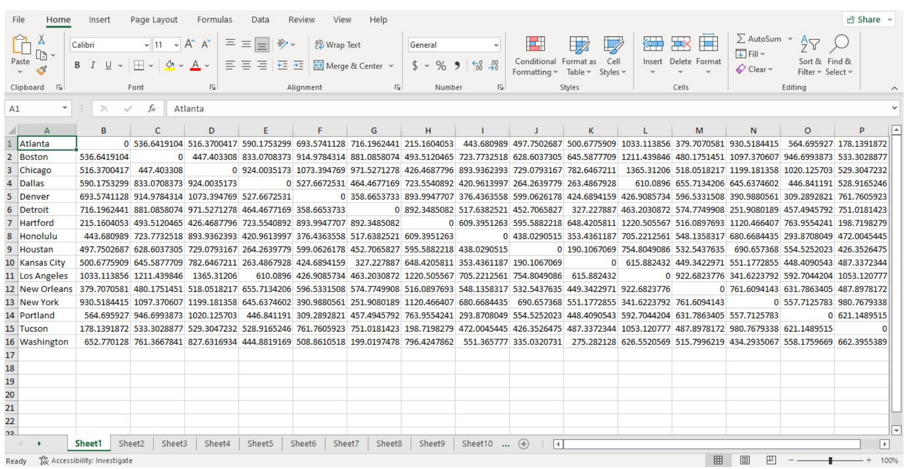

Figure 4.2: Screenshot (2)

Usually these clusters can be represented by matrices at each level. In this program, it was possible to create level matrices in single and complete methods, but for Average, Wards's, Median and Centroid methods could not able to create level matrices. That can be cited as one of the drawbacks of this program. On the one hand, it could be recognized that the calculations of obtaining the metrics are a little complicated and difficult to program. On the other hand, it should be said that there were weaknesses in our programming and the shortcomings in the study of that scope.

And there is no way to check the validity of the data entered in this program or whether the clustering method chosen for that data is appropriate. That is one of the disadvantages of this program. And this program does not have the ability to measure the correlation at the end of the clustering method. Matlab itself or another programming language can be used to avoid these above-mentioned shortcomings. In particular, using a programming language with artificial intelligence capabilities can achieve more effective results. And as an improvement, non-hierarchical methods can be added to this program.

During this project we learned more about the importance of clustering, clustering methods and their uses. We also gained proficiency in using Matlab. We look forward to using them in another project.

CHAPTER 5
CONCLUSION

In this project we had to create a clustering program using Matlab. We are happy to say that we have been able to do it successfully to some extent. Here 'somewhat' means we were able to add all the hierarchical methods but not the non-hierarchical methods. Also some hierarchical methods could not display level matrices. But this program is able to process the data provided and give required outputs. Then the given dendrograms can lead to conclusions. And this program is very user friendly. So easy to use. In the end we are satisfied with this project.

REFERENCES

[1] A.C. Rencher. *Methods of Multivariate Analysis*. Wiley Series in Probability and Statistics. Wiley, 2003.

[2] G. Gan, C. Ma, and J. Wu. *Data Clustering: Theory, Algorithms, and Applications*. ASA-SIAM Series on Statistics and Applied Probability. Society for Industrial and Applied Mathematics, 2007.

[3] S.N. Alam, S.N. Alam, and S.K. Patel. *Advanced Guide to MATLAB: Practical Examples in Science and Engineering*. I.K. International Publishing House Pvt. Limited, 2015.

[4] N. Majumdar and S. Banerjee. *MATLAB Graphics and Data Visualization Cookbook*. Quick answers to common problems. Packt Pub., 2012.

