UNIVERSITY OF BERGEN

Bergen, Spring 2018

INFO284

Machine Learning

Second Group Assignment

Candidate Numbers: 34, 117, 82

Deadline: 01.06.201

Scikit-learn

This assignment utilized the scikit-learn project (*Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.).*

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.

Finding the right number of clusters

We excluded the last feature of the dataset. This feature is the classification of each datapoint, which would enable clusters to be arranged by their classification. We looked for a technique for analyzing the remainder of the dataset to find the ideal number of clusters/components(*n*). In clustering, specifying the number of clusters is key. An n set too high would lead to few wrongly estimated datapoints and smaller chance of overlap, but it would result in many false positives. By underfitting, we would risk a crude classification. To solve this, we applied 'elbow analysis'(EA) and 'silhouette analysis'(SA). The EA 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 graphed-out result includes "elbow point(s)" where the graph flattens. From our graph (fig. 1) we could infer that both *n*=2 and *n*=3 are possible values. To improve our EA we used principal component analysis(PCA). PCA rotates the data so that features are statistically uncorrelated. We reduced data to *n=2,* to remove all data without useful variance. The result can be seen in fig. 2, indicating *n*=3.

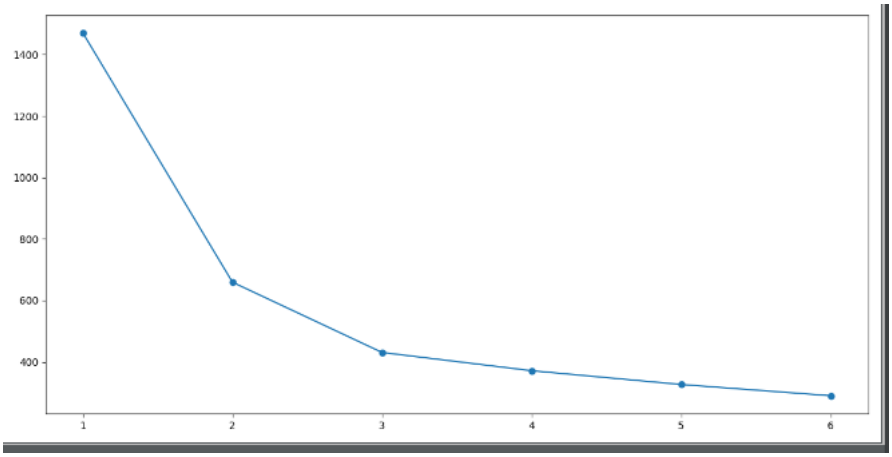

Fig. 1 – EA pre-PCA. X-axis: n-clusters, Y-axis: Percentage of information variance

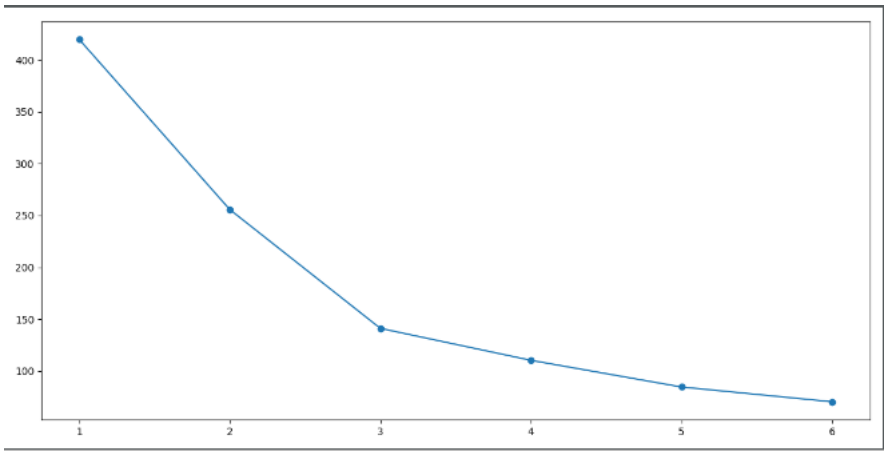

Fig. 2 – EA post-PCA reduction. X-axis: n-clusters, Y-axis: percentage of information variance.

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. We reduce the dimensionality to 2 and observed a clear view of the data and the optimal number of clusters post-PCA.

Pre-PCA (fig. 3).

