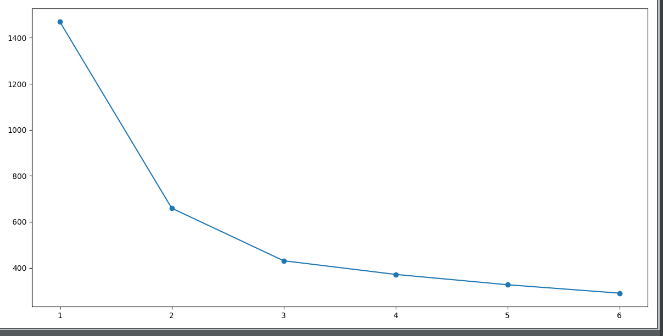
Report on Data Clustering – Kmeans and Gaussian Mixture Model.

Preparing data

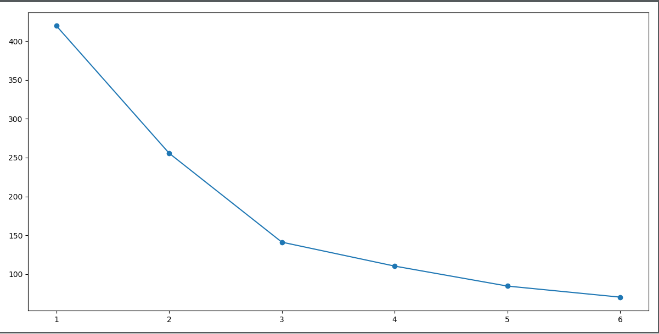
Before we could apply Kmeans and gaussian mixture model (GMM) to the seed dataset, we had to process the data. At first, we used regex delimiting to tabs and double tabs. After the data was positioned, we used NumPy and pandas to convert the dataset to a NumPy array.

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

We excluded the last feature of the dataset. This column gave the classification of each row, which in this context would enable clusters arranged by the class feature we were supposed to predict. Instead we looked for a technique for analyzing the remainder of the dataset to find the ideal number of clusters/components(*n*). In both Kmeans and GMM, specifying the number of clusters is key in avoiding under- and overfitting. An n set high would equal less wrongly estimated datapoints and smaller chance of overlap. However, by overfitting we would risk many false positives. By underfitting, we would risk a too crude classification. To solve this, we applied ‘elbow analysis’(EA) and ‘silhouette analysis’(SA). The elbow method applies Kmeans to range *r* of possible *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 graphed, and one identifies an “elbow point” where the graph flattens out. From our graph (fig. 1) we could infer that the right number of cluster was either *n=*2 or *n=*3. The results were ambiguous, so to improve our elbow we used principal component analysis(PCA) to the data. PCA reduces data dimensionality by removing features that have insignificant variance. We reduced the data to *n=2,* to remove data without useful variance. The resulting graph can be seen in fig. 2 which indicates *n=*3.



*Fig. 1 – EA pre-PCA. X-axis: number of clusters, Y-axis: Percentage of information variance*

