

Introduction to Machine Learning

Assignment 2

Group 31

Stijn Kammer (s4986296) & Ramon Kits (s5440769)

September 28, 2022

1. INTRODUCTION

In this report, results of the second assignment of the course Machine Learning will be discussed. The assignment was to implement the agglomerative hierarchical clustering algorithm. Different linkage methods were implemented and tested on the dataset delivered by the course. The results of the different linkage methods and finding the optimal number of clusters are discussed in this report.

2. METHODS

The agglomerative hierarchical clustering algorithm is an iterative algorithm which starts with each data point in its own cluster. The algorithm then iteratively merges the two closest clusters. The distance between two clusters is defined by the linkage method. The algorithm stops when all data points are in one cluster. After the algorithm has finished, the data points are clustered in a tree structure. The tree structure can be used to find the optimal number of clusters. Four linkage methods that may be used are single, complete, average and ward's linkage.

Single linkage uses the minimum distance between objects in two clusters. To determine the distance between two clusters, the minimum distance between all objects in the two clusters is used. This works well for clusters that are clearly separated. Complete linkage uses the maximum distance between objects in two clusters. To determine the distance between two clusters, the maximum distance between all objects in the two clusters is used, this is the opposite of single linkage. This can lead to clusters that are very spread out because clusters with closer objects may not be merged because it has objects that are further away. Both of the last two linkage methods are highly sensitive to outliers and noise. Average linkage uses the average distance between objects in two clusters. It uses the average distance between all objects in the two clusters to determine the distance between the two clusters. This makes it more robust to outliers. At last, ward's linkage is the most complex linkage method of the four. It uses the variance of the clusters to determine the distance between the clusters. The variance is the sum of the squared distances between the objects in the cluster and the mean of the cluster. When the increase in variance is at its minimum, the clusters will be merged. This linkage method is in many cases preferred over the other three because it is very robust to outliers and it is more sensitive to the shape of the clusters.

3. IMPLEMENTATION

For the data clustering the agglomerative hierarchical clustering algorithm was implemented. The algorithm was implemented in Python and the results were plotted using matplotlib. All four

linkage methods were implemented and tested on the dataset delivered by the course. Also the amount of clusters was determined using the dendrogram and the silhouette score.

Listing 1: *Plotting the dendrogram*

```
1 import numpy as np
2 import pandas as pd
3 import matplotlib.pyplot as plt
4 from scipy.cluster.hierarchy import dendrogram, linkage
5 from sklearn.cluster import AgglomerativeClustering
6 from sklearn.metrics import silhouette_score
7
8 AFFINITY = 'euclidean'
9 LINKAGES = ['single', 'complete', 'average', 'ward']
10
11 # Read data from csv file
12 data = pd.read_csv('data_clustering.csv', header=None)
13 data = data.values
14
15 for linkageMeasures in LINKAGES:
16     # create hierarchical clustering model
17     # generate and display dendrogram and color the clusters using the
18     # above set colors
19     plt.figure(figsize=(10, 7))
20     plt.title("Dendrogram for {} linkage".format(linkageMeasures))
21     plt.xlabel("Data points")
22     plt.ylabel("Euclidean distance")
23     dendrogram(linkage(data, method=linkageMeasures,
24                        metric=AFFINITY), no_labels=True, color_threshold=0)
```

In Listing 1 the most important part of the code is shown which plots the dendrogram. The code imports the data from the csv file and performs hierarchical clustering using the four linkage methods. The code then plots the dendrograms using the scipy library. This dendrogram can be used to determine the optimal number of clusters.