CHAPTER 6
APPENDIX

6.1 Matlab Code

```matlab

clear;
clc;

% Input for linkage selection ----------------------------------------------
fprintf('Hello there , Please choose your Clustering method.\n\n')

fprintf('Your Options are\n\n')

fprintf(' 1 - Single Linkage method\n')
fprintf(' 2 - Complete Linkage method\n')
fprintf(' 3 - Average Linkage method\n')
fprintf(' 4 - 'Wards Method\n')
fprintf(' 5 - Centroid method\n')
fprintf(' 6 - Median method\n\n')

Sel = input('Please Enter your choice - ');
%--------------------------------------------------------------------------

% Check for input validity
% ----------------------------------------------

while isempty(Sel) == 1 || (Sel <1 || Sel >6 || (floor(Sel) ~= ceil(Sel)))
   clear Sel
   Sel = input('\n\nPlease enter a valid number - ');
end


```

```
%---------------------------------------------------------------------

%Check selection and start ---------------------------------------------
if Sel == 1
   fprintf('\n\nYou Have chosen Single Linkage Method\n');
   input('\n\nIs your Input File Ready ? If YES Press enter\n\n','s');
   [A,C,D,m,n] = Input();
   SingleLinkage(A,D,C,n,m);

elseif Sel == 2
   fprintf('\n\nYou Have chosen Complete Linkage Method\n');
   input('\n\nIs your Input File Ready ? If YES Press enter\n\n','s');
   [A,C,D,m,n] = Input();
    CompleteLinkage(A,D,C,n,m);

elseif Sel == 3
   fprintf('\n\nYou Have chosen Average Linkage Method\n');
   input('\n\nIs your Input File Ready ? If YES Press enter\n\n','s');
   [A,C,D,m,n] = Input();
    AverageLinkage(A,D,C,n,m);

elseif Sel == 4
   fprintf('\n\nYou Have chosen Ward s Method\n');
   input('\n\nIs your Input File Ready ? If YES Press enter\n\n','s');
   [A,C,D,m,n] = Input();
    Wards(A,D,C,n,m);

elseif Sel == 5
   fprintf('\n\nYou Have chosen Centroid Method\n');
   input('\n\nIs your Input File Ready ? If YES Press enter\n\n','s');
   [A,C,D,m,n] = Input();
    centroid(A,D,C,n,m);

```

```
elseif Sel == 6
   fprintf('\n\nYou Have chosen Median Method\n');
   input('\n\nIs your Input File Ready ? If YES Press enter\n\n','s');
   [A,C,D,m,n] = Input();
    median(A,D,C,n,m);

end


Res=input('Do you want to Restart the Programme ? If Yes Input Y. Any other Input will consider as No\n\n','s');
if Res=='Y'
   run('Final.m');
else
   fprintf('                                  <strong>------------PROGRAMME ENDED | THANK YOU-----------</strong>\n\n\n')
end
%-----------------------------------------------------------------------

% Input taking Function--------------------------------------------------
function [A,C,D,m,n] = Input()

Inp_Tbl = readtable('input.xlsx','ReadRowNames',true);
row_num = (height(Inp_Tbl)+1);
H= strcat('A1:A',num2str(row_num));
Row_names = readtable('input.xlsx','Range',H);
NTbl2mat=table2array(Inp_Tbl);
CTbl2mat=table2array(Row_names);

   n=(row_num-1);

   m=n;

