Introduction to Intelligent Systems
Lab Session 1

Group 17
Tobias Pucher (s5751659) & Thanakit Chaichanawong (s5752094)

September 27, 2023

Assignment 1

1. Introduction

The amount of data for analysis purpose is getting larger and larger as time goes on and technology improves. Therefore traditional analysis on raw data is going to be harder as well. Fortunately, with the introduction of machine learning, there exist various algorithms, functions, and libraries to help mitigate such problems. One of the example is clustering and cluster analysis.

Cluster analysis is the analysis of data by partitioning groups of data points by the factor of similarity between data points. More similarity between data points means more probability they will be in the same cluster, and vice versa, the criteria is depending on the configuration or methods used for determining such similarity. Cluster analysis is considered to be a unsupervised learning technique. The analysis is static with the result variation dependent on the algorithm used and of course data. It is one type of exploratory data analysis, which main purpose is to summarize the principal characteristic of the given data set and visualize such groups/clusters.

There are many different clustering algorithms available with different purpose and result. Centroid based clustering is the analysis with non-hierarchical cluster but high efficiency. K-means is the most common algorithm of Centroid basedGoogle [2] analysis. Next, Density based is the clustering of data points by detecting the area where data is concentrated. DBSCAN is one of such algorithms [1]. Distribution based clustering is done by grouping data points based on probability of belonging to a given statistical dsitribution such as Gaussian and BinomialJoshi [3]. Lastly, connectivity-based clustering is based on similarity between data points, one prime example is hierarchical clustering. Agglomerative clustering, one of the algorithms used for hierachical clustering, works from the bottom-up by grouping datapoints. Top down is when the data is divided instead of grouped. When combining points to clusters, the algorithm is merging the most similar pairs (points and clusters) together from the closest one, until the last one A.K.A. root node emerges. For visualization purposes only certain levels can be observed, often times starting from the root node.

2. Methods

2.1. AgglomerativeClustering function

```
from sklearn.cluster import AgglomerativeClustering
label = AgglomerativeClustering(n_clusters= int(k),linkage='single').
    fit(data_array).labels_
```

AgglomerativeClustering is a function of SKlearn, it encoprate 3 function together which are:

1. Distance between nodes
2. Linkage between nodes
3. labeling data points

2.2. Distance

```
from scipy.spatial.distance import pdist
dist = pdist(data_array,'euclidean')
```

First and foremost, the distance matrix is 2 a dimension array of square matrix, which the length of the Axis depending on number of data points available in data set. In this case with 200 distinct data points, the distance matrix is 200 on the X-axis and 200 on the Y-axis. The content of the distance matrix are the distances between each node. When visualized, on the same x and y index, the distance will be 0 as it measure to and from the same point. The matrix is symmetrical along its diagonal.

The distance matrix can be calculate with different methods of calculation. The distance function can be 'braycurtis', 'canberra', 'chebyshev', 'cityblock', 'correlation', 'cosine', 'dice', 'euclidean', 'hamming', 'jaccard', 'jensenshannon', 'kulczynski1', 'mahalanobis', 'matching', 'minkowski', 'rogerstanimoto', 'russellrao', 'seuclidean', 'sokalmichener', 'sokalsneath', 'sqeuclidean', 'yule' according to the manual from scipy. In this case euclidean is chosen according to the instruction. Euclidean is distance matrix calculate from the cartesian coordinate or simply x,y axis using pythagorean theorem. It can be used for 2-dimension and 3-dimension up to n-dimension. The equation for calculation (2 dimensions) is

```
d(x,y) = sqrt((x1-x2)^2 + (y1-y2)^2)
```

2.3. Merging close cluster together

```
from scipy.cluster.hierarchy import linkage
link = linkage(dist,method='single',metric='euclidean')
```

After obtaining distance matrix, the next step is grouping close cluster together. Linkage function will handle both distance calculation and merging. The function will return a linkage matrix, which can be further used for visualizing as a dendrogram and for data points labeling. For this experiment the utilized linkage function is from the scipy library, same as the distance matrix. It can take both 1-D distance matrix or 2-D array of vector. Merging is done from bottom up with condition and initial state depending on chosen method. There are 4 different method that produce different results.