Listing 2: *Plotting the dendrogram*

```
1 def ownSilhouetteScore(data, labels):
2     # Calculate the silhouette score for the given data and labels
3
4     # Calculate intra-cluster distances
5     intraClusterDistances = np.zeros(len(data))
6     for i in range(len(data)):
7         # Calculate the average distance to all other points in the
8         # same cluster
9         intraClusterDistances[i] = np.mean(np.linalg.norm(
10             data[labels == labels[i]] - data[i], axis=1))
11
12     # Calculate inter-cluster distances
13     interClusterDistances = np.zeros(len(data))
14     for i in range(len(data)):
15         # Calculate the average distance to all other points in the
16         # nearest cluster
17         interClusterDistances[i] = np.min(
```

```

16         np.mean(np.linalg.norm(data[labels != labels[i]] - data[i],
17                               axis=1)))
18
19     # Calculate the silhouette score
20     silhouetteScore = np.mean((interClusterDistances -
21                               intraClusterDistances) /
22                               np.maximum(intraClusterDistances,
23                                           interClusterDistances))
24
25     return silhouetteScore
26
27 # Read data from csv file
28 data = pd.read_csv('data_clustering.csv', header=None)
29 data = data.values
30
31 # calculate silhouette score for different configurations of the model
32 results = {'single': [], 'complete': [], 'average': [], 'ward': []}
33 nclustersList = range(MINCLUSTERS, MAXCLUSTERS + 1)
34 for nClusters in nclustersList:
35     for linkage in results.keys():
36         hac = AgglomerativeClustering(
37             n_clusters=nClusters, affinity=AFFINITY, linkage=linkage)
38         hac.fit(data)
39         results[linkage].append(silhouette_score(data, hac.labels_, ))
40
41 # display results in a bar chart
42 x = np.arange(len(nclustersList))
43 width = 0.20
44 fig, ax = plt.subplots()
45 plt.ylim(top=1)
46 rects1 = ax.bar(x - 2*width, results['single'], width, label='Single')
47 rects2 = ax.bar(x - width, results['complete'], width, label='Complete')
48
49 rects3 = ax.bar(x, results['average'], width, label='Average')
50 rects4 = ax.bar(x + width, results['ward'], width, label='Ward')
51
52 for rect in rects1 + rects2 + rects3 + rects4:
53     height = rect.get_height()
54     ax.annotate(round(height, 2),
55                 xy=(rect.get_x() + rect.get_width() / 2, height),
56                 textcoords="offset points",
57                 ha='center', va='bottom', rotation=22.5)
58
59 ax.set_ylabel('Silhouette score')
60 ax.set_xlabel('Number of clusters')
61 ax.set_title('Silhouette score for different configurations')
62 ax.set_xticks(x)
63 ax.set_xticklabels(nclustersList)
64 ax.legend()
65 fig.tight_layout()
66 plt.show()

```

In Listing 2 is the code shown which uses agglomerative hierarchical clustering using the built-in

function of sklearn and calculates the silhouette scores. This code makes it possible to use either the silhouette calculation of the sklearn library as well as the possibility to calculate the silhouette score manually which uses the following formula:

$$S = \frac{1}{n} \sum_{i=1}^n \frac{(b_i - a_i)}{\max(a_i, b_i)}$$

Where a_i is the average distance between the object and all other objects in the same cluster. b_i is the average distance between the object and all other objects in the next nearest cluster. And n is the number of objects in the dataset.

The avrage scores are calculated for the four linkage methods and a range of possible number of clusters. After the scores are calculated, the code plots the scores using matplotlib in a bar chart. This chart makes it easy to find the optimal number of clusters.

Listing 3: *Plotting the clusters for different linkage methods*

```

1 import numpy as np
2 import pandas as pd
3 import matplotlib.pyplot as plt
4 from scipy.cluster.hierarchy import dendrogram, linkage
5 from sklearn.cluster import AgglomerativeClustering
6 from sklearn.metrics import silhouette_score
7
8 AFFINITY = 'euclidean'
9 LINKAGES = ['single', 'complete', 'average', 'ward']
10 MINCLUSTERS = 2
11 MAXCLUSTERS = 4
12
13 # Read data from csv file
14 data = pd.read_csv('data_clustering.csv', header=None)
15 data = data.values
16
17 # display data in scatter plot
18 plt.figure(figsize=(10, 7))
19 plt.title("Data without clustering")
20 plt.scatter(data[:, 0], data[:, 1], c='black', s=7)
21 plt.savefig('output/data_scatter.png')
22
23
24 def calculateWSS(data, labels):
25     # Calculate the WSS for the given data and labels
26
27     # calculate the centroids
28     centroids = np.zeros((len(np.unique(labels)), len(data[0])))
29     for i in range(len(np.unique(labels))):
30         centroids[i] = np.mean(data[labels == i], axis=0)
31
32     # calculate the cluster sum of squares
33     wss = 0
34     for i in range(len(data)):
35         wss += np.sum((data[i] - centroids[labels[i]]) ** 2)
36
37     return wss

```

```

38
39
40 def calculateBSS(data, labels):
41     # Calculate the BSS for the given data and labels
42
43     # calculate the centroids
44     centroids = np.zeros((len(np.unique(labels)), len(data[0])))
45     for i in range(len(np.unique(labels))):
46         centroids[i] = np.mean(data[labels == i], axis=0)
47
48     # calculate the cluster sum of squares
49     bss = 0
50     for i in range(len(centroids)):
51         bss += np.sum((centroids[i] - np.mean(centroids)) ** 2)
52
53     return bss
54
55
56 for linkageMeasure in LINKAGES:
57     for nClusters in range(MINCLUSTERS, MAXCLUSTERS + 1):
58         hac = AgglomerativeClustering(
59             n_clusters=nClusters, affinity=AFFINITY, linkage=
linkageMeasure)
60         # fit the model
61         hac.fit(data)
62         # display clusters in scatter plot
63         plt.figure(figsize=(10, 7))
64         plt.title("Scatter plot for {} linkage and {} clusters".format
(
65             linkageMeasure, nClusters))
66         plt.xlabel("x1")
67         plt.ylabel("x2")
68         plt.scatter(data[:, 0], data[:, 1], c=hac.labels_, cmap='
rainbow')
69         print("{} WSS: {} with {} clusters".format(linkageMeasure,
70             round(calculateWSS(data, hac.labels_), 2), nClusters))
71         print("{} BSS: {} with {} clusters".format(linkageMeasure,
72             round(calculateBSS(data, hac.labels_), 2), nClusters))
73
74         # save the plot
75         plt.savefig(
76             'output/{_}_{_clusters.png}'.format(linkageMeasure,
nClusters))

```

In Listing 3 the code is shown which uses agglomerative hierarchical clustering with the four linkage methods and a range of number of clusters and plots the clusters using matplotlib. This way it is possible to see how the clusters are formed by the different linkage methods.