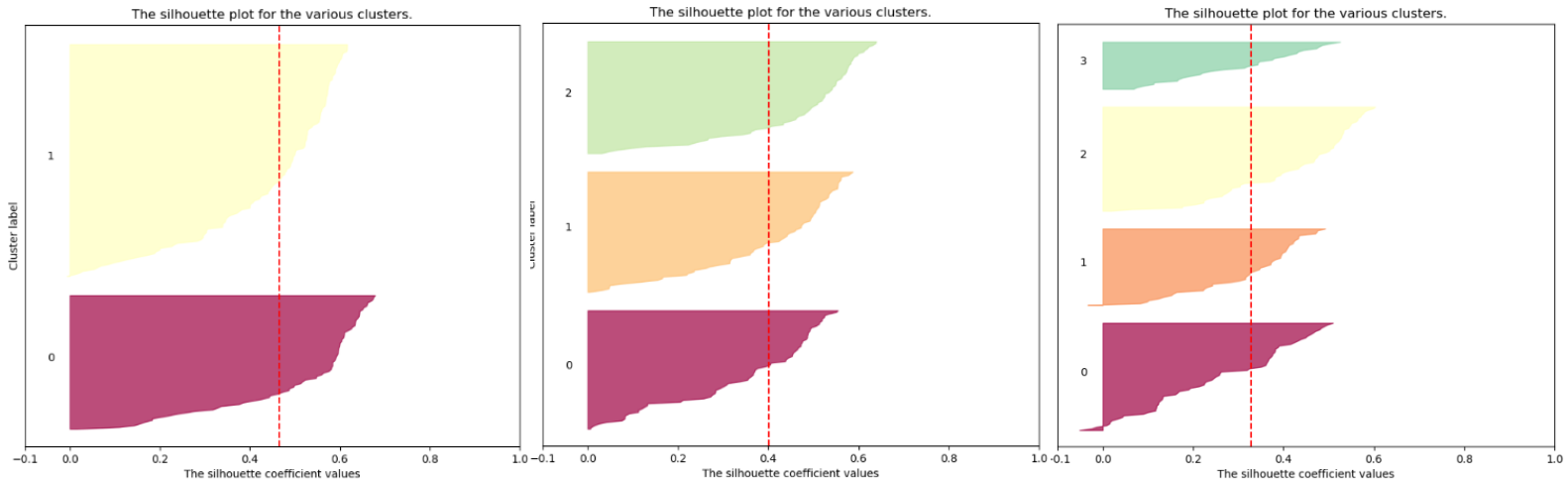

Fig. 3 – SA.

X-axis = co\ef value. Y-axis = n-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. The silhouettes indicate that the right *n* value is *2* or *3*.

Post-PCA (Fig.4)

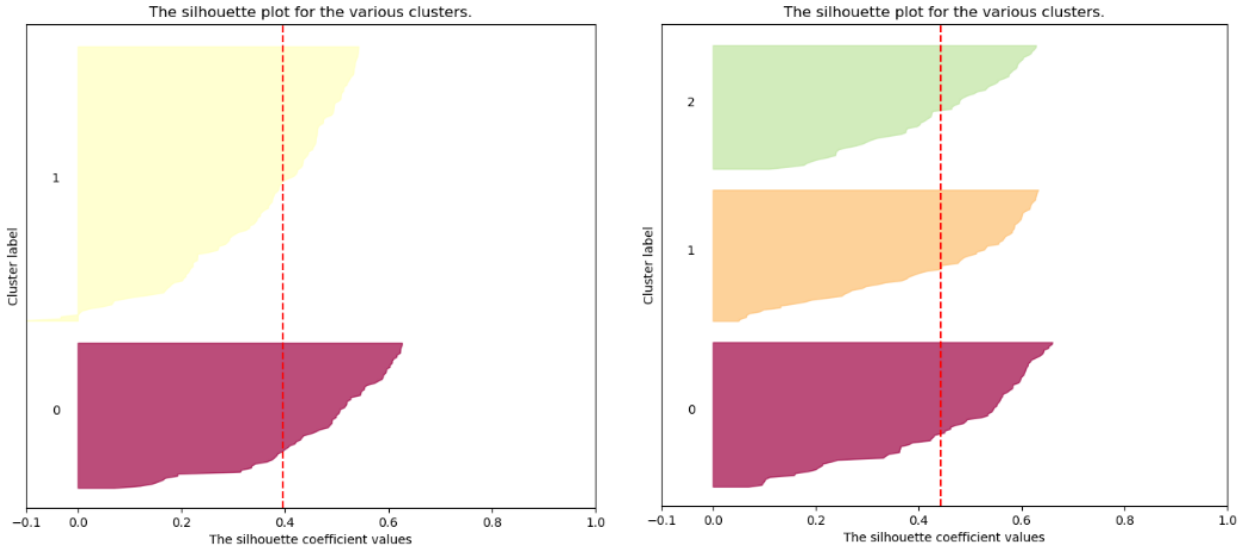

Fig. 4 – SA. X-axis = coef value. Y-axis = n-clusters. Red line = Avg. coef value).

