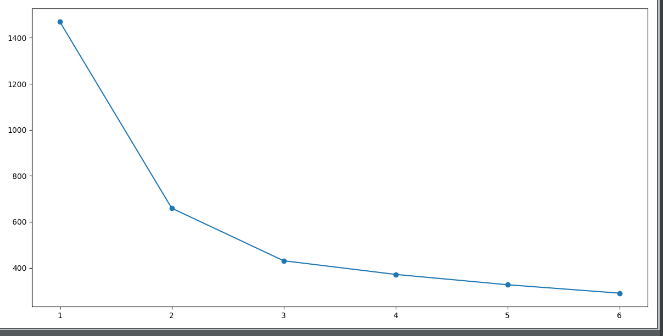
Report on Data Clustering – K-means and Gaussian Mixture Model.

Preparing data

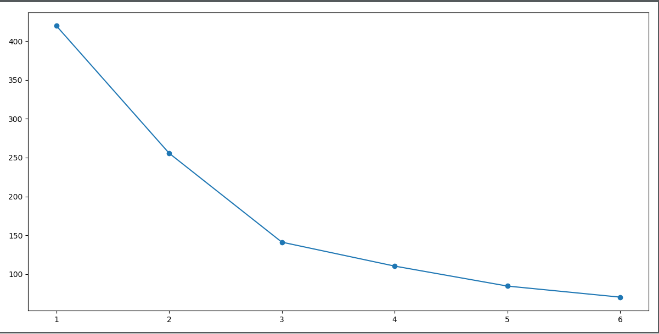
Before we could apply K-means and gaussian mixture model (hereby referred to as GMM) to the seed dataset, we had to make the dataset fully processable. At first, we used regex to position data with respect to the delimiters tabs and double tabs. Some columns were separated with more tabs in between than other columns. After the data was positioned, we used NumPy and pandas to convert the dataset to a NumPy array.

Balancing under- and overfitting – finding the right number of clusters

When applying the algorithms, the last feature of the dataset was excluded. This column gave the actual classification of each row, which in this context would make the classification supervised and result in a perfect score (all clusters could be arranged by the class feature). Instead we took the classification feature out and looked for a suitable technique for analyzing the dataset to find the ideal number of clusters. In both K-means and gmm, specifying the number of components/clusters is key in avoiding under- and overfitting. A n < 5 clusters would equal less wrongly estimated datapoints and smaller chance of overlap. However, by overfitting the dataset, the number of clusters would surpass the real number of clusters and become too specific. Too few clusters, underfitting, could result in a crude classification with no useful information. To solve this, we applied ‘elbow analysis’ and ‘silhouette analysis’ to our dataset. The elbow method applies K-Means to range *r* of possible clusters, n-components(*n*), and calculates the sum of squared errors of prediction (SSE). The SSE calculate a score given on how close a prediction is to the actual data. The SSE is then graphed, and one tries to identify the “elbow point” where the graph flattens out. From our graph (figure 1) we could infer that the right number of clusters(*n-components)* was either *n=*2 or *n=*3. The results were ambiguous so to improve our elbow we applied principal component analysis(PCA) to the data. PCA reduces the dimensionality of the data by removing features that have a small amount of variance. We reduced the data to two components. This decision was built on observation of the data itself where a lot of features where seemingly with little or random variance, so they could be removed without compromising the data too much. The result post-PCA would therefore clean the data and make it easier to classify. The resulting graph can be seen in figure 2 which has a clear indication that the *n* number of clusters should be three (3).



*Figure 1 – elbow analysis pre-PCA. X-axis: number of clusters, Y-axis: Percentage of information variance*