2.3.1. Single

Single or Nearest point algorithm. This method start from shortest distance between 2 data points first, they will be merged together forming a cluster, then the second shortest and so on. The number of cluster will rise along the way as new cluster will be form and some clusters will merge together. At the end, the number of cluster will be 1 and the last merge point will be consider as root node. The equation is as follow:

```
d(C1,C2) = min(dist(C1[i],C2[i]))
i as a point
C1 and C2 as clusters
```

2.3.2. Complete

Complete or Farthest point algorithm work similarly to Nearest point algorithm with difference being start from farthest and finish at the nearest point. At the end number of cluster is also 1 and the last data point that being merge will be a root node. The equation is as follow:

```
d(C1,C2) = max(dist(C1[i],C2[i]))
i as a point
C1 and C2 as clusters
```

2.3.3. Average

Average linkage is measured with average distance of every points from one cluster to every points of another cluster. The method is similar to Neartest point algorithm where the initial node is the closest distance and merging together. After merging, however, Average linkage will re calculate all distances between every data points in group to all other data points with average distance. The next merging node is also the next nearest data points in the set of data, and so on. Until reaching the root node as the number of clusters reaches 1. The equation is as follow

```
d(C1,C2) = sum(dist(C1[i],C2[j])/(|i|*|j|))
i and j as points
C1 and C2 as clusters
```

2.3.4. Ward

Lastly, Ward's linkage is measured by calculating the centroid of existing node as a point of distance measure. Initially, where every data point is not merged yet, the initial merging node starts the same way as Nearest point algorithm where the shortest distance between the node is the first one. Once the cluster is formed, the centroids of the cluster will be calculated and compares the deviation of the centroids to points in cluster, the smallest deviation will be chosen.

```
d(C1,C2) = sqrt(((|C2|+|C3|)/T *d(C2,C3)^2)+((|C2|+|C4|)/T *d(C2,C4)
    ^2)-(|C2|=T *d(C3,C4)^2))
C1 as new cluster
C2,C3,C4 as unused clusters
T as |C2|+|C3|+|C4|
```

2.3.5. Dendrogram

Once linkage matrix is completed, the dendrogram can be constructed with the linkage matrix. With the help of a function from Scipy and matplot. Interpreting the dendrogram: The horizontal lines represent the cluster at each stage, where vertical lines will point to children. When drawing the horizontal line across the vertical line at any height, the intersect line measure the number of cluster. Thus the higher the cross line, the lower number of cluster and vice versa. The height represents the distance between the immediate (now merged) children.

```
from scipy.cluster.hierarchy import dendrogram
import matplotlib.pyplot as plt
gram = dendrogram(link)
plt.show()
```

2.4. Labeling data points

Then, the labeling data points function, named fcluster from scipy, is the function to interpret linkage matrix to the labeling matrix. This label matrix combined with the data matrix will be used to plot the graph of clusters and data points. The label matrix will mark the data point according to the index in data matrix by the number. For example, in the data set with 3 clusters, there will be 3 different number in label matrix repeating. The function is taken the linkage matrix and number of desired cluster. The process is similar to drawing a line across the dendrogram. With specifying number of clusters, the algorithm interprets the linkage matrix to the label matrix. In short, The function works by merging 2 clusters with lowest distance together for the number of times equal to length of linkage matrix + 1 to the blank matrix with size of linkage matrix + 1.

```
from scipy.cluster.hierarchy import fcluster
label = fcluster(link,t=k)
```

2.5. Silhouette score

Silhouette score is a internal metric for determining the goodness of a clustering method. The score is calculated based on intra-cluster distance and Nearest cluster distance. The intra-cluster distance should be relatively lower while nearest cluster distance should be relatively higher for the score to be considered good. The score ranges from -1 to 1 and it is the average distance of all clusters. The arrays take into this function are the label and data array.

```
from sklearn.metrics import silhouette_score
silhouette_avg = silhouette_score(data_array, label)
```