```

```
    C=transpose(CTbl2mat);
    A = NTbl2mat;
    B= pdist(A);
    D =squareform(B);
end
%-----------------------------------------------------------------------
% Single Linkage Function-----------------------------------------------
function SingleLinkage(A,D,C,n,m)
    fprintf('                                  <strong>Single Linkage method(Nearest Neighbor)</strong>\n\n\n')
    CC=C;
    DT=array2table(D,'RowNames',C);
    disp(DT)
    % generating file name ----------------------------------------------
    Datetime = datestr(now,'mmmm_dd_yyyy_HH_MM_SS_PM');
    DateName1= strcat('output_Table(Single_Linkage)_',Datetime,'.xlsx');
    DateName2= strcat('output_Dendrogram(Single_Linkage)_',Datetime,'.png');
    %--------------------------------------------------------------------
    %File Creating ------------------------------------------------------
    writetable(DT,DateName1,'Sheet',1,'WriteRowNames',true,'WriteMode',
    'overwritesheet','AutoFitWidth',true,'PreserveFormat',true,
    'WriteVariableNames',false);
    %--------------------------------------------------------------------
    %Loop for cluster process -------------------------------------------
```

```
for k=1:m-2
   [numRows,numCols] = size(D);
   U=zeros(numRows,numCols);
   V=zeros(numRows,numCols);

   % Finding minimum number -----------------------------------------
   min=10000000000000000000000000000;
   for i=1:n
      for j=1:n
         if D(i,j) >0
            if D(i,j) < min
               min=D(i,j);
               min_i = i;
               min_j= j;
            end
         end
      end
   end
   %----------------------------------------------------------------

   % loop For Horizontal minimum chekup -----------------------------
   for i=min_j
      for j=1:numCols
         if D(i,j) >0
            if D(i,j) < D(min_i,j)
               V(i,j)= D(i,j);
            else
              V(i,j)= D(min_i,j);
            end
         end
      end
   end
   [numRows,numCols] = size(D);
```

```
%-----------------------------------------------------------------

% loop For VERTICAL minimum chekup -------------------------------
for j=min_j
   for i=1:numRows
      if D(i,j) >0
         if D(i,j) < D(i,min_i)
           U(i,j)= D(i,j);
         else
           U(i,j)= D(i,min_i);
         end
      end
   end
end
%-----------------------------------------------------------------

%reshaping distance matrix ---------------------------------------
for i= min_j
   for j=1:numCols
      D(i,j) = V(i,j);
   end
end

for j= min_j
   for i=1:numRows
      D(i,j) = U(i,j);
   end
end

D(min_i,:)=[];
D(:,min_i)=[];
%-----------------------------------------------------------------

```

```
    % Auto clustre name generator-------------------------------------
    [numRowsCity,numColsCity] = size(C);
    CN = num2str(abs(m-n+1));
    CNN = ['C' CN];
    %-----------------------------------------------------------------

    % adding genereted name to city matrix----------------------------
    C2 = [C(1 : min_j-1),CNN,C(min_j+1:numColsCity)];
    C2(:,min_i)=[];
    C=C2;
    %-----------------------------------------------------------------

    %Creating display table-------------------------------------------
    [numRows,numCols] = size(D);
    DT=array2table(D,'RowNames',C2);
    disp(DT)

    sheet = k+1;
    writetable(DT,DateName1,'Sheet',1,'WriteRowNames',true,'WriteMode',
    'overwritesheet','AutoFitWidth',true,'PreserveFormat',true,
    'WriteVariableNames',false);
    %-----------------------------------------------------------------

    n= n-1;
end
%---------------------------------------------------------------------

%Creating dedrogram---------------------------------------------------
BB= pdist(A);
tree = linkage(A,'single');
leafOrder = optimalleaforder(tree,BB);
%create cell of labels
labels = cellstr(CC);
```

```
    %plot dendogram with custom labels
    dendrogram(tree, 0, 'Labels', labels, 'orientation', 'left')
    saveas(gcf,DateName2)
    %--------------------------------------------------------------------

end
%-----------------------------------------------------------------------

% complete Linkage Function---------------------------------------------
function CompleteLinkage(A,D,C,n,m)

    fprintf('                                  <strong>Complete Linkage method(Farthest Neighbor)</strong>\n\n\n')

    CC=C;
    DT=array2table(D,'RowNames',C);
    disp(DT)

    % generating file name ---------------------------------------------
    Datetime = datestr(now,'mmmm_dd_yyyy_HH_MM_SS_PM');
    DateName1= strcat('output_Table(complete_Linkage)_',Datetime,'.xlsx');
    DateName2= strcat('output_Dendrogram(complete_Linkage)_',Datetime,'.png');
    %--------------------------------------------------------------------

    %File Creating -------------------------------------------------------
    writetable(DT,DateName1,'Sheet',1,'WriteRowNames',true,'WriteMode',
    'overwritesheet','AutoFitWidth',true,'PreserveFormat',true,
    'WriteVariableNames',false);
    %--------------------------------------------------------------------



    %Loop for cluster process ------------------------------------------
```

```
  for k=1:m-2
     [numRows,numCols] = size(D);
     U=zeros(numRows,numCols);
     V=zeros(numRows,numCols);

     % Finding maximum number -----------------------------------------
     min=1000000000000000000000;
     for i=1:n
        for j=1:n
           if D(i,j) >0
              if D(i,j) < min
                 min=D(i,j);
                 max_i = i;
                 max_j= j;
              end
           end
        end
     end
     %-----------------------------------------------------------------

     % For Horizontal maximum chekup ----------------------------------
     for i=max_j
        for j=1:numCols
           if D(i,j) >0
              if D(i,j) > D(max_i,j)
                 V(i,j)= D(i,j);
              else
                V(i,j)= D(max_i,j);
              end
           end
        end
     end
     [numRows,numCols] = size(D);
```

```
%-----------------------------------------------------------------

% For VERTICAL maximum chekup ---------------------------------------
for j=max_j
   for i=1:numRows
      if D(i,j) >0
         if D(i,j) > D(i,max_i)
           U(i,j)= D(i,j);
         else
           U(i,j)= D(i,max_i);
         end
      end
   end
end
%-----------------------------------------------------------------

%reshaping distance matrix ------------------------------------------
for i= max_j
   for j=1:numCols
      D(i,j) = V(i,j);
   end
end

for j= max_j
   for i=1:numRows
      D(i,j) = U(i,j);
   end
end

D(max_i,:)=[];
D(:,max_i)=[];
%-----------------------------------------------------------------