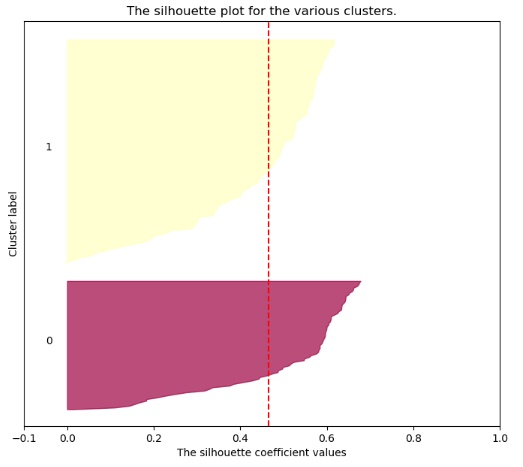
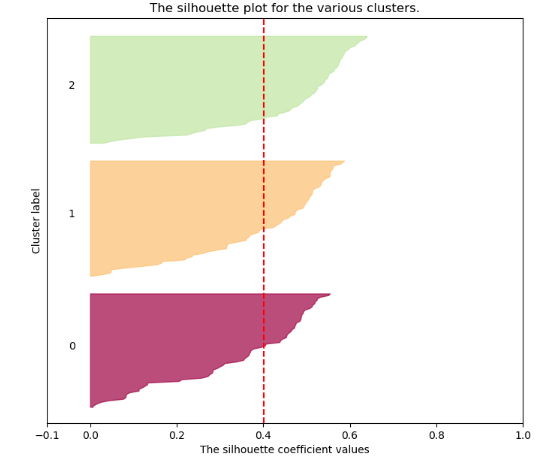
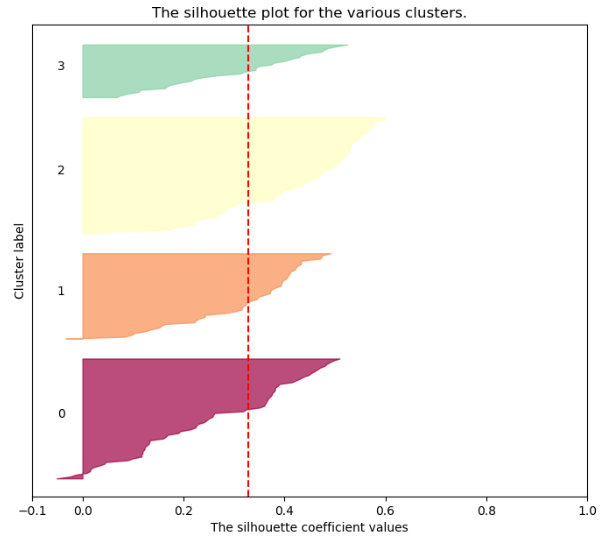


*Figure 2 – elbow analysis post-PCA reduction. X-axis: number of clusters, Y-axis: percentage of information variance.*

Silhouette analysis returns a value which is a calculation of cohesion and separation. Cohesion is a measure on how similar an object is to the other objects in its own cluster and separation is a measure on how different it is compared to the other clusters.

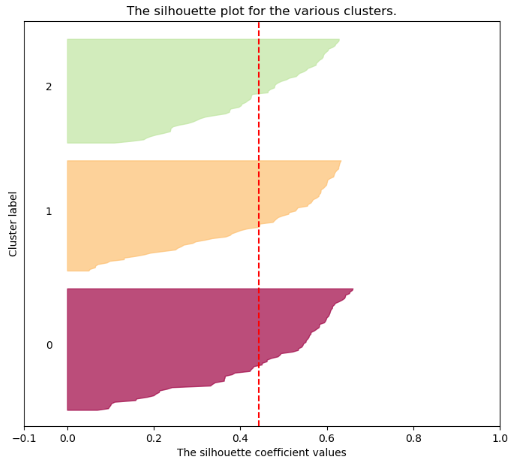
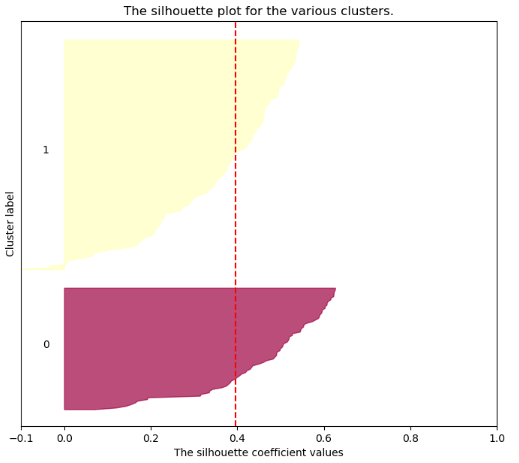
The analysis makes use of a graphical representation of each cluster and the objects cohesion and separation. Here we also had a pre- and post-PCA reduction graph. Again, we reduce the dimensionality to two and saw, in our opinion, a clearer representation of the data and the optimal number of clusters post-PCA.

Pre-PCA (figure 3).



*Figure 3 – Silhouette Analysis.   
X-axis = coefficient value*. *Y-axis = Number of clusters. Red line = Avg. coefficient value (higher is better).*

The score ranges from -0.1 to 1, and higher is better. A high average (the red line) indicates good cohesion. If any nodes in a cluster is going into the negatives as seen on the 4-cluster silhouette, it indicates that nodes in that cluster have been misplaced. Overall these silhouettes indicated to us that the right number of cluster is either *n=2* or *n=3*, with *n=2* being the preferred alternative.