3. Result

3.1. Dendrograms for different linkage types with cluster cutoff thresholds

Figure 1: *Dendrogram of single linkage with cluster cutoff thresholds*

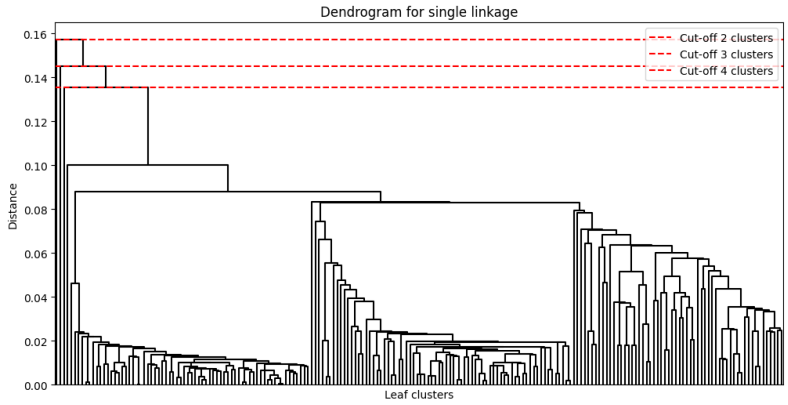


Figure 2: *Dendrogram of average linkage with cluster cutoff thresholds*

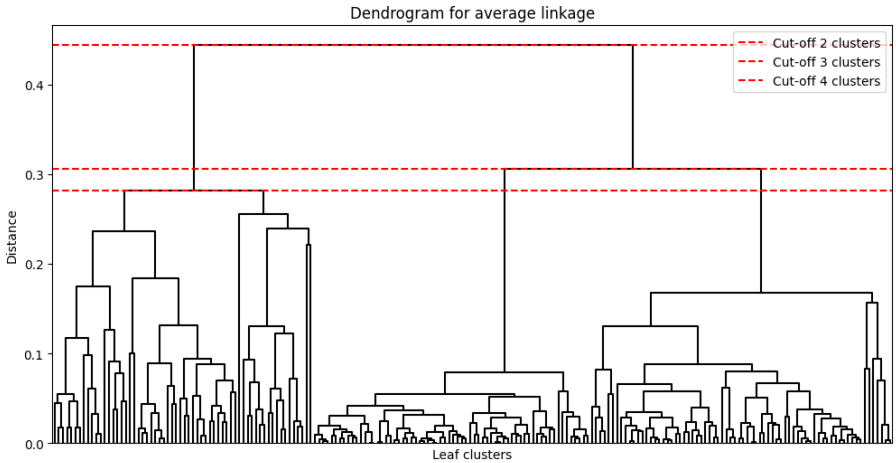


Figure 3: *Dendrogram of complete linkage with cluster cutoff thresholds*

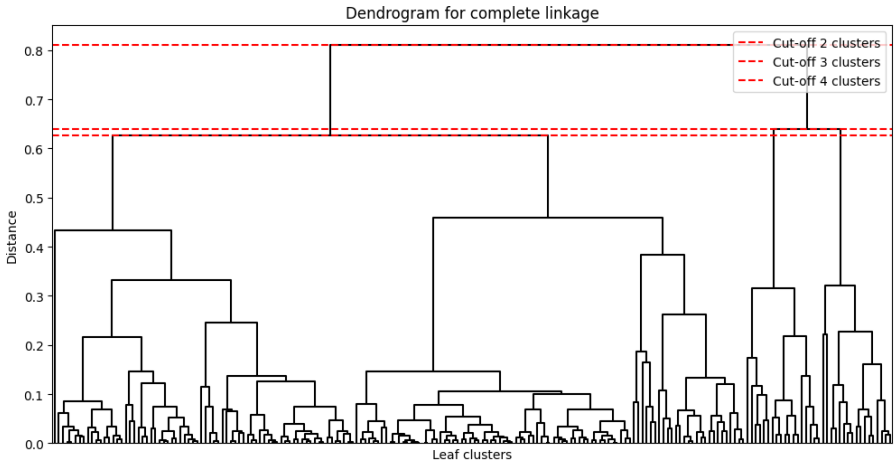
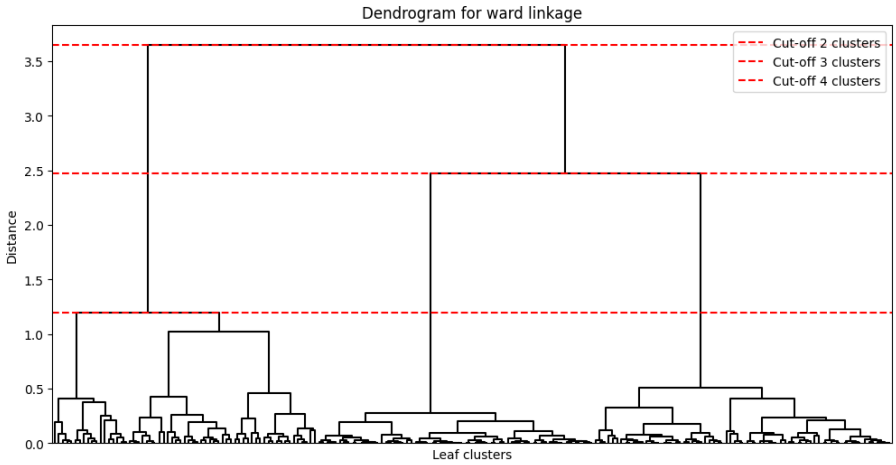