```

```
    % Auto clustre name generator-----------------------------------------
    [numRowsCity,numColsCity] = size(C);
    CN = num2str(abs(m-n+1));
    CNN = ['C' CN];
    %--------------------------------------------------------------------

    % adding genereted name to city matrix--------------------------------
    C2 = [C(1 : max_j-1),CNN,C(max_j+1:numColsCity)];
    C2(:,max_i)=[];
    C=C2;
    %--------------------------------------------------------------------

    %Creating display table-----------------------------------------------
    [numRows,numCols] = size(D);
    DT=array2table(D,'RowNames',C2);
    disp(DT)
    sheet = k+1;

    writetable(DT,DateName1,'Sheet',sheet,
    'WriteRowNames',true,'WriteMode',
    'overwritesheet','AutoFitWidth',true,
    'PreserveFormat',true,'WriteVariableNames
    ',false);
    %--------------------------------------------------------------------

    n= n-1;
  end
  %--------------------------------------------------------------------

  %Creating dedrogram-----------------------------------------------------
  BB= pdist(A);
  tree = linkage(A,'complete');
  leafOrder = optimalleaforder(tree,BB);
```

```
    %create cell of labels
    labels = cellstr(CC);
    %plot dendogram with custom labels
    dendrogram(tree, 0, 'Labels', labels, 'orientation', 'left')
    saveas(gcf,DateName2)
    %---------------------------------------------------------------------
end
%-------------------------------------------------------------------------

% Average Linkage Function------------------------------------------------
function AverageLinkage(A,D,C,n,m)

    na= n;
    nb= n;
    x= A;
    y= A;

    fprintf('                                  <strong>Average Linkage method</strong>\n\n\n')

    CC=C;
    D =pdist(A);
    D2=squareform(D);

    DT=array2table(D2,'RowNames',CC);

    % generating file name ----------------------------------------------
    Datetime = datestr(now,'mmmm_dd_yyyy_HH_MM_SS_PM');
    DateName1= strcat('output_Table(Average_Linkage)_',Datetime,'.xlsx');
    DateName2= strcat('output_Dendrogram(Average_Linkage)_',Datetime,'.png');
    %--------------------------------------------------------------------

    %File Creating -------------------------------------------------------
```

```
    writetable(DT,DateName1,'Sheet',1,'WriteRowNames',true,'WriteMode',
    'overwritesheet','AutoFitWidth',true,'PreserveFormat',true,
    'WriteVariableNames',false);
    %---------------------------------------------------------------------

    %Creating Dendrigram--------------------------------------------------
    disp(DT)
    %generating tree
    tree = linkage(D2,'average');
    %create cell of labels
    labels = cellstr(CC);
    %plot dendogram with custom labels
    dendrogram(tree, 0, 'Labels', labels, 'orientation', 'left')
    saveas(gcf,DateName2)
    %---------------------------------------------------------------------
end
%-------------------------------------------------------------------------

% wards Function---------------------------------------------------------
function Wards(A,D,C,n,m)

    na=n;
    nb=n;
    x=A;
    y=A;

    CC=C;
    DT=array2table(D,'RowNames',C);
    disp(DT)

    % generating file name -----------------------------------------------
    Datetime = datestr(now,'mmmm_dd_yyyy_HH_MM_SS_PM');
    DateName1= strcat('output_Table(Wards)_',Datetime,'.xlsx');
```

```
   DateName2= strcat('output_Dendrogram(Wards)_',Datetime,'.png');
   %---------------------------------------------------------------------