What the code also is able to do is manually calculating the Within Cluster Sum of Squares

$$WSS = \sum_{i=1}^{N_C} \sum_{x \in C_i} d(X, \bar{X}_{C_i})^2$$

which represents the sum of the squared distances between the objects within the cluster. And the Between Cluster Sum of Squares

$$BSS = \sum_{i=1}^{N_C} |C_i| \cdot d(\bar{X}_{C_i}, X)^2$$

Which represents the sum of the squared distances between the mean of the clusters and the objects in the dataset. This applied to the different linkage methods and range of number of clusters. These values are being printed in the console so they can be analyzed for each combination of linkage method and number of clusters.

4. RESULTS

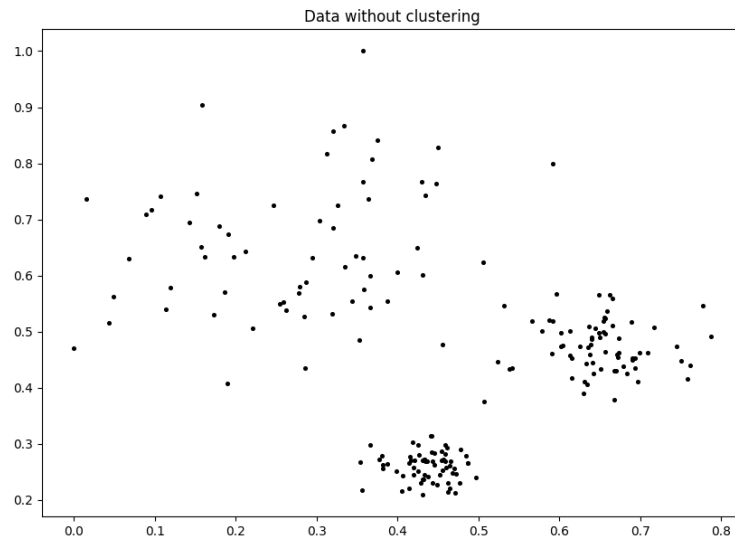


Figure 1: *Data scatter plot*

the data scatter plot is shown in Figure 1. The data is not separated in clusters.

4.1. DENDROGRAMS SINGLE LINKAGE

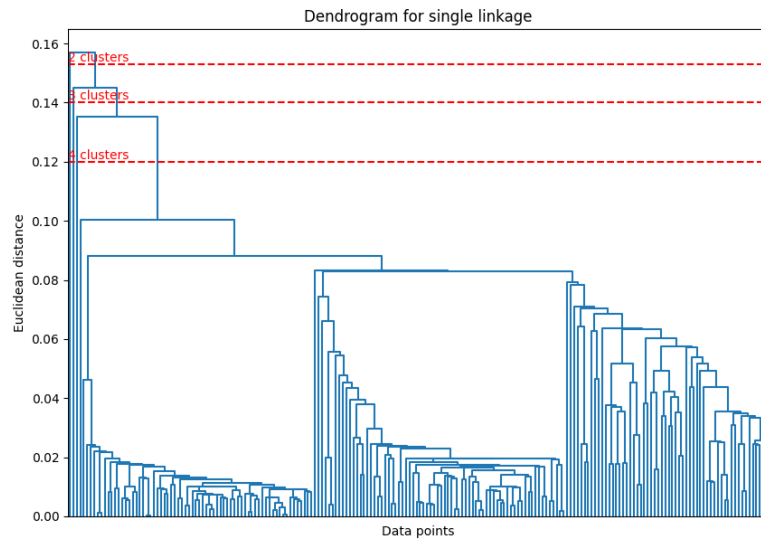


Figure 2: *Dendrogram for single linkage*

The dendrogram for single linkage is shown in Figure 2. The optimal number of clusters is 3 because the distance between the clusters is the smallest at this point. Single linkage is very sensitive to outliers and noise and this is visible in the dendrogram. This causes outliers and noise to be in their own cluster, making it hard to get a good result.

