Student Name: Ben Beissner
Student ID: I6336899

Data Mining Lab 04 - Clustering

Analytical Report - 16.10.2023

Assignment 1

a) In this first part of the assignment we are going to investigate the influence of different values for the number of clusters in the k-means algorithm. We are going to generate 4 random clusters of data points according to a Gaussian distribution with standard deviation of 0.6. We are going to generate a total number of 300 datapoints.

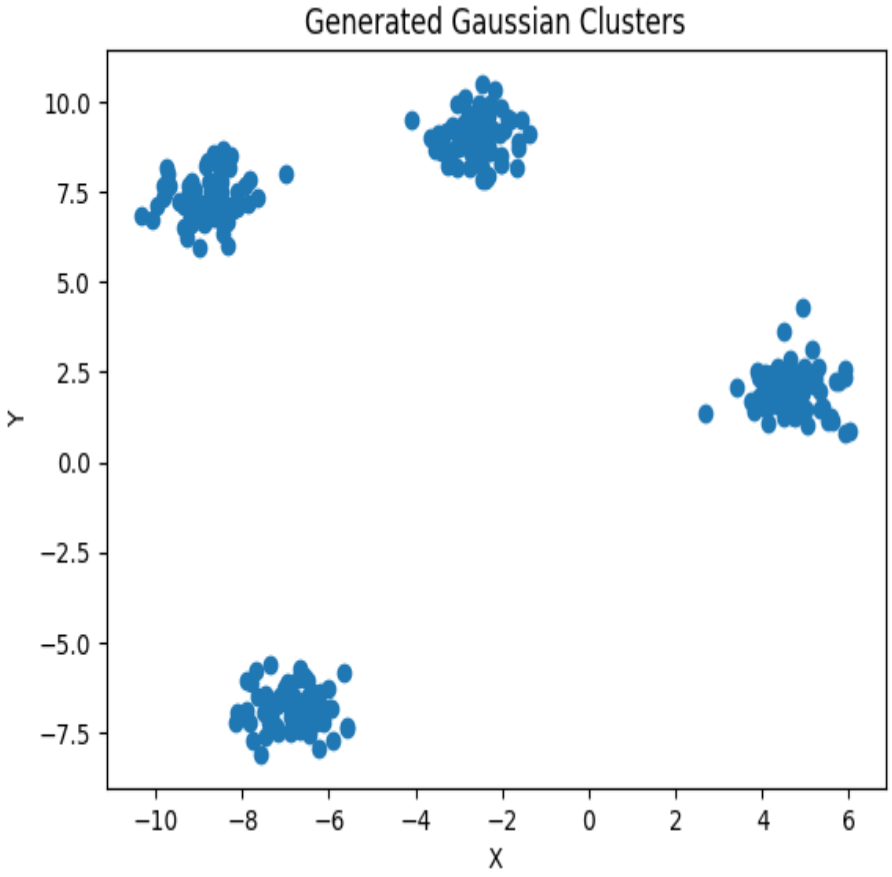

Figure 1: Clusters of initial data generation

b) After generating the data, the first task is to run the k-means algorithm for different values of k (1-10). We will check on the visualizations of the clusters and their centers and analyse the contingency tables. By looking at the plots for the calculated clusters of the k-means algorithm, we can observe the accuracy of assigned datapoints to a cluster. Since the parameter k is set to decide in how many clusters the datapoints are going to be divided, for k = 1 all datapoints are classified to the same cluster which does not reflect the distribution of datapoints into 4 clusters as seen above. When increasing k, more cluster assignments seem to be reasonable, with a correct classification of the datapoints for k = 4. When further increasing the value for k, we can observe that the initial 4 clusters are split into further sub-clusters. This

can be reasonable for more complex data with more true clusters but for this data generation k = 4 seems to be the optimal value for the parameter k because it enables the highest inter-class distance and lowest intra-class distance. By looking at the contingency tables (can be found in the code of the notebook), we can see how many of the 300 datapoints are assigned to which cluster for the different k-values. We observe a perfectly equal separation of 75 daatapoints for each cluster at k = 4, which further justifies 4 as the ideal value for k.