After PCA, the results matched our findings from the elbow method. *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 for n is 3.

Applying the clustering algorithms

To measure the performance of our learning algorithms, we visualized on a 2D plane using matplotlib. At an early stage, we interpreted the dataset and found that the 'area' and 'perimeter' features were most cluster-significant. We nonetheless, checked multiple other various to see if other combinations visualized the features in a more satisfying angle. We tweaked with parameters 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 results we found to be good. Some of the best 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.

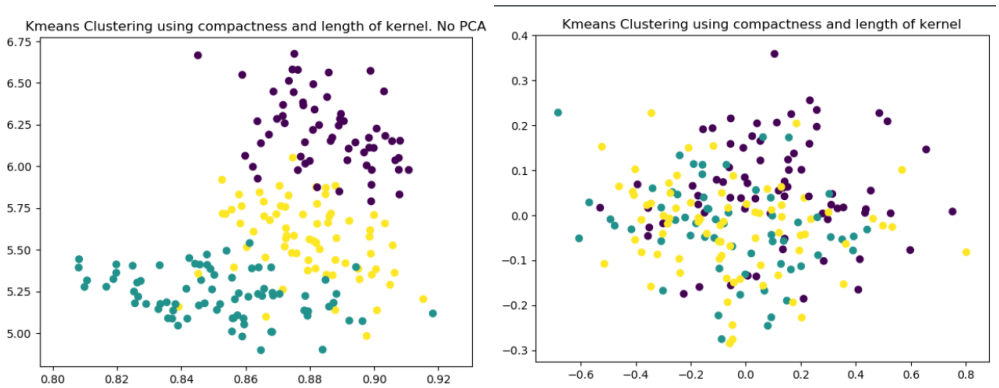

Fig. 4 and 5 – kmeans with compactness and kernel length as features with and without PCA reduction.

Feature one and two clustered our dataset the best, shown in fig. 6. We used PCA reduction for DR, and reduced dimensionality to 2 components. Adjusting the component parameter for the PCA was done to check a multitude of different "angles". There were other parameters in the PCA class to adjust, but none had significant effect on our small dataset. Both the sklearn documentation and our own experimentation indicated this. For the Kmeans class parameters, we tried the various algorithm executions, but they clustered the same. In addition to applying PCA, we tried factor analysis, shown in fig. 7.

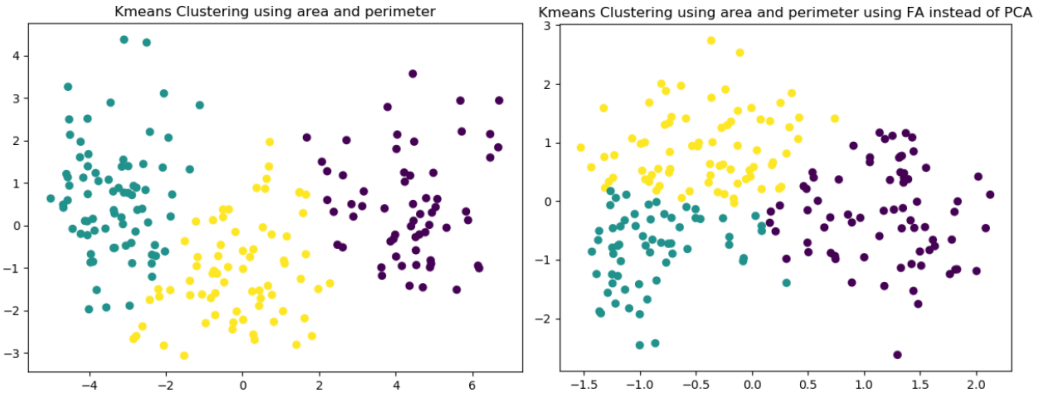

Fig. 6 and 7 - Kmeans with area and perimeter as features, with PCA in fig. 6 and with FA in fig. 7.

We found FA also performed well, but that PCA 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.