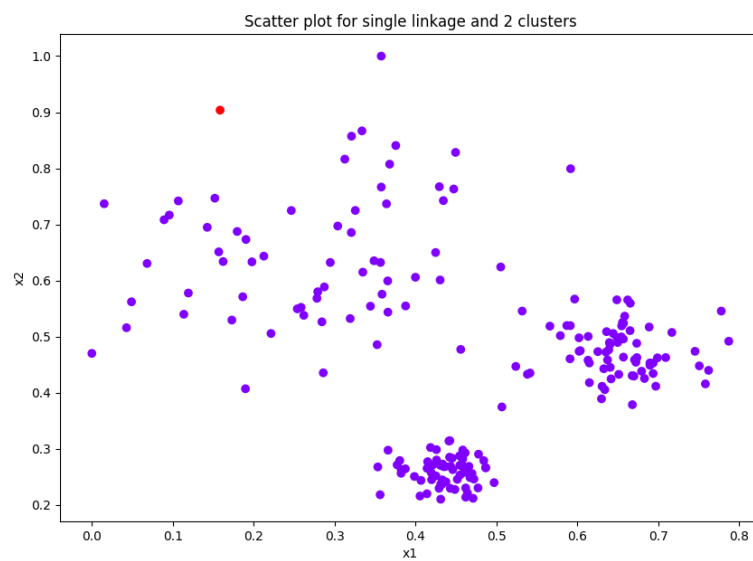


Figure 3: *Single 2 clusters*

The clusters for single linkage with 2 clusters are shown in Figure 3. One point is selected for a cluster and the other points are selected for the other cluster. This is because the distance between the points is the smallest for this combination.

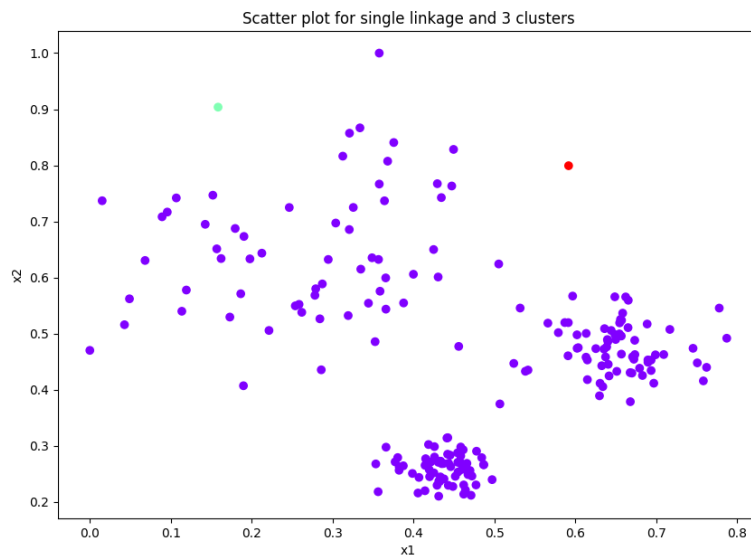


Figure 4: *Single 3 clusters*

The clusters for single linkage with 3 clusters are shown in Figure 4. Here two clusters have only a single point and the third cluster has the other points. This is because the distance between the points is the smallest for this combination.

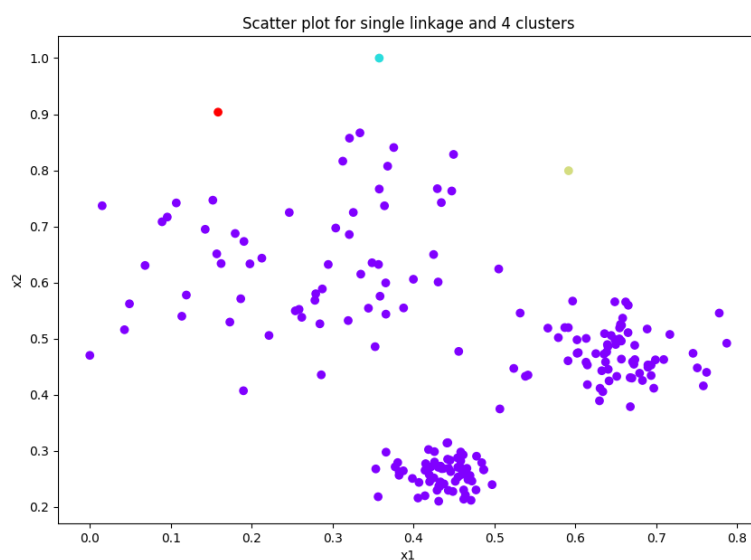


Figure 5: *Single 4 clusters*

The clusters for single linkage with 4 clusters are shown in Figure 5. Here three clusters have only a single point and the fourth cluster has the other points. This is because the distance between the points is the smallest for this combination.