Post-PCA (Figure4)

*Figure 4 – Silhouette Analysis. X-axis = coefficient value*. *Y-axis = Number of clusters. Red line = Avg. coefficient value (higher is better).*

This time the results were in line with our results from the elbow method. The *n=3* has a higher average coefficient score (the red line) and has no nodes in the cluster with a negative value.

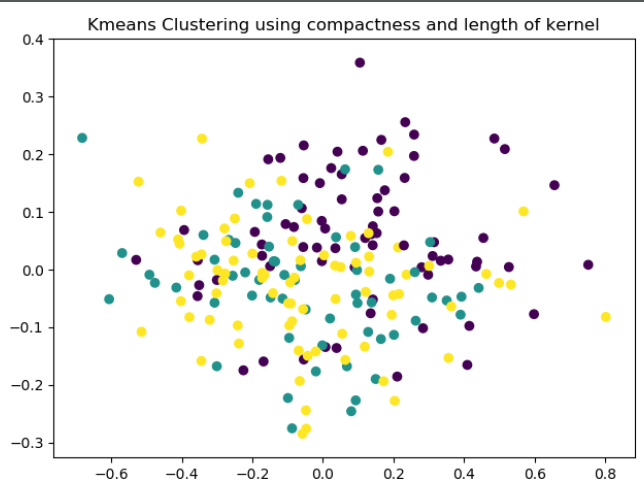
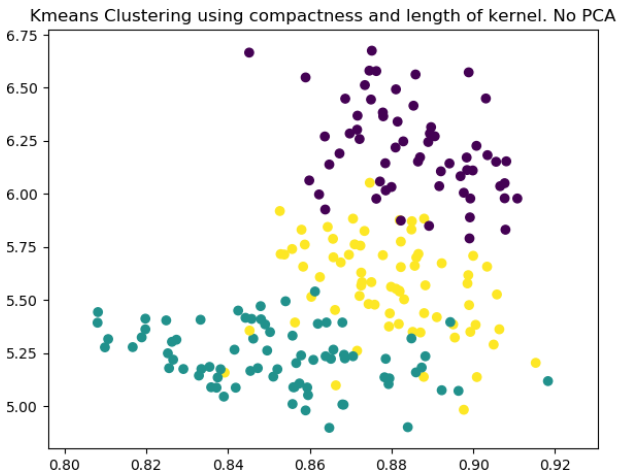
The conclusion from our both our analysis’ is that the right number of n-components(clusters) in our dataset is three (3).

Applying the clustering algorithms

With this information in hand, we fitted K-means and the GMM to our dataset. To measure the performance, we visualized on 2D plane. For visualizing the dataset, we used the matlotlib library. Here, we had to decide which two features of seven that provided the best cluster variance. We were easily able to tell that the two first features of the dataset, ‘area’ and ‘perimeter’ respectively, were the most cluster-significant features. Despite being aware of this, we tried various other feature combinations, trying to find the visualization where datapoints overlap the least. For the parameters included in the K-means and the gaussian mixture model sklearn classes, we did a lot of tweaking to find the values that gave valuable results. We also experimented by including and excluding PCA reduction throughout the process.

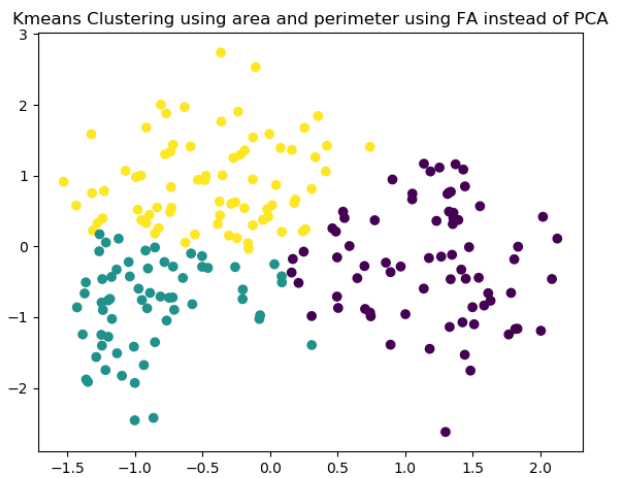
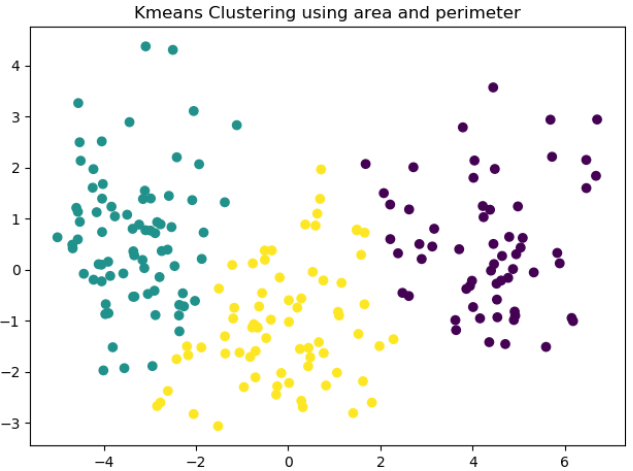
K-means