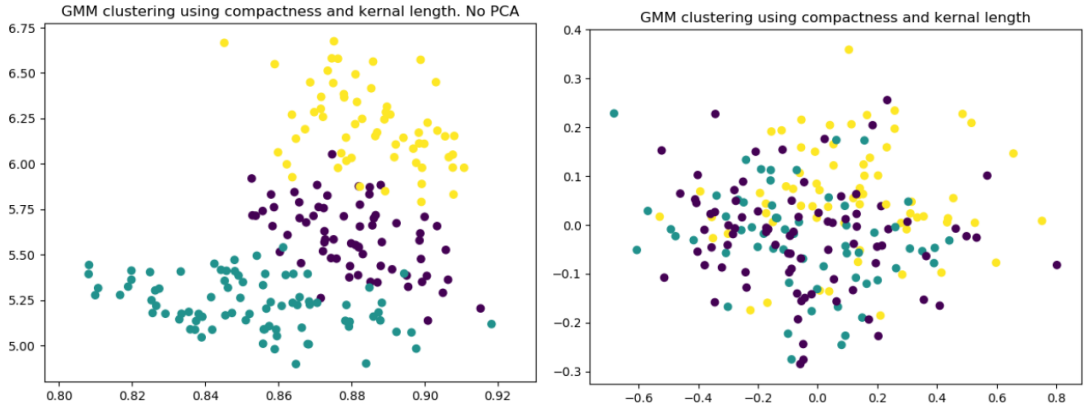

Fig. 8 and 9 – GMM with compactness and kernel length as features with and without PCA reduction.

A method in the GMM class is 'covariance type'. GMM offers four different variables to the covariance parameter to prevent covariance of the clustering. 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.

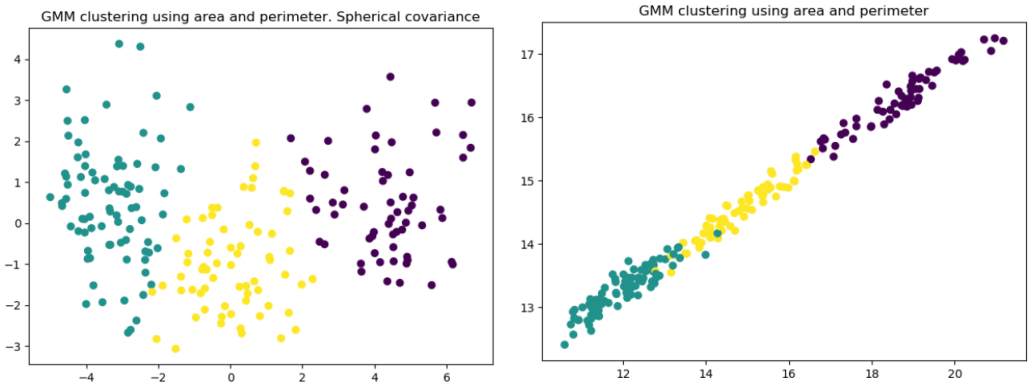

Fig. 10 and 11 – GMM with area and perimeter as features, with and without PCA reduction.

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.

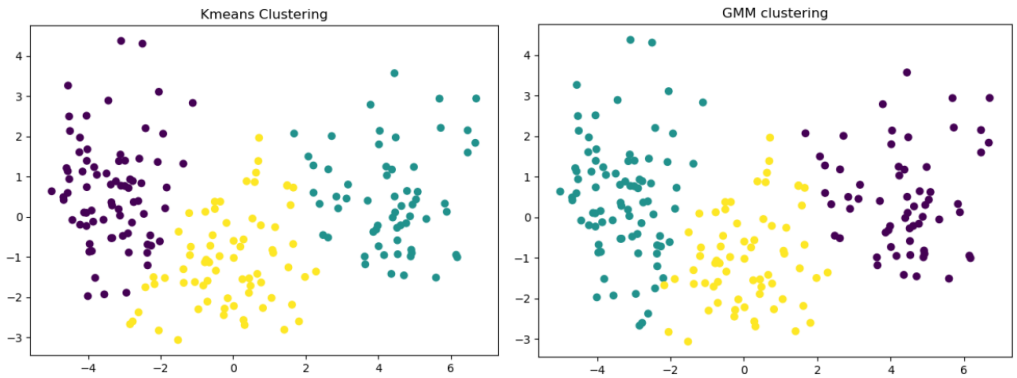

Fig. 12 and 13 – Our best results of Kmeans and GMM on the seed dataset.

References:

Scikit-learn: Machine Learning in Python

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, Édouard Duchesnay; 12(Oct):2825−2830, 2011.