4.2. DENDROGRAMS COMPLETE LINKAGE

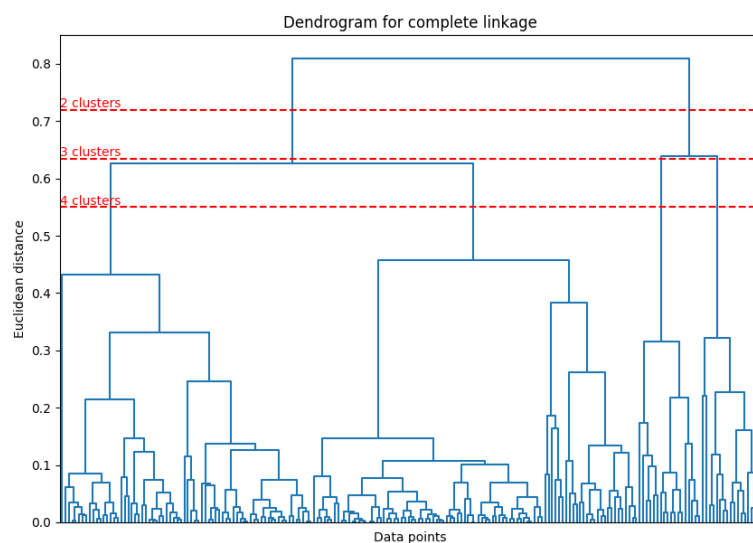


Figure 6: *Dendrogram for complete linkage*

The dendrogram for complete linkage is shown in Figure 6. The optimal number of clusters is 3 because the distance between the clusters is the smallest at this point. The clusters are clearly separated and the clusters are not very spread out. The clusters are more spread out than the clusters for single linkage.

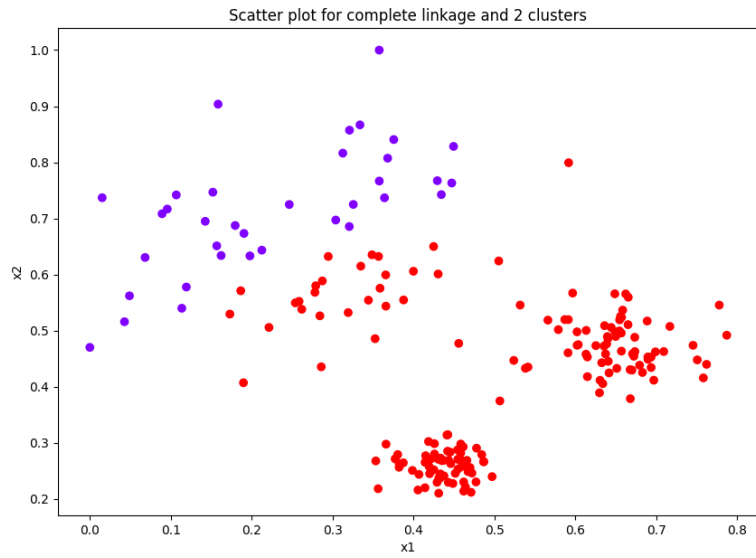


Figure 7: *Complete 2 clusters*

The clusters for complete linkage with 2 clusters are shown in Figure 7. Here the points are clearly split in two clusters. This is because the distance between the points is the smallest for this combination.

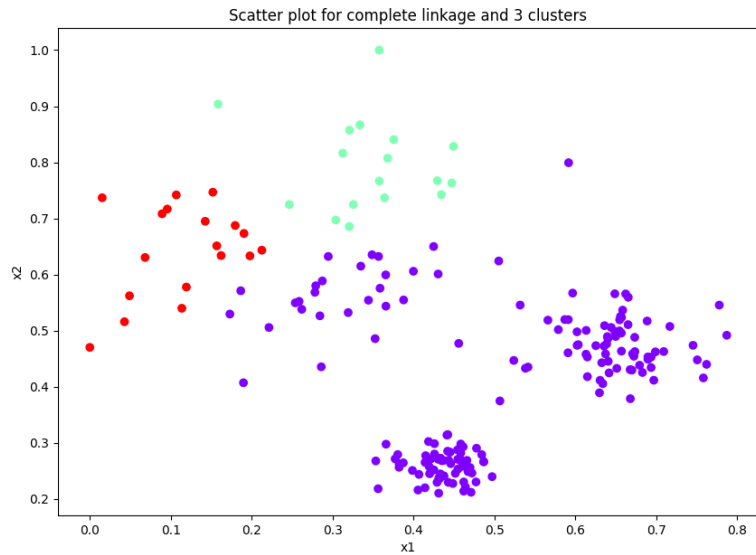


Figure 8: *Complete 3 clusters*

The clusters for complete linkage with 3 clusters are shown in Figure 8. Here the points are clearly split in three clusters. This is because the distance between the points is the smallest for this combination.