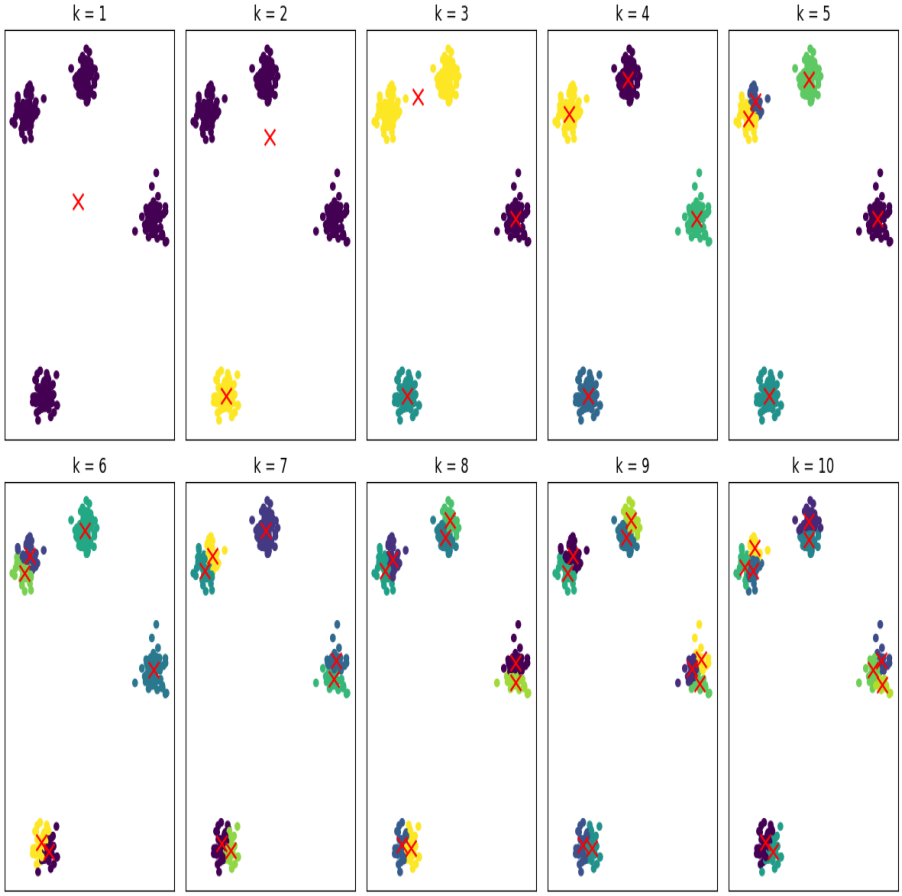

Figure 2: k-means clusters for k = [1-10]

c) To further validate the optimal value for parameter k, we are going to visualize the sum of square errors (SSE) for each of the k-values. SSE measures the total distance of all datapoints to their assigned cluster centre. Therefore, as more clusters are added (higher k-value) the SSE will decrease. Since, as seen above, adding more cluster centres does not always to find the most reasonable cluster separation, a trade-off between error and number of clusters is needed. The point where the SSE curve starts to bend (often called elbow point) is often used to determine the sweetspot of this trade-off. In the plot of the SSE curve we can see that this point is at k = 4, since for larger values for k the error does not decrease significantly anymore. This helps to justify 4 as the natural number of clusters.

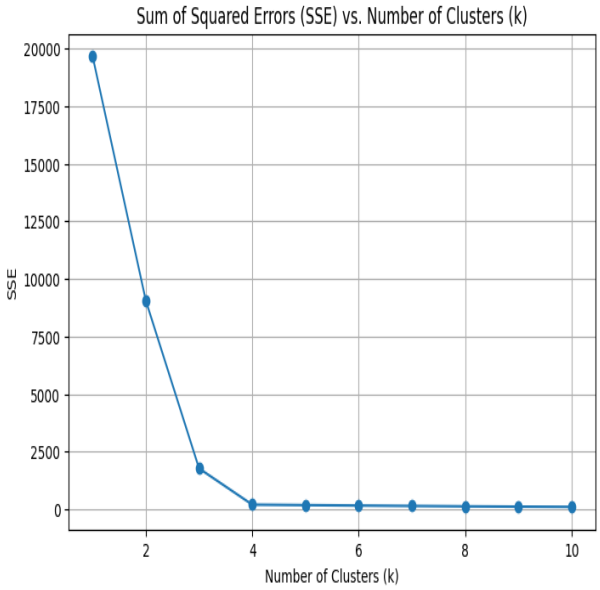

Figure 3: SSE vs. number of cluster

d) To further evaluate our findings about the behaviour of the k-means algorithm we will repeat the steps above with newly generated data. This time we will generated two datasets with a standard deviation of 0.1 and 2.5 respectively. The newly generated differs, according to the different standard deviations, in the density of the clusters, which can be seen in the following plots:

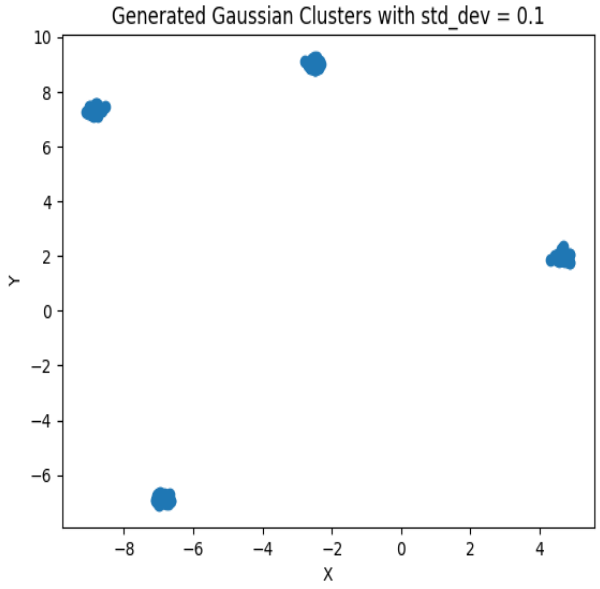

Figure 4: Clusters of data generation with standard deviation = 0.1

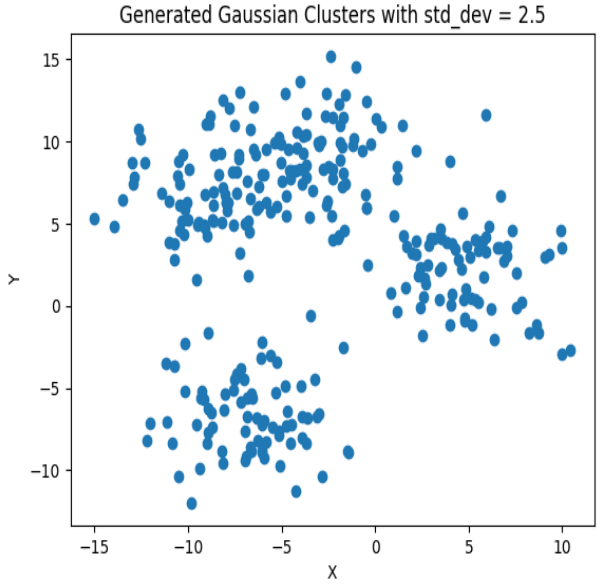

Figure 5: Clusters of data generation with standard deviation = 2.5

The visualizations of the different cluster assignments by the k-means algorithm for the dataset with standard deviation of 0.1 provide similar findings than with the initial data generation. Because the true clusters are even more dense than in the initial generation, we can say again that the optimal value for parameter k is 4 in this case. In is worth mentioning that because of the dense cluster the SSE value for k = 1 is higher than before, because the distance of the cluster centres increases slightly. For k-values higher than 4 we see a similar behaviour as before with the difference that the split of the more dense clusters in sub-clusters seems even more unreasonable. The SSE plot shows again that the elbow point is at k = 4 which indicates that the natural number of clusters is 4. The plots can be seen below:

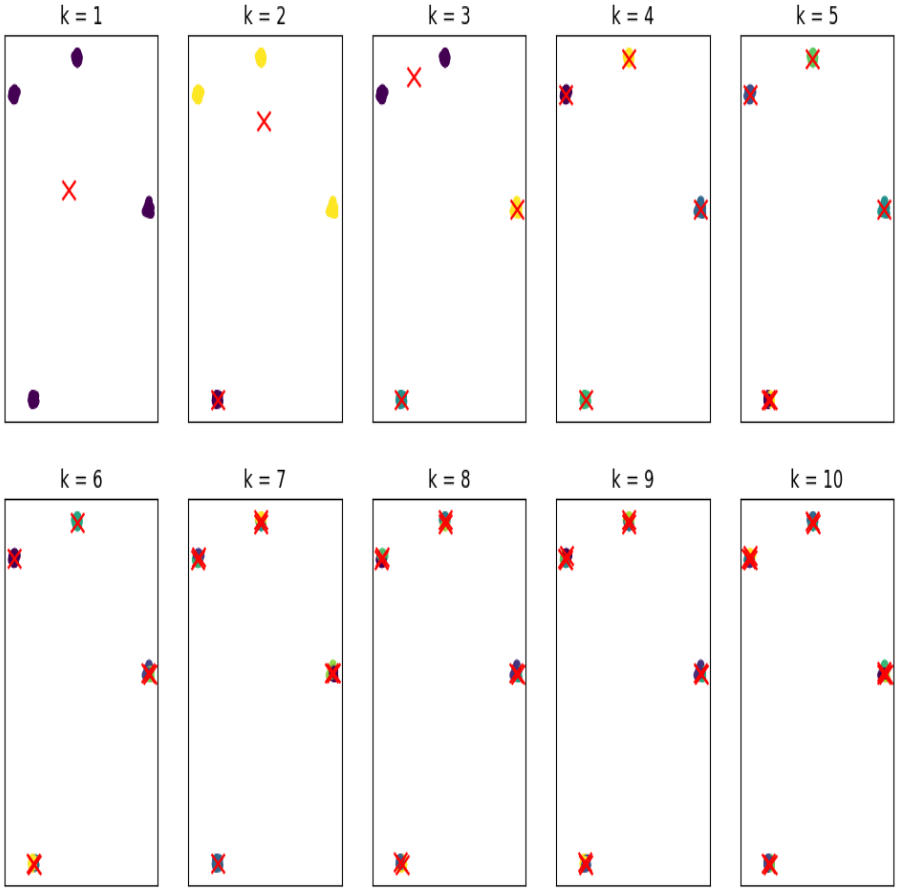

Figure 6: k-means clusters for k = [1-10] with std_dev = 0.1

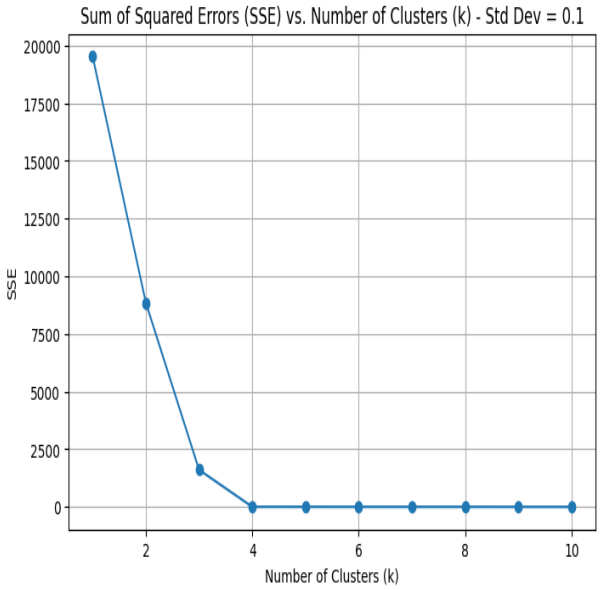

Figure 7: SSE vs. number of cluster for std_dev = 0.1

For the data generated with a higher standard deviation of 2.5 we can observe different findings. Since the the datapoints have a much lower density, it becomes hard to clearly distinguish different clusters. When looking at the plot of the cluster assignments by the k-means algorithm, multiple values of k seem to make reasonable cluster assignments. While k = 1 again looks not reasonable, values for k between 2 and 5 provide arguably good clusters. Also higher values for k are looking more reasonable than before. By analysing the SSE plot it is not as clear as before if 4 is the natural number of clusters, because for higher values of k the SSE value still decreases by higher amounts than before. The plots can be found below:

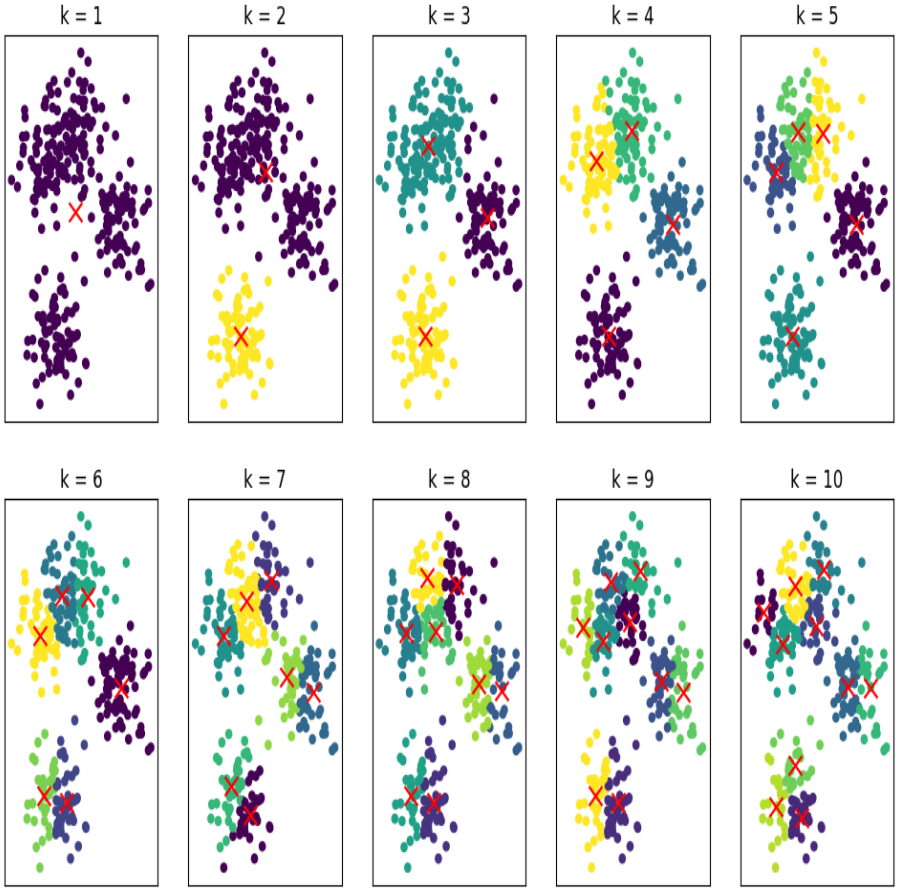

Figure 8: k-means clusters for k = [1-10] with std_dev = 2.5

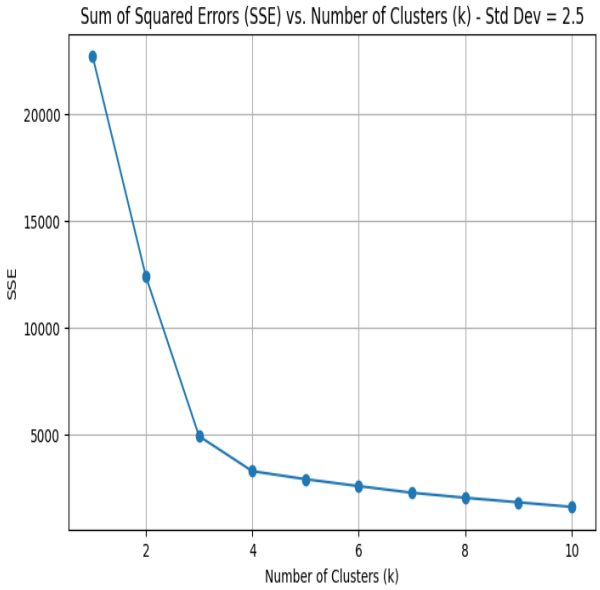

Figure 9: SSE vs. number of cluster for std_dev = 2.5

e) In this part of the assignment, we will repeat the experiments seen above with the difference that we will set the random_state parameter now to a fixed integer (42). We will first check the similarity of the clustering results and then propose an extension of the k-means initialization that is less dependent on the parameter. The plots of the clusters obtained from the k-means algorithm with a fixed random state can be seen here:

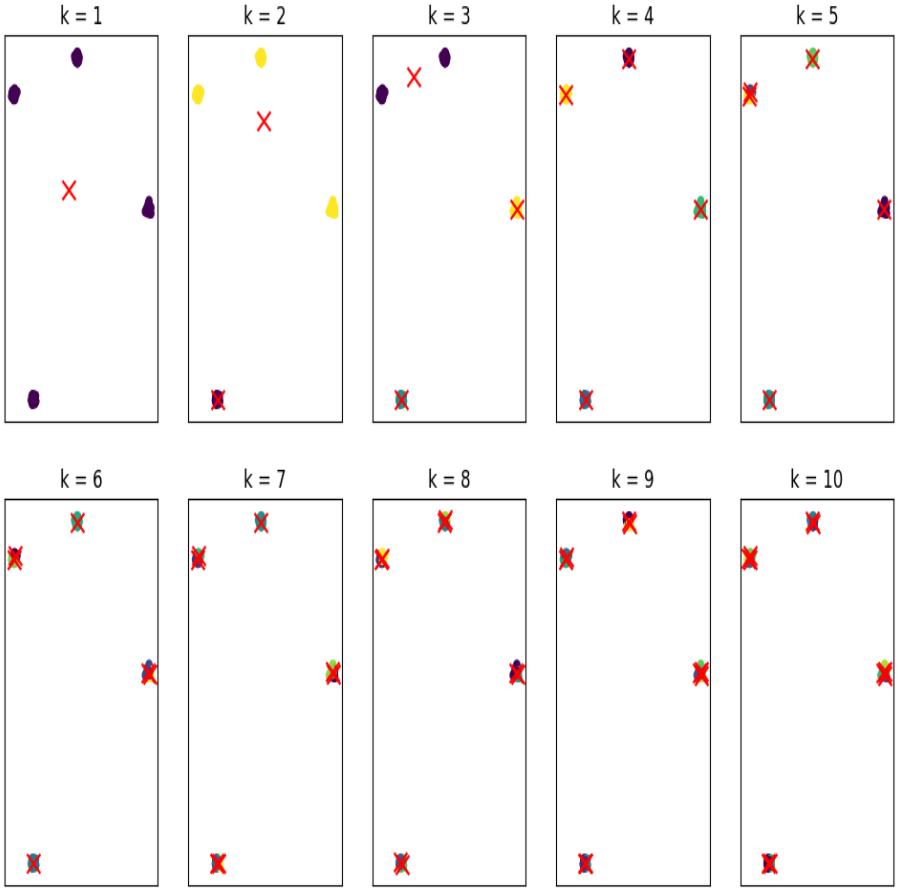

Figure 10: k-means clusters for k = [1-10] with std_dev = 0.1 and fixed random_state

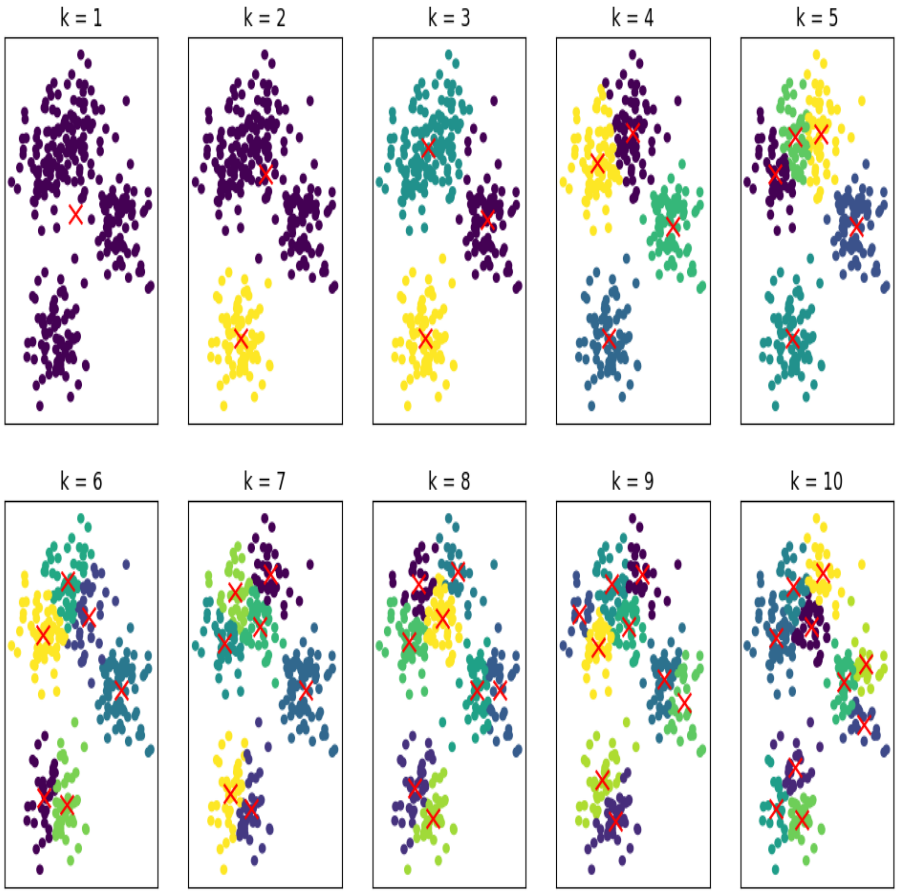

Figure 11: k-means clusters for k = [1-10] with std_dev = 2.5 and fixed random_state

While the clustering solutions in terms of optimal number of the obtained clusters and therefore the obtained SSE values (can be seen in the notebook) are very similar, we can observe other differences. Especially, when comparing the plots for the k-means clusters with standard deviation = 2.5 we can observe that the cluster labels (colors of the clusters) are different. This is due to the random initialization of the k-means algorithm which effects the initial centroids placement. It is expected that when rerunning the k-means algorithm with random_state = None the cluster labels can potentially change again while they are expected to stay the same when a random seed is set in the k-means initialization. A reason for the difference in the clustering solution can be that due to the randomness of the initialization a different local minima for the SSE is found which can lead to different clusters. (Note: During the implementation, a random seed for the data generation has been set as well. Otherwise the datapoints would differ each time and a comparison would not be reasonable.)

