Assignment 3:
Unsupervised learning with PCA, t-SNE, k-means, AHC and SOM
Neuronal and evolutionary computing
2023 - 2024

Pawel Puzdrowski

2023-