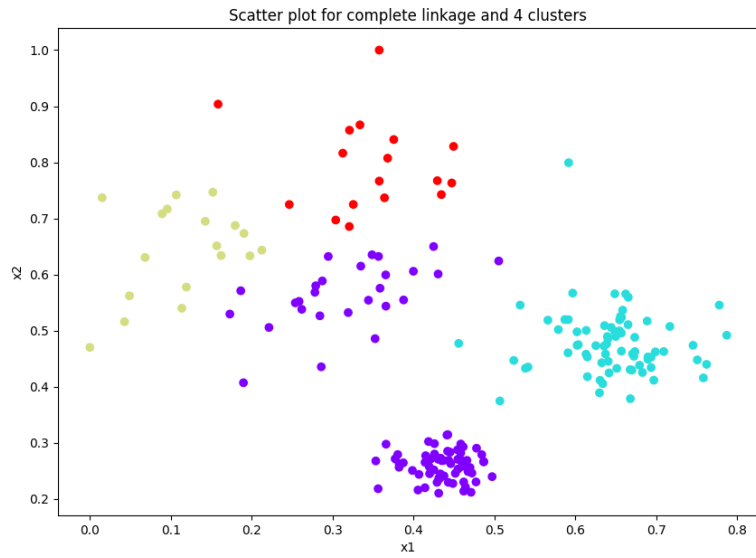


Figure 9: Complete 4 clusters

The clusters for complete linkage with 4 clusters are shown in Figure 9. Here the points are clearly split in four clusters. This is because the distance between the points is the smallest for this combination.

4.3. DENDROGRAMS AVERAGE LINKAGE

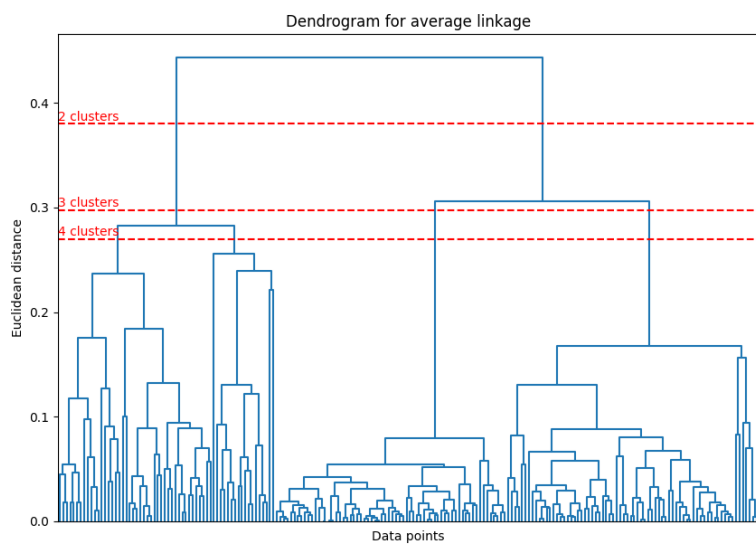


Figure 10: Dendrogram for average linkage

The dendrogram for average linkage is shown in Figure 10. The optimal number of clusters is 3 because the distance between the clusters is the smallest at this point. The clusters are clearly separated and the clusters are not very spread out. The clusters are more spread out than the clusters for complete linkage.

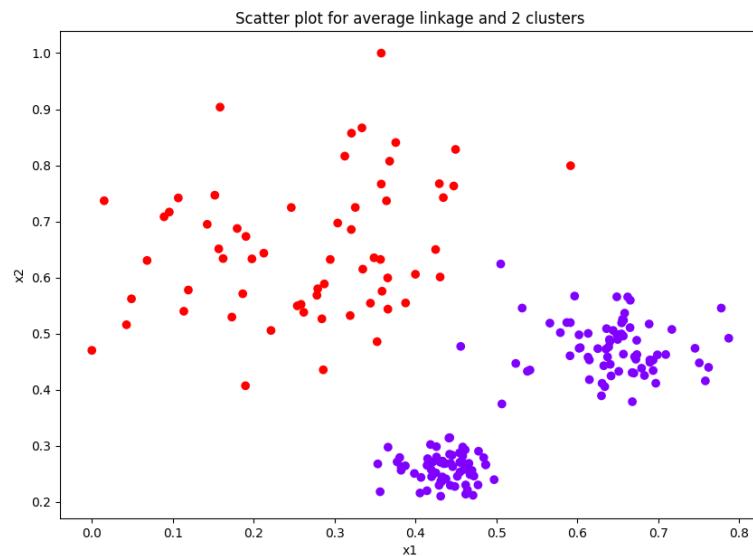


Figure 11: Average 2 clusters

The clusters for average linkage with 2 clusters are shown in Figure 11. Here the points are clearly split in two clusters. This is because the distance between the points is the smallest for this combination using average linkage.