Applying k-means, we observed that it was a selection of features that gave quite good results when plotted onto a 2D chart. Some of the best result, were surprisingly achieved without the use dimensionality reductions. Feature three and four (compactness and length of kernel) shown in figure 4 and 5, shows an approach of the dataset where the algorithm clusters well without PCA. We found that figure 4 shows a relatively satisfying clustering result, while the clustering shown in figure 5 was less successful.



*Figure 4 and 5 – k-means with compactness and kernel length as features with and without PCA reduction.*

Feature one and two clustered our dataset the best, shown in figure 6. Here we used PCA reduction for dimensionality reduction, and reduced dimensionality to two components. Adjusting the component parameter for the PCA class had to be done with respect to which features we tried to combine, not to exceed indices. There were a lot of other parameters in the PCA class to adjust, but none had any significant effect on our relatively small dataset. Both the sklearn documentation and our own experimentation indicate that the impact would be bigger if the dataset was bigger. For the K-means class parameters, we tried the various algorithm executions, but they all had the same clustering. In addition to applying PCA, we tried another reduction method: factor analysis, shown in figure 7.

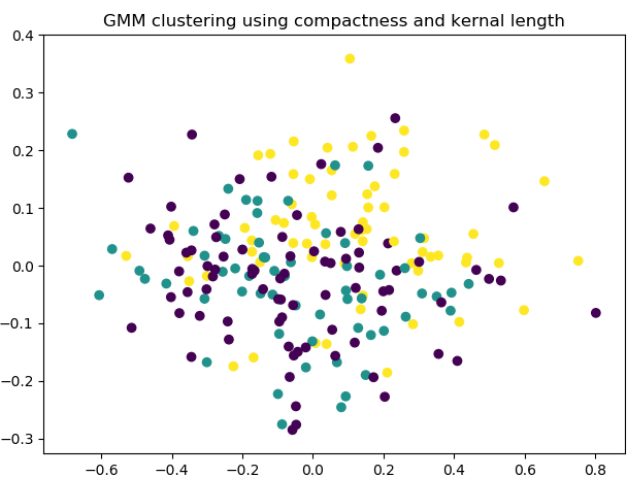
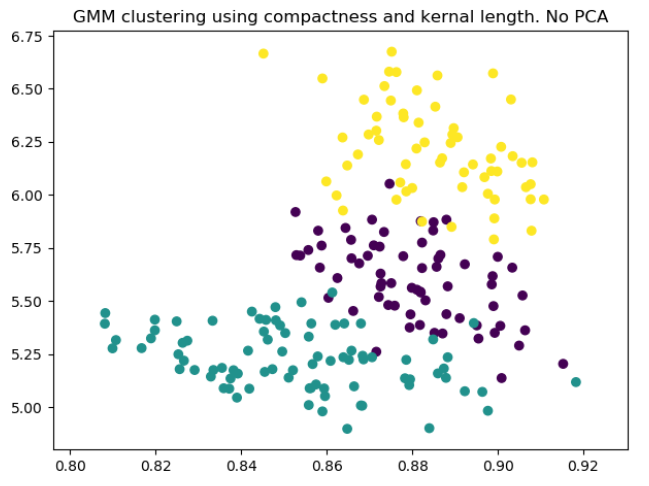


*Figure 6 and 7 - K-means with area and perimeter as features, with PCA reduction in figure 6 and with FA reduction in figure 7.*

We found FA reduction also performed well, but that PCA reduction clustered the best.

Gaussian mixture model

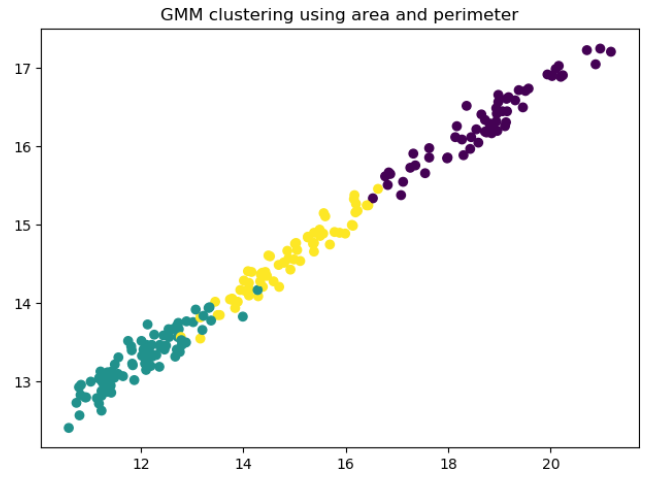
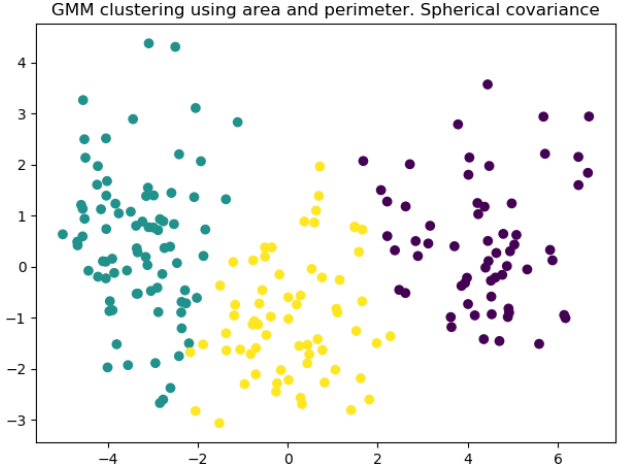
Just as we did for the K-means method, we experimented with various parameters, both for the GMM class and for PCA and FA reduction, when applying GMM. Also with GMM, we experienced that feature combinations other than the first two features, resulted in relatively satisfying clustering, but only without the use of dimensionality reduction methods.



*Figure 8 and 9 – GMM with compactness and kernel length as features with and without PCA reduction.*

An incorporated method in the GMM class is the covariance type we choose for the algorithm. GMM offers four different variables on the covariance parameter on preventing covariance of the clusters being estimated. Running these, we get slightly various results – some perform well on a selection of features, while others perform better at other features. Tweaking these features helped us when searching for the best use of GMM on the dataset.

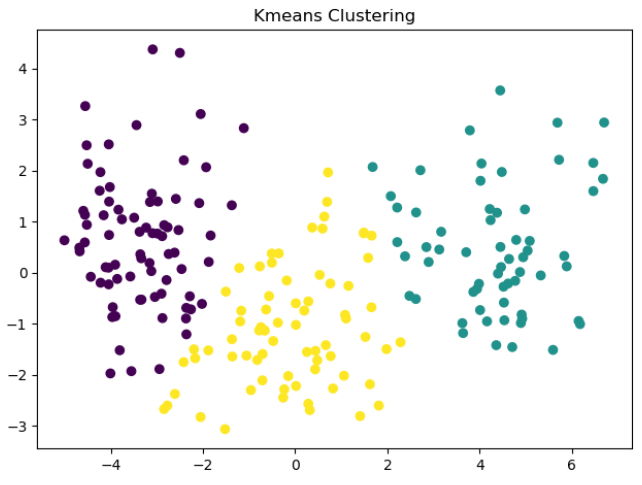
It was feature 1 and 2 (shown in figure 10 and 11, where figure 11 does not apply PCA) that clustered the dataset the best. Dimensionality reduction performed better than including all dimensionalities, and we found that PCA reduction was a bit more precise than FA. For the covariance type parameter, all other values than ‘spherical’ had a significant set of overlapping datapoints. Spherical performed very well.



*Figure 10 and 11 – GMM with area and perimeter as features, with and without PCA reduction.*

Comparing K-means and GMM

After having applied both K-means and GMM to the seed dataset, it is our understanding that the two algorithms can perform very accurate clustering when having adjusted the key parameters rightfully. We also found that the results we were the most pleased with, were strikingly alike. Shown in figure 12 and 13, they cluster almost every datapoint the same, with exception of only a small number. However, we needed to do more fitting with the covariance type within the GMM class to create the result.



*Figure 12 and 13 – Our best results of K-means and GMM on the seed dataset.*