   %creating file--------------------------------------------------------
   writetable(DT,DateName1,'Sheet',1,'WriteRowNames',true,'WriteMode',
   'overwritesheet','AutoFitWidth',true,'PreserveFormat',true,
   'WriteVariableNames',false);
   %---------------------------------------------------------------------

   %Creating dedrogram----------------------------------------------------
   BB= pdist(A);
   tree = linkage(A,'ward');
   leafOrder = optimalleaforder(tree,BB);
   %create cell of labels
   labels = cellstr(CC);
   %plot dendogram with custom labels
   dendrogram(tree, 0, 'Labels', labels, 'orientation', 'left')
   saveas(gcf,DateName2)
   %----------------------------------------------------------------------

end
%--------------------------------------------------------------------------

% wards Function----------------------------------------------------------
function centroid(A,D,C,n,m)

   na=n;
   nb=n;
   x=A;
   y=A;

   CC=C;
   DT=array2table(D,'RowNames',C);
```

```
   disp(DT)

   % generating file name ----------------------------------------------
   Datetime = datestr(now,'mmmm_dd_yyyy_HH_MM_SS_PM');
   DateName1= strcat('output_Table(Centroid)_',Datetime,'.xlsx');
   DateName2= strcat('output_Dendrogram(Centroid)_',Datetime,'.png');
   %--------------------------------------------------------------------

   %creating file-------------------------------------------------------
   writetable(DT,DateName1,'Sheet',1,'WriteRowNames',true,'WriteMode',
   'overwritesheet','AutoFitWidth',true,'PreserveFormat',true,
   'WriteVariableNames',false);
   %--------------------------------------------------------------------

   %Creating dedrogram---------------------------------------------------
   BB= pdist(A);
   tree = linkage(A,'centroid');
   leafOrder = optimalleaforder(tree,BB);
   %create cell of labels
   labels = cellstr(CC);
   %plot dendogram with custom labels
   dendrogram(tree, 0, 'Labels', labels, 'orientation', 'left')
   saveas(gcf,DateName2)
   %---------------------------------------------------------------------
end
%-------------------------------------------------------------------------

% wards Function---------------------------------------------------------
function median(A,D,C,n,m)

   na=n;
   nb=n;
   x=A;
```

```
    y=A;

    CC=C;
    DT=array2table(D,'RowNames',C);
    disp(DT)

    % generating file name -----------------------------------------------
    Datetime = datestr(now,'mmmm_dd_yyyy_HH_MM_SS_PM');
    DateName1= strcat('output_Table(Median)_',Datetime,'.xlsx');
    DateName2= strcat('output_Dendrogram(Median)_',Datetime,'.png');
    %--------------------------------------------------------------------

    %creating file------------------------------------------------------
    writetable(DT,DateName1,'Sheet',1,'WriteRowNames',true,'WriteMode',
    'overwritesheet','AutoFitWidth',true,'PreserveFormat',true,
    'WriteVariableNames',false);
    %--------------------------------------------------------------------

    %Creating dedrogram--------------------------------------------------
    BB= pdist(A);
    tree = linkage(A,'median');
    leafOrder = optimalleaforder(tree,BB);
    %create cell of labels
    labels = cellstr(CC);
    %plot dendogram with custom labels
    dendrogram(tree, 0, 'Labels', labels, 'orientation', 'left')
    saveas(gcf,DateName2)
    %---------------------------------------------------------------------
end
%-----------------------------------------------------------------------
```