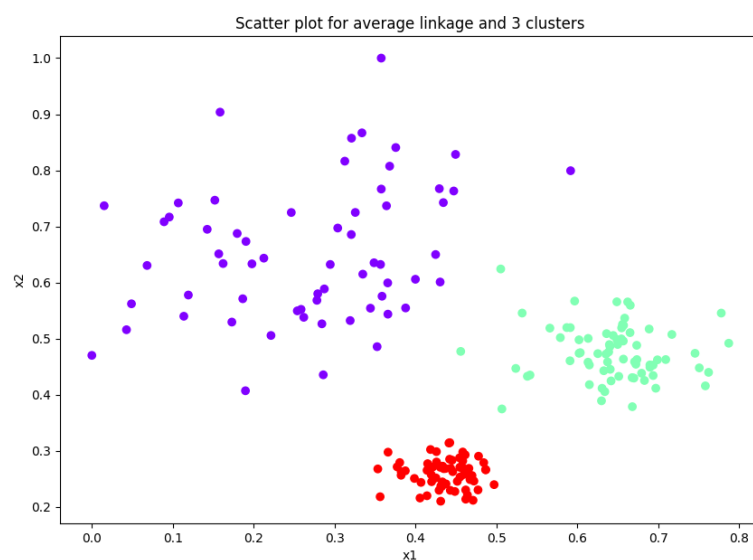


Figure 12: Average 3 clusters

The clusters for average linkage with 3 clusters are shown in Figure 12. Here the points are clearly split in three clusters. This is because the distance between the points is the smallest for this combination using average linkage.

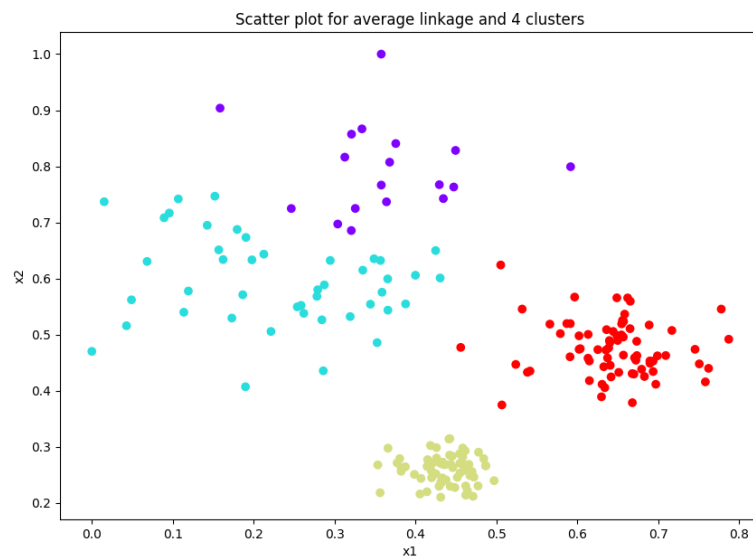


Figure 13: Average 4 clusters

The clusters for average linkage with 4 clusters are shown in Figure 13. Here the points are clearly split in four clusters. This is because the distance between the points is the smallest for this combination using average linkage.

4.4. DENDROGRAMS WARD LINKAGE

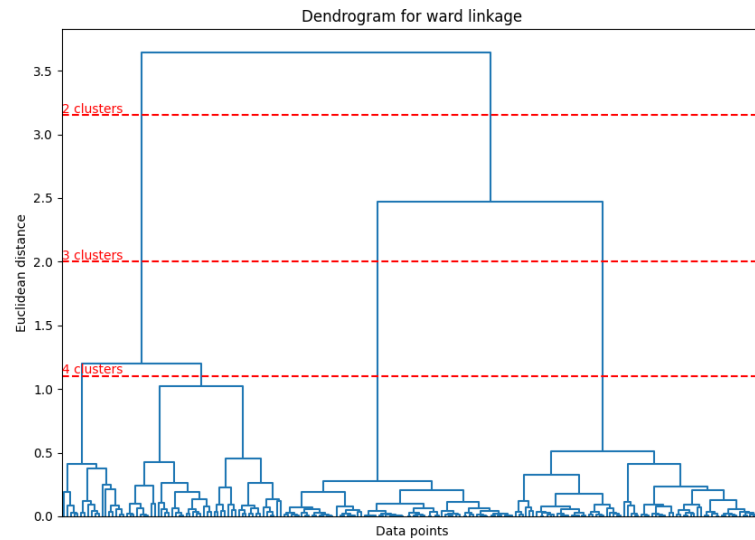


Figure 14: *Dendrogram for ward's linkage*

The dendrogram for ward's linkage is shown in Figure 14. The optimal number of clusters is 3 because the distance between the clusters is the smallest at this point. The clusters are clearly separated and the clusters are not very spread out. The clusters are more spread out than the clusters for average linkage.