Figure 4: *Dendrogram of ward linkage with cluster cutoff thresholds*



3.2. Scatterplots of the original dataset and different cluster experiments

Figure 5: *2D Scatterplot of the original dataset*

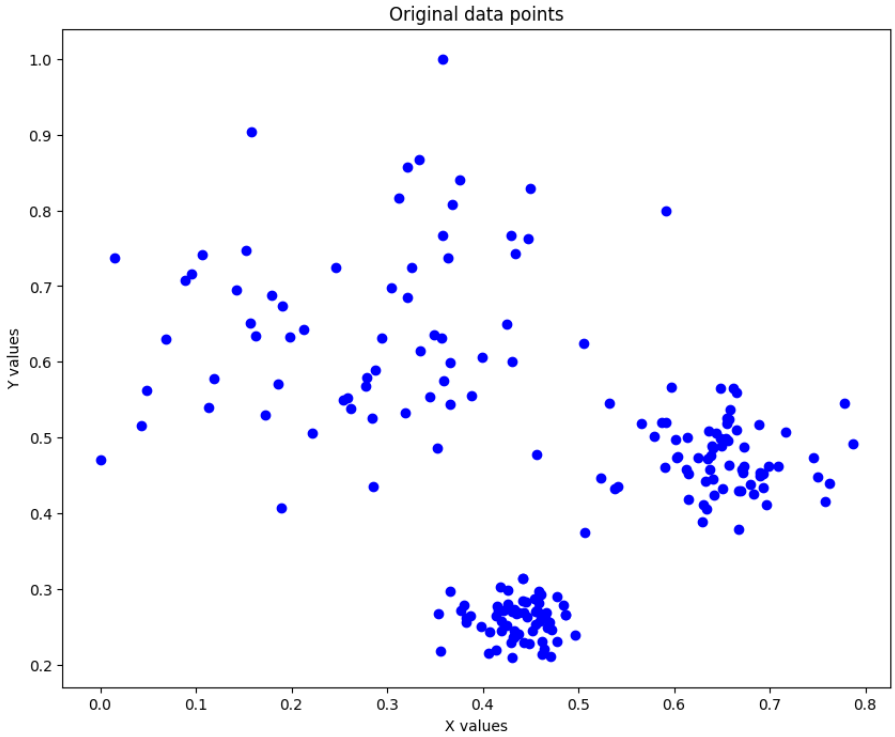


Figure 6: *2D Scatterplot of single linkage with 2 clusters*

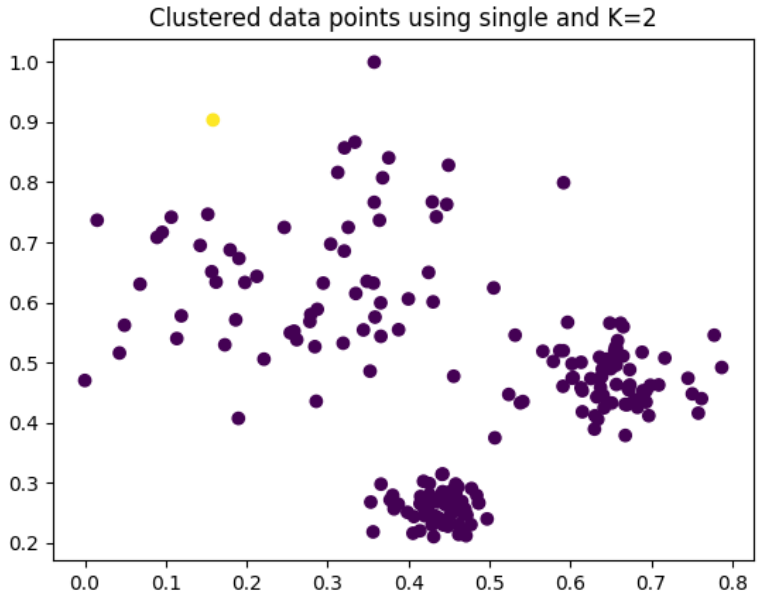


Figure 7: *2D Scatterplot of single linkage with 3 clusters*

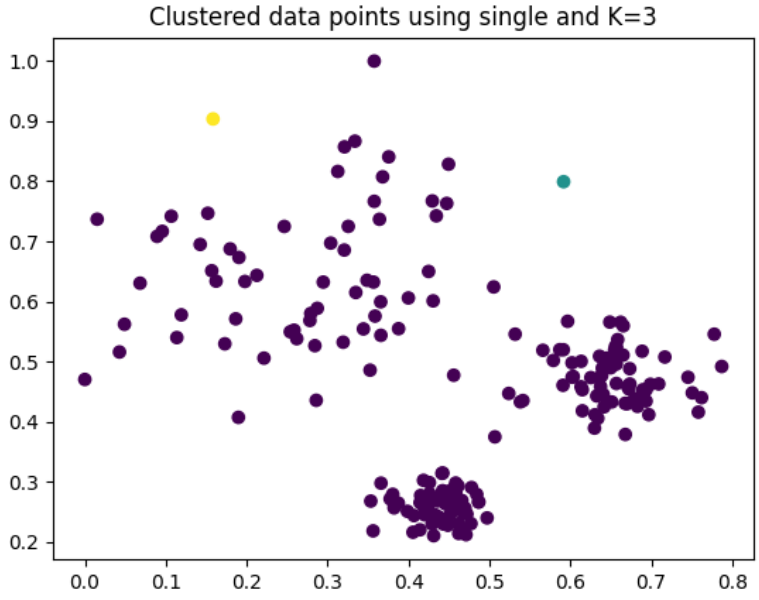


Figure 8: *2D Scatterplot of single linkage with 4 clusters*

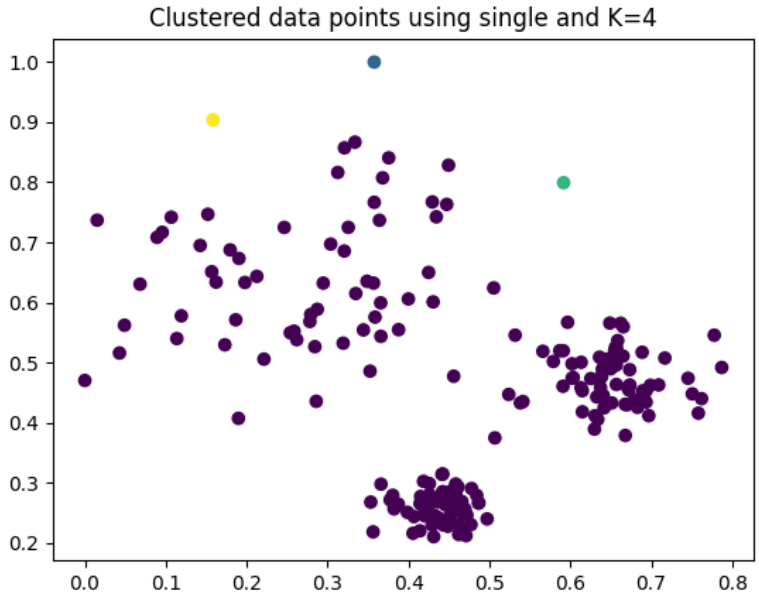


Figure 9: *2D Scatterplot of average linkage with 2 clusters*

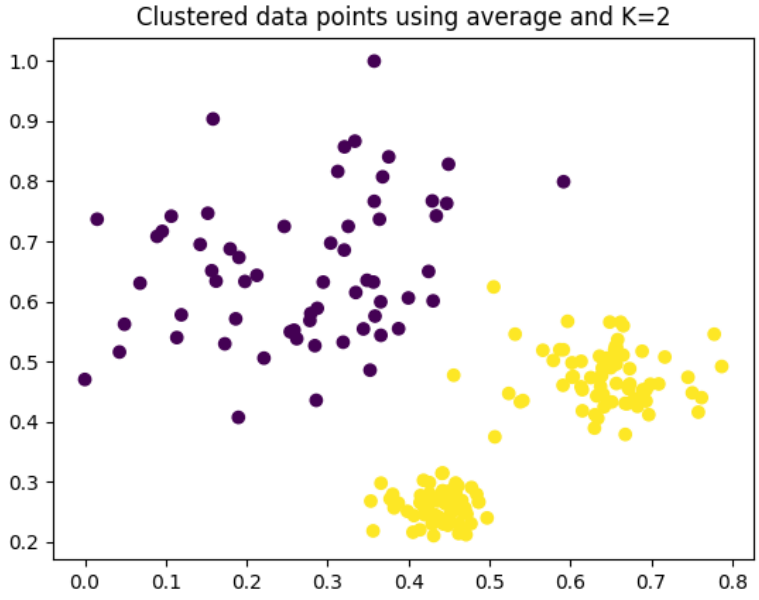


Figure 10: *2D Scatterplot of average linkage with 3 clusters*

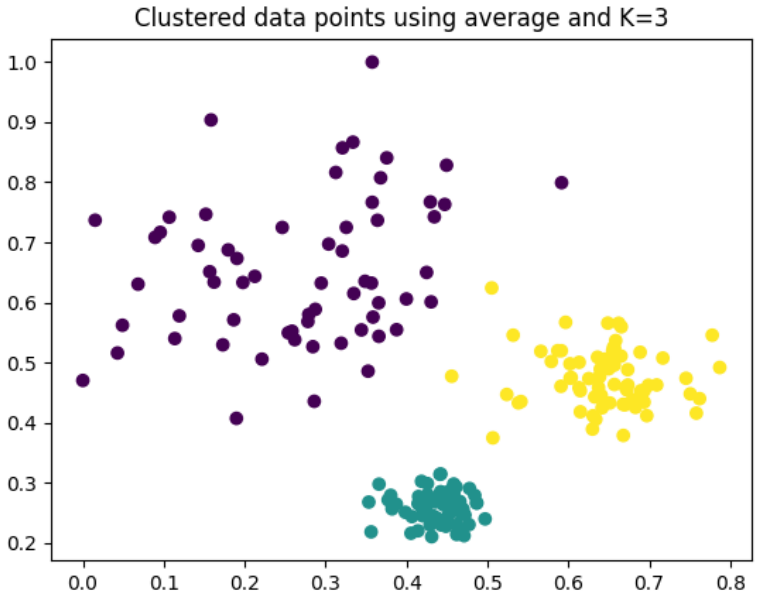


Figure 11: *2D Scatterplot of average linkage with 4 clusters*

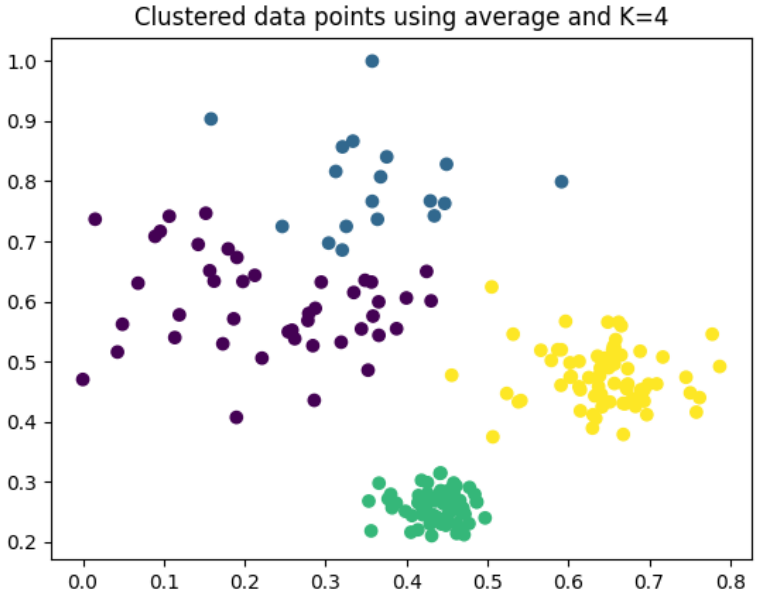


Figure 12: *2D Scatterplot of complete linkage with 2 clusters*