Furthermore, we want to propose an extension to the k-means initialization that is less dependent on random_state. One way to extend the k-means initialization is to set the init parameter to k-means++. K-means++ improves the initial centroid placement by spreading them out across the data. It helps to prevent the algorithm from converging to sub-optimal solutions.

Assignment 2

In this second assignment we want to analyse the effects of different linkage methods (min-,max-, and average-Linkage) for hierarchical clustering. After performing hierarchical clustering with these three methods, we plot the results as dendrograms. In a dendrogram the height at which two clusters are

merged reflects the distance between two clusters. The obtained dendrograms can be seen below:

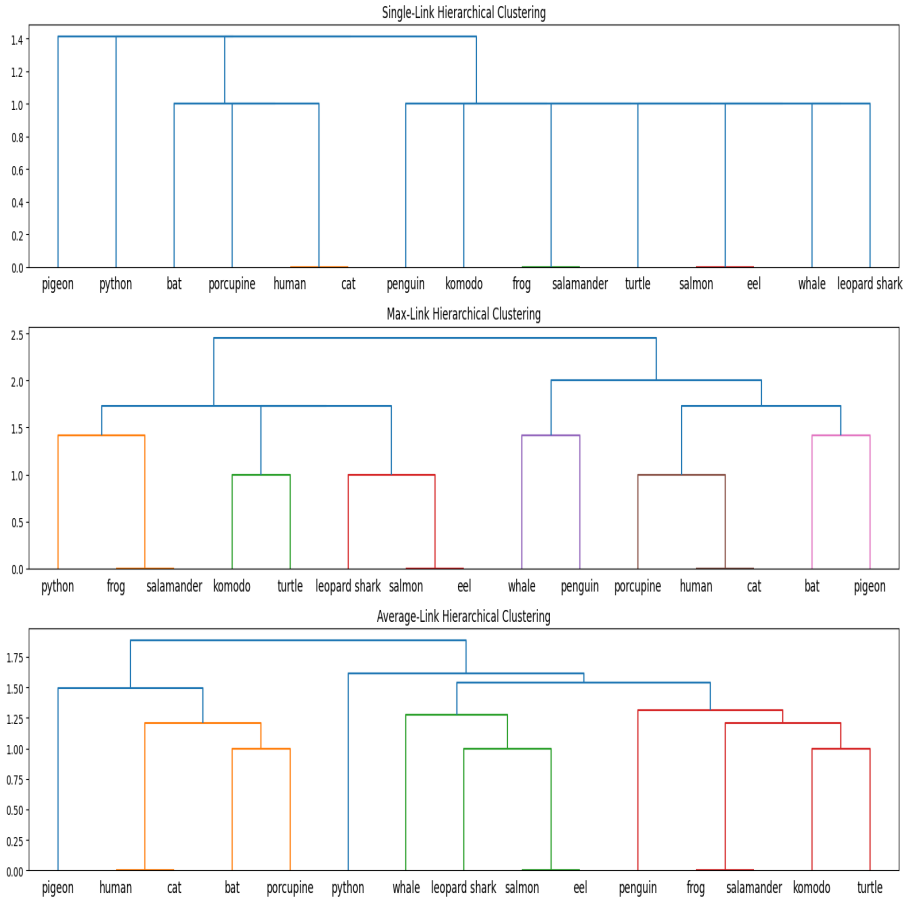

Figure 12: Dendrograms for different linkage methods

To find the most natural hierarchy we have look at the dendrograms in more detail. The first dendrogram using the single-link method results in just a few clearly distinguishable clusters, since most of them have equal height. Therefore, it does not look like a natural cluster hierarchy. On the other hand, the second dendrogram using max-link provides clearly separable clusters. The height of the merged cluster bins differs enough to give a natural insight in the underlying hierarchy. The third dendrogram using average-link does also provide some reasonable distinguishable clusters but their height differs less than with the max-link method. Overall, the hierarchy obtained by using the max-link methods is the most natural with the given data.

Assignment 3

In this third part of the assignment we are going to analyse the DBSCAN clustering algorithm for the chameleon dataset which is visualized here:

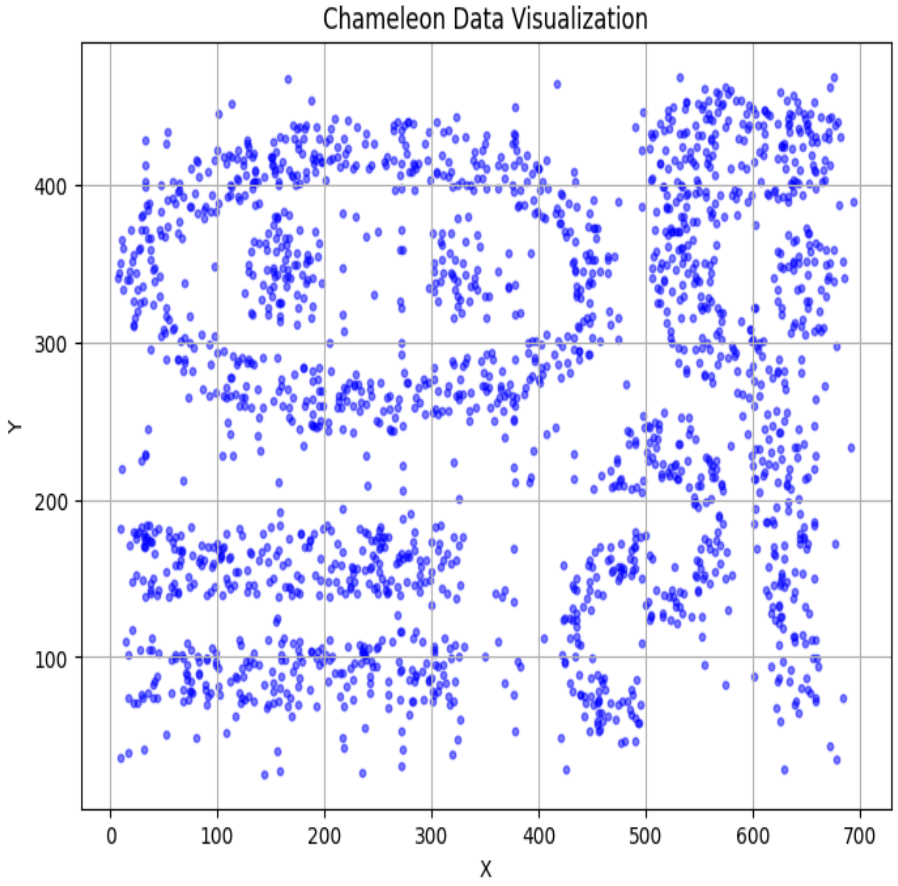

Figure 13: Chameleon dataset visualization

First, we are going to run the algorithm with a maximum radius of the neighbourhood of 15.5 and 5 minimum points in an Eps-neighborhood. After that we will experiment with eps and min_samples values between 1 and 21. The visualization for the first run of the DBSCAN algorithm can be seen here:

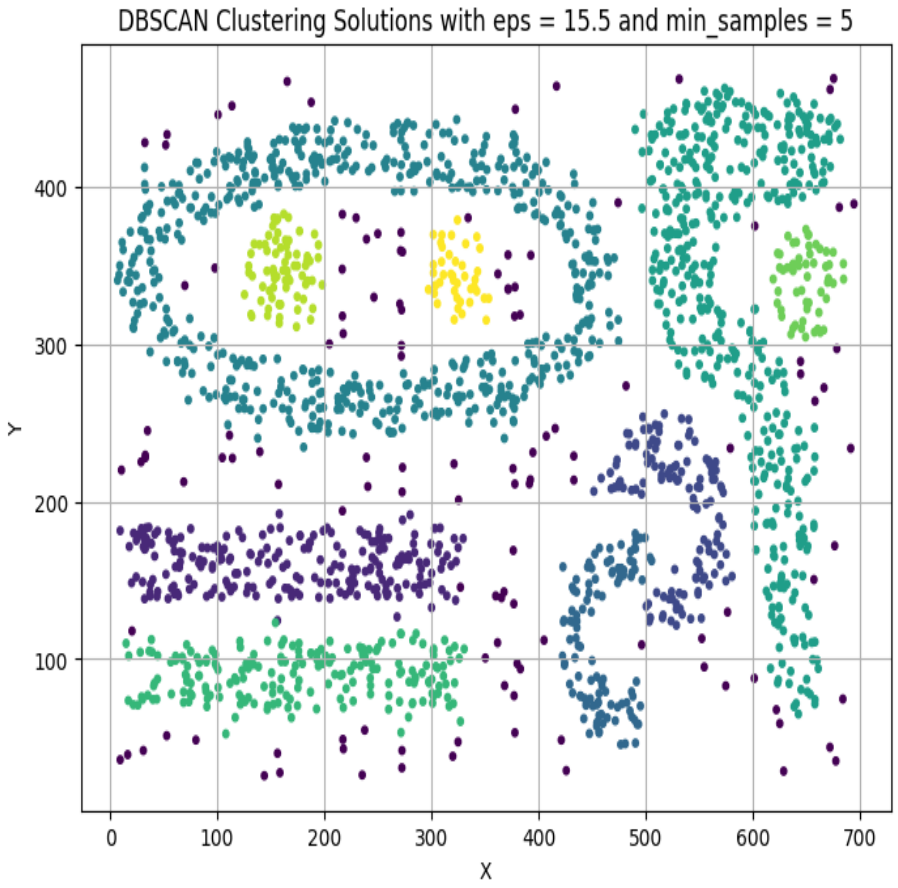

Figure 14: DBSCAN Clustering with eps = 15.5 and min_samples = 5

We can observe that nearly all the points that are part of a shape are assigned to one cluster. This results in a clear separation of the points that should be clustered together and points that can be considered as outliers. The different labels (colors) in the plot indicate that DBSCAN recognises the points in the different shapes as individual clusters. It can be said, that a good trade-off between the maximum radius of a neighbourhood and minimum number of points in an EPS-neighbourhood was found.
In the last part of this exercise we are going to run the DBSCAN algorithm with different values for the mentioned parameters. The plot of the clustering solutions can be seen here:

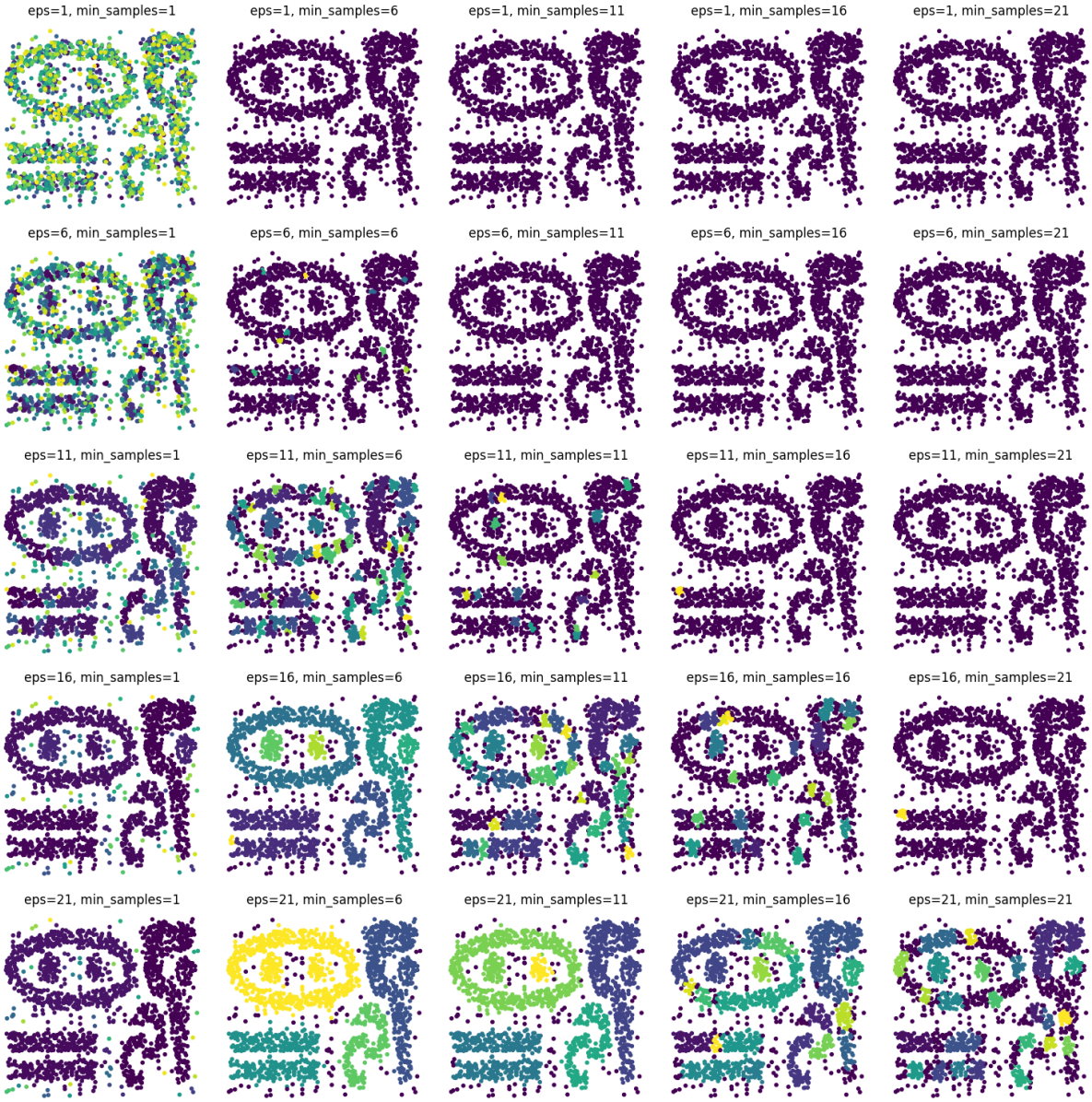

Figure 15: DBSCAN Clustering with different eps and min_samples values

We can observe that for an eps-value of 1, no matter the minimum number of points in the eps-neighbourhood, no reasonable clustering solution can be found. We can further see that for a min_sample value of 1 a lot of single points are assigned to one cluster while for values of 6 or larger all points are assigned to the same cluster. If we increase the eps-value to 6, similar observations can be made. The difference lies in the decreased density in different cluster when min_samples = 1 and that some outlier clusters are identified when min_samples = 6. For eps = 11 and min_samples = 1 first reasonable clusters are recognizable. Especially, point-like shapes with high density are among the first correctly classified clusters. This observation holds for a min_samples value of 6 as well but with higher values a tendency to assign all points to a single cluster is noticeable again. Interestingly, when we increase the eps-value to 16 most of the shapes in the datapoints receive the same label and the outliers/ noise receives different labels. For eps = 16 and min_samples = 6 we can nearly label all points that belong to a shape correctly with minor inaccuracies. Outliers are now assigned to a single cluster label and therefore made accessible. When further increasing the minimum number of points in an eps-neighbourhood breaks down the cluster labels in the shapes and therefore makes the clustering result worse. For eps = 21 we see similar results

when the min_samples = 1. When increasing the min_samples value we can again see clear clusters according to the shapes a datapoint belongs to. Since the maximum radius of the neighbourhood is increased, we can observe that now shapes that are close to each other get assigned the same label which leads to poorer clustering result than before. When further increasing min_samples we observe again that shapes tend to be separated into sub-clusters. Overall, we can observe that the clustering result of the DBSCAN algorithm highly depends of the maximum radius of the neighbourhood and the minimum number of points in an eps-neighbourhood of a point. Since DBSCAN is a density-based clustering approach it can identify arbitrary shapes and can also be suitable to distinguish outliers and noise from clusters in highly complex datasets.

Academic integrity Declaration

I, Ben Beißner, hereby declare that I have not used large language models or any automated tools for generating the written answers and interpretations in this Analytical Report.