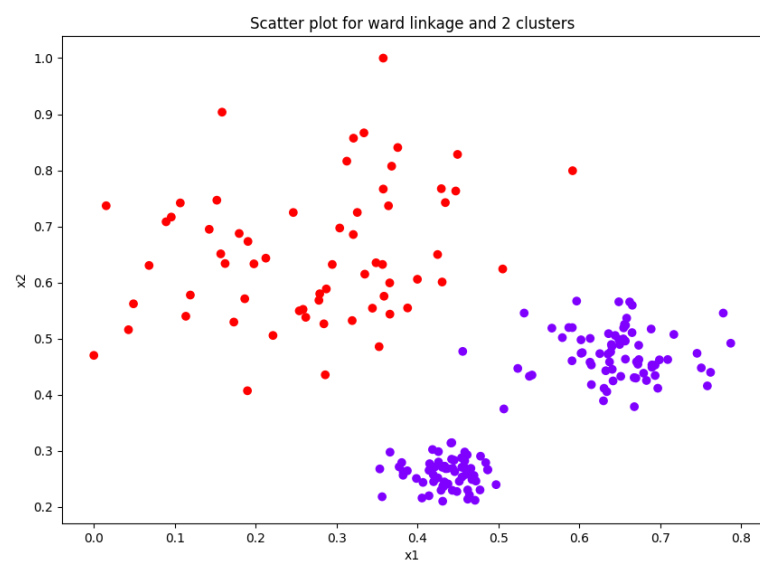


Figure 15: *Ward 2 clusters*

The clusters for ward's linkage with 2 clusters are shown in Figure 15. Here the points are clearly split in two clusters. This is because the distance between the points is the smallest for this combination using ward's linkage.

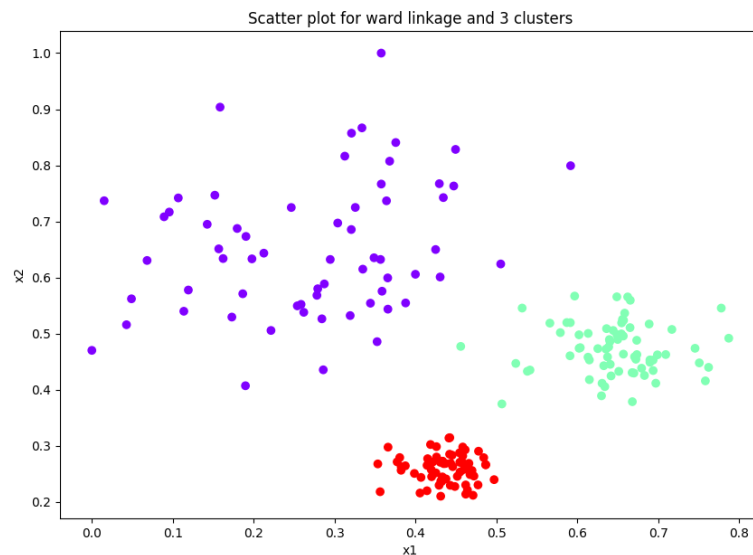


Figure 16: *Ward 3 clusters*

The clusters for ward's linkage with 3 clusters are shown in Figure 16. Here the points are clearly split in three clusters. This is because the distance between the points is the smallest for this combination using ward's linkage.

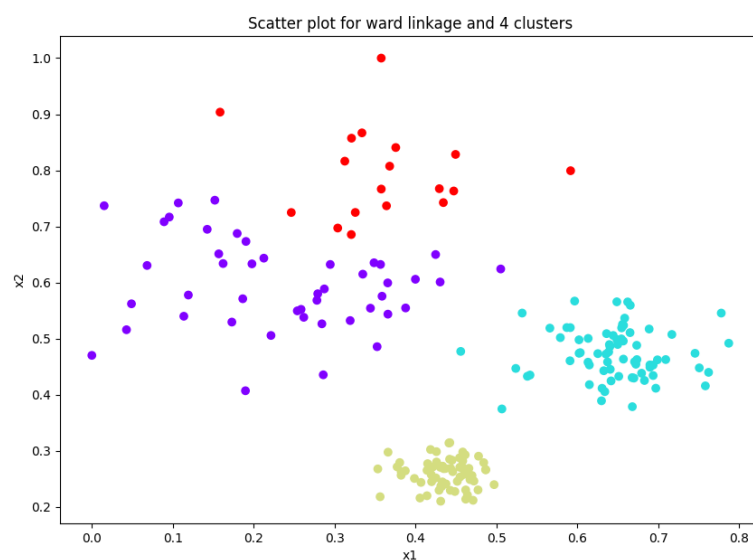


Figure 17: *Ward 4 clusters*

The clusters for ward's linkage with 4 clusters are shown in Figure 17. Here the points are clearly split in four clusters. This is because the distance between the points is the smallest for this combination using ward's linkage.

5. DISCUSSION

The dendrograms for the different linkages are shown in Figure 2, Figure 6, Figure 10 and Figure 14. The optimal number of clusters is 3 for all the linkages. The single linkage delivers the worst results because the clusters are very spread out and the splits are higher in the dendrogram. In dendrograms you want the split to be as close as possible to the bottom of the dendrogram. The ward linkage delivers the best results because the clusters are not very spread out and the splits are lower in the dendrogram. The average linkage delivers the second best results because the clusters are not very spread out and the splits are lower in the dendrogram than the single linkage.

6. WORK DISTRIBUTION

This week the work has been divided relatively equally between the group members. Ramon has been working a bit more on the report because he has been working on the code for the dendrogram a bit more. While Stijn has spend more time on analysing and writing the report. Causing both of us to gain the same amount of knowledge.