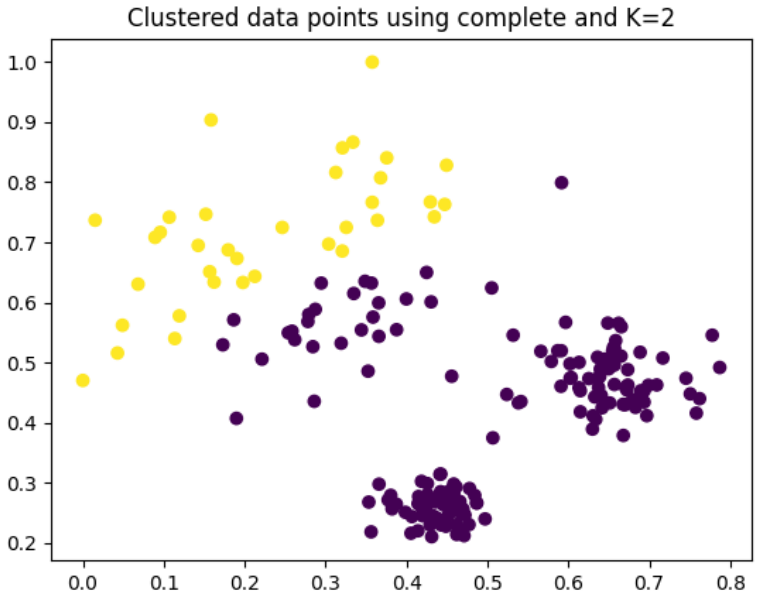


Figure 13: *2D Scatterplot of complete linkage with 3 clusters*

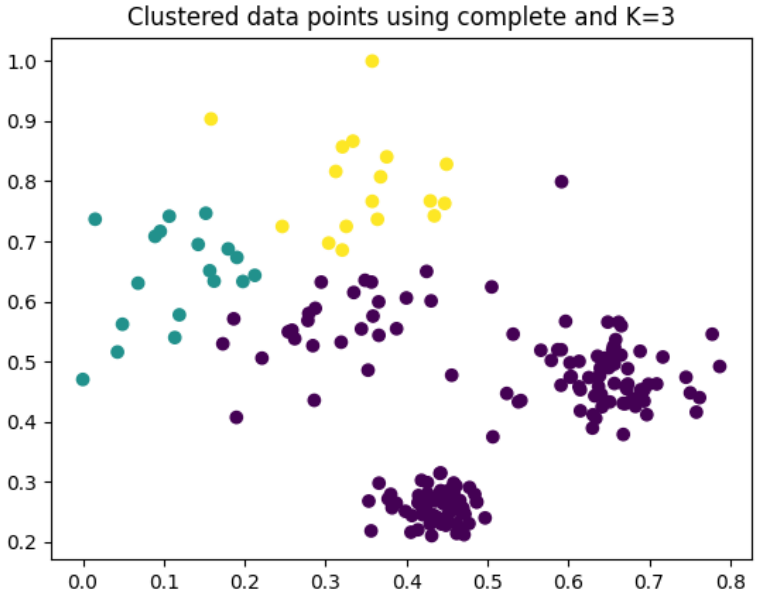


Figure 14: *2D Scatterplot of complete linkage with 4 clusters*

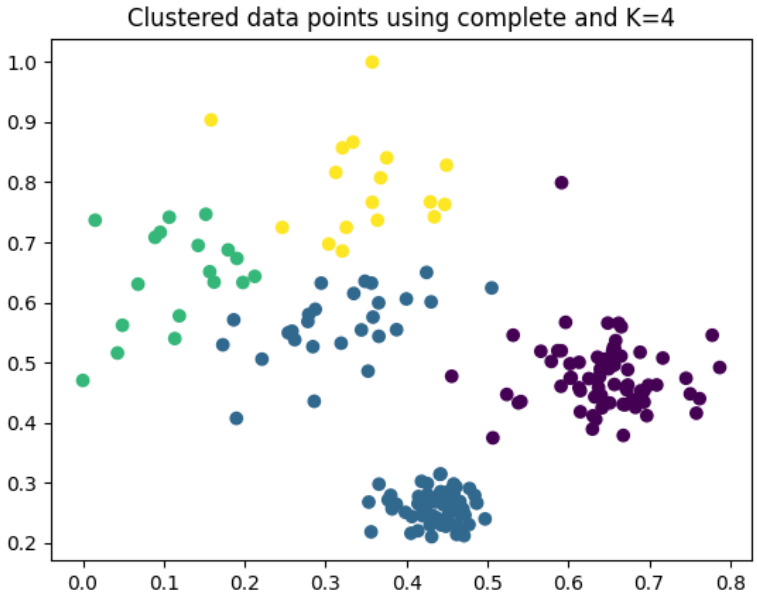


Figure 15: *2D Scatterplot of ward linkage with 2 clusters*

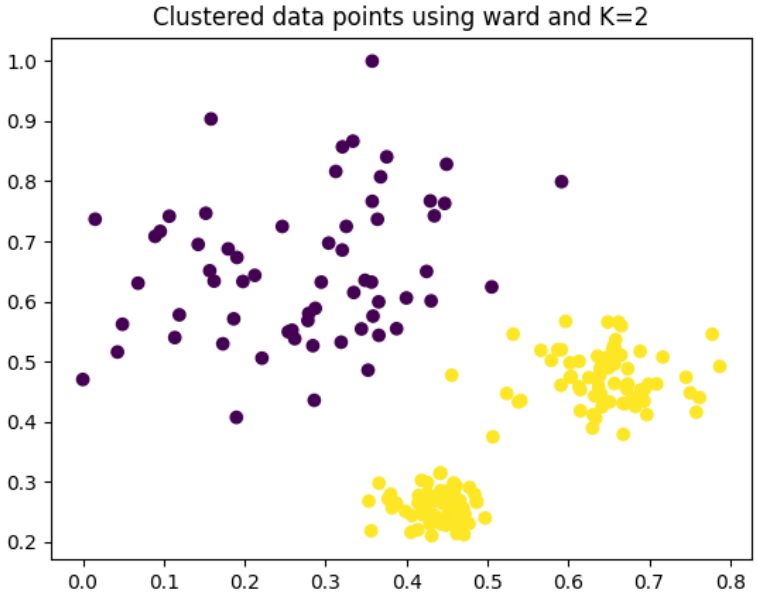


Figure 16: *2D Scatterplot of ward linkage with 3 clusters*

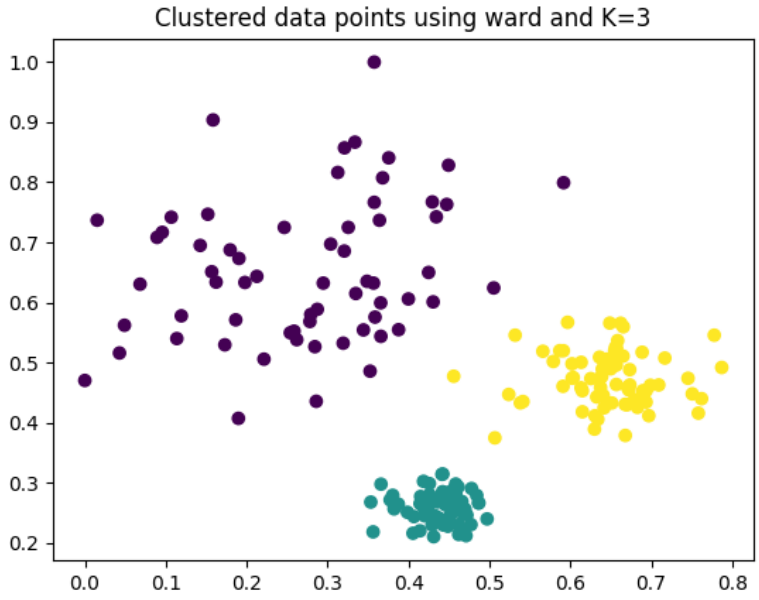
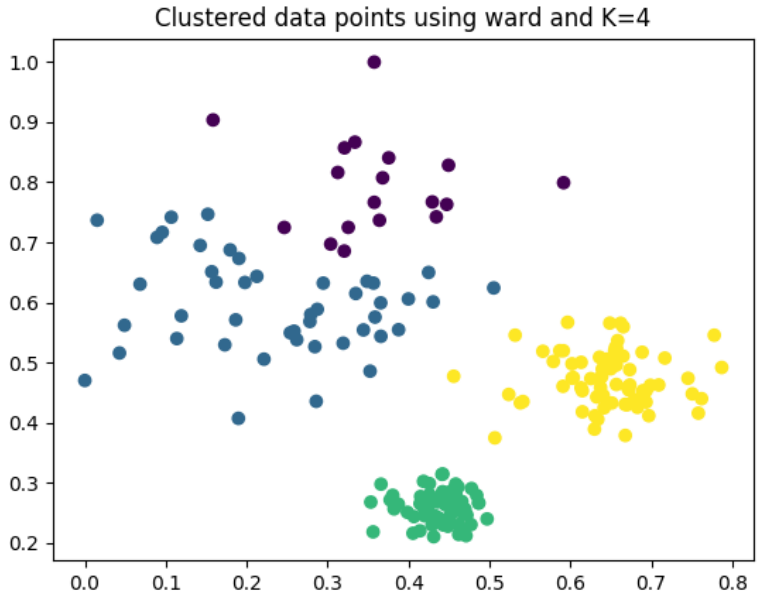


Figure 17: *2D Scatterplot of ward linkage with 4 clusters*



3.3. Silhouette scores

The following table shoe the silhouette score for each linkage type and number of Clusters.

Linkage Type	K-Clusters	Silhouette score
Single	2	0.3773
-	3	0.1804
-	4	0.1852
Average	2	0.5412
-	3	0.4207
-	4	0.4901
Complete	2	0.4654
-	3	0.4207
-	4	0.4901
Ward	2	0.5411
-	3	0.6509
-	4	0.6251

4. Discussion