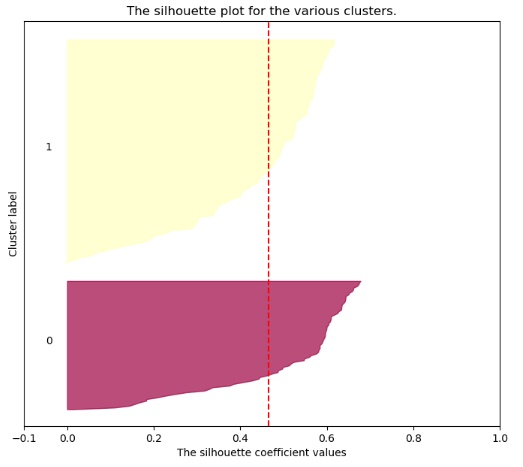
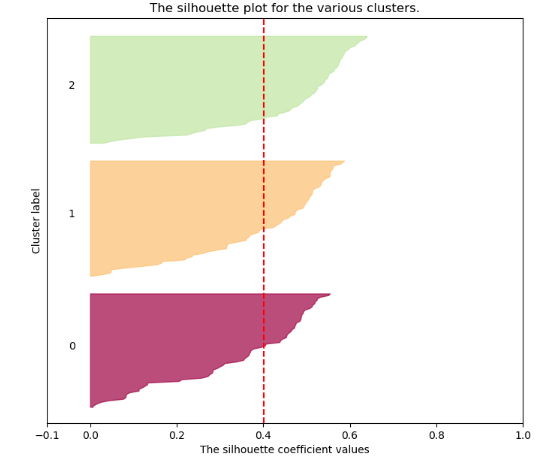
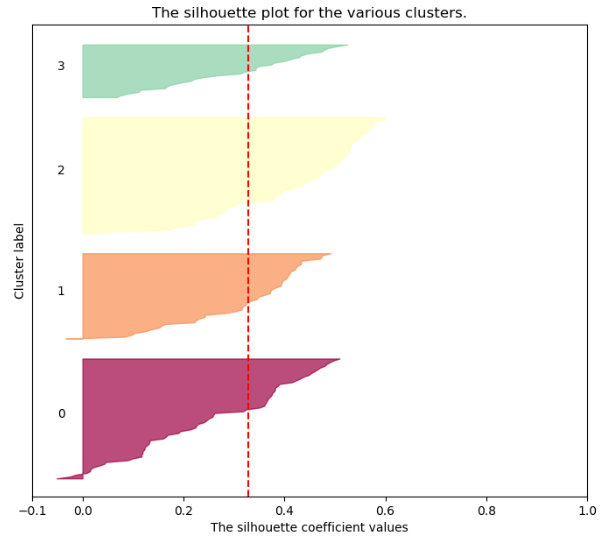


*Fig. 2 – EA post-PCA reduction. X-axis: number of clusters, Y-axis: percentage of information variance.*

SA returns a value which is a calculation of cohesion and separation. Cohesion is a measure on how similar an object is to other objects in its own cluster and separation is a measure on how different it is compared to the other clusters. This is represented by a coefficient value(coef) where higher equals better.

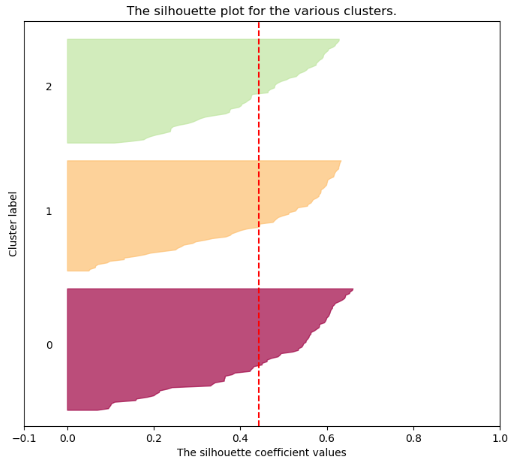
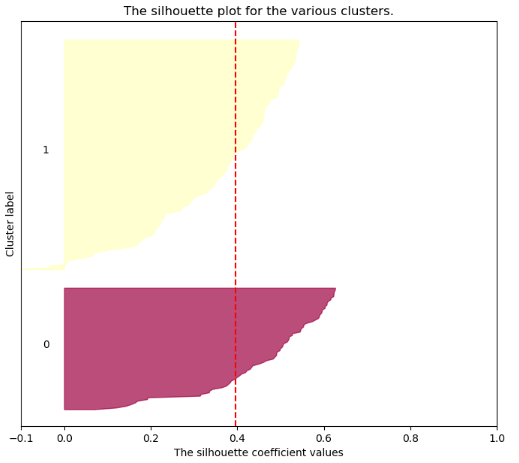
The analysis makes use of a graphical representation of each cluster and the objects cohesion and separation. Here we also had a pre- and post-PCA reduction graph. Again, we reduce the dimensionality to 2 and saw, in our opinion, a clearer representation of the data and the optimal number of clusters post-PCA.

Pre-PCA (fig. 3).



*Fig. 3 – SA.   
X-axis = co\ef value*. *Y-axis = Number of clusters. Red line = Avg. coef value.*

The score ranges from -0.1 to 1. Negative coef values (as seen in the 4-cluster silhouette), indicates that nodes in that cluster are misplaced. Overall the silhouettes indicates that the right *n* value is *n=2* or *n=3*, with *n=2* being the preferred alternative.

Post-PCA (Fig.4)

*Fig. 4 – SA. X-axis = coef value*. *Y-axis = Number of clusters. Red line = Avg. coef value).*

This time the results matched 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.

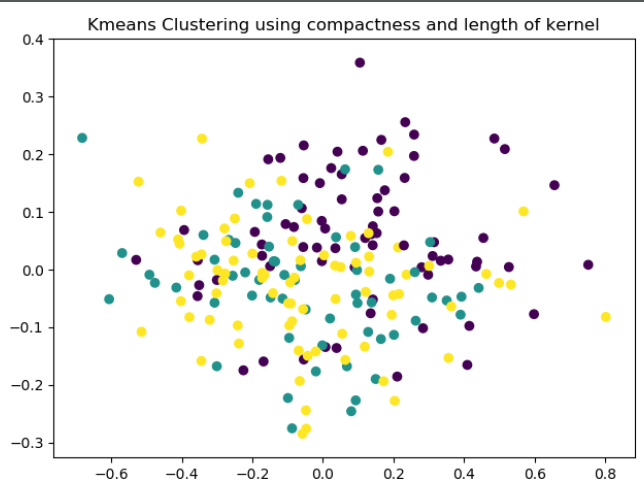
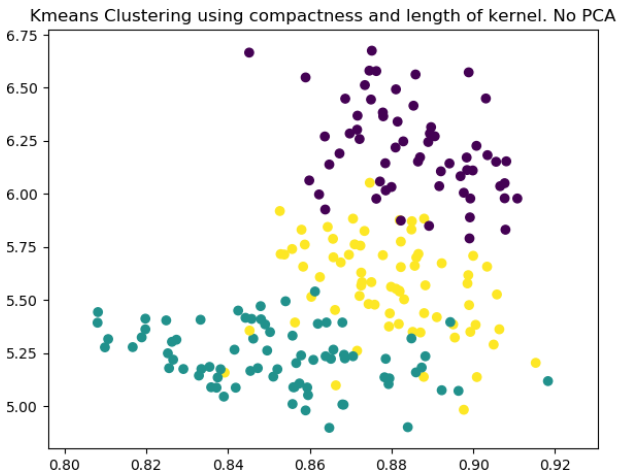
The conclusion from our both our analysis’ is that the right number of clusters(components) is three.

Applying the clustering algorithms

To measure the performance of our learning algorithms, we visualized on a 2D plane. To visualize we used the matlotlib library. We decided which two features of the seven that provided the best angle to separate the clusters. We found that the two first features of the dataset, ‘area’ and ‘perimeter’, were the most cluster-significant features. We nonetheless, checked multiple other various to be certain in our choice. We also tweaked with parameters included in the Kmeans and the gmm classes. We also experimented by including and excluding PCA.

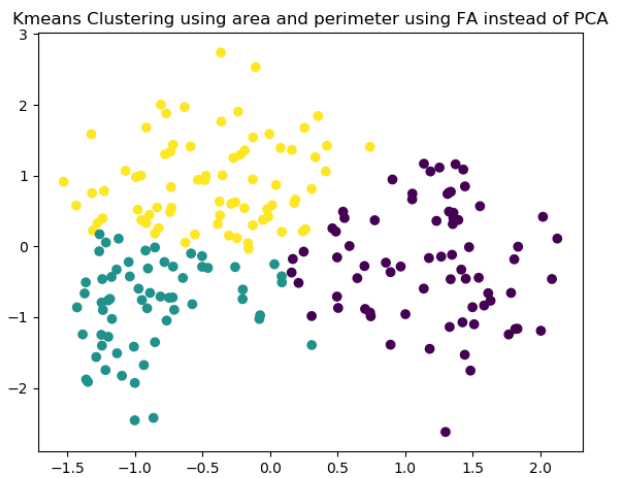
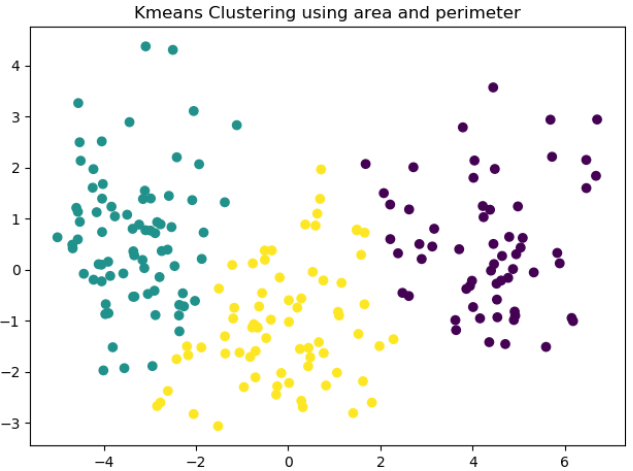
Kmeans

Applying kmeans, we observed that several features that gave good results when visualized. Some of the best result, were achieved without the use dimensionality reductions (DR). Feature three and four (compactness and length of kernel) shown in fig. 4 and 5, substantiates this.



*Fig. 4 and 5 – kmeans with compactness and kernel length as features with and without PCA reduction.*

Feature one and two clustered our dataset the best, shown in fig. 6. We used PCA reduction for DR, and reduced dimensionality to two components. Adjusting the component parameter for the PCA was done to check a multitude of different “angles”. 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 indicated this. For the Kmeans class parameters, we tried the various algorithm executions, but clustered the same. In addition to applying PCA, we tried another reduction method: factor analysis, shown in fig. 7.

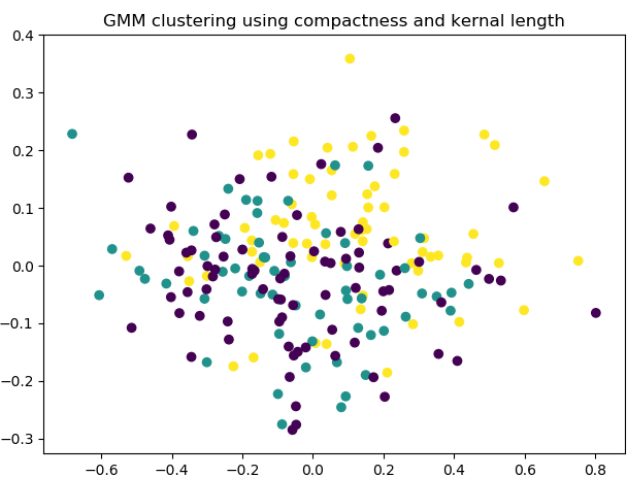
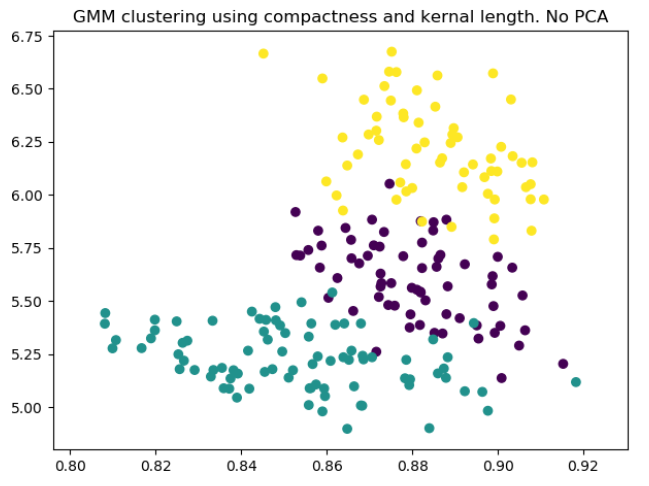


*Fig. 6 and 7 - Kmeans with area and perimeter as features, with PCA reduction in fig. 6 and with FA reduction in fig. 7.*