For single linkage it is interesting to see that the clusters are heavily leaned towards the first linked clusters on the left. Upon further inspection we noticed the cause for this phenomenon in Figure 6 and Figure 7. As discussed in the lecture the single linkage method falls short when dealing with random data. When trying to cluster using the single distance the yellow and blue points in the mentioned graphs are the furthest apart from the actual cluster and point cloud. For such kind of data it did not seem applicable to use single linkage for relatively low K. On the other hand an increase in K could detect the actual clusters, but also introduce many smaller noise clusters aswell as seen in Figure 1. This also explains the lower silhouette score when compared to other methods like ward. From out observations the methods average and complete seem fairly similar in the clusters detected from a visual perspective as seen in Figure 11 and Figure 14. From these figures it can also be seen from a visual comparison that average did the better job. When looking at Figure 17 it produced similar results as the previously discussed methods. A clearer distinction is made from the lower (green) cluster and the blue noise, where the previous methods have included such points into the cluster. This makes the ward clusters the clearest from a visual point. This leads us to last observation being the silhouette scores. In these the ward method also ranked the best. Similar to the visual test both average and complete linkage scored similar scores.

A code-related observation was made when we were implementing the Agglomerative clustering algorithm. There is function for the whole algorithm, but it was not possible to plot the dendrogram directly. By using separated functions consisting of distance, linkage, and labeling, it is possible to obtain dendrogram and the plot of cluster data.

5. Contributions

For Thanakit's responsible are coding the method of distance, linkage, and labeling. The report are introduction and method.

Tobias did the user input and final visualization methods. This includes rendering of the dendrograms with correct cutoff thresholds.

We did both the work equally 50:50, and spent time revising the report and the code together and make changes to each other parts when needed.

References

[1] In: ().

[2] Google. “Clustering Algorithms”. In: (2022). URL: https://developers.google.com/machine-learning/clustering/clustering-algorithms#:~:text=Centroid%2Dbased%20clustering%20organizes%20the,to%20initial%20conditions%20and%20outliers..

[3] Shreya Joshi. “What is Clustering in Machine Learning: Types and Methods”. In: (2022). URL: https://www.analytixlabs.co.in/blog/types-of-clustering-algorithms/.