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

Gaussian mixture model

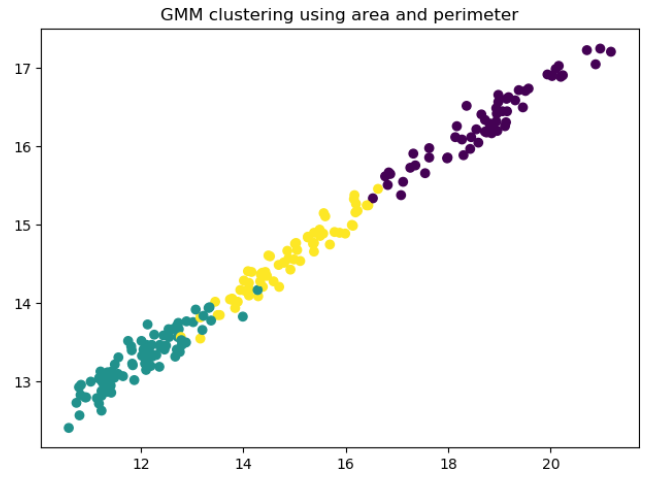
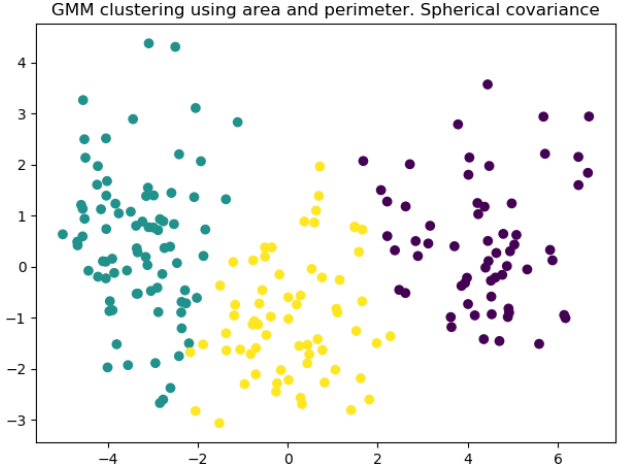
Just as we did for the Kmeans method, we experimented with various parameters, both for the GMM class and for PCA and FA reduction. For gmm too, we experienced that feature combinations other than the first two features, resulted in relatively satisfying clustering, but only without the use of DR methods.



*Fig. 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. GMM offers four different variables on the covariance parameter on preventing covariance of the clusters being estimated. Running these, we get various results – some perform well on a selection of features, while others perform better at other.

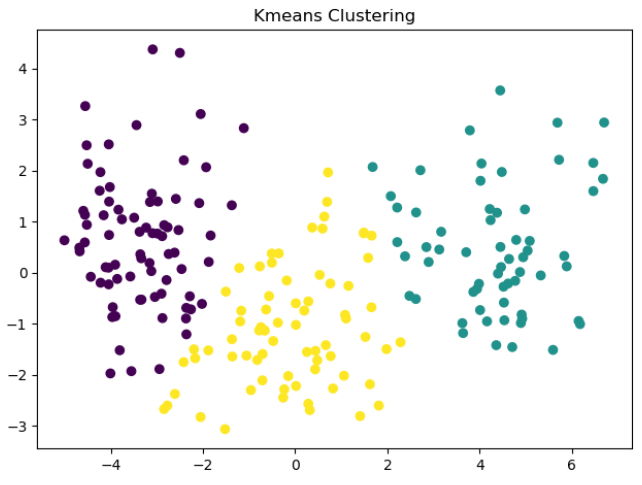
It was feature 1 and 2 (shown in fig. 10 and 11) that clustered the dataset the best. DR performed better than including all dimensionalities, and we found that PCA reduction was more precise than FA. For the covariance type parameter, all other values than ‘spherical’ had a significant set of overlapping datapoints. Spherical performed well.



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

Comparing Kmeans and GMM

After trying both kmeans and GMM, 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 fig. 12 and 13, they cluster almost every datapoint the same, with exception of only a small number.



*Fig. 12 and 13 – Our best results of Kmeans and GMM on the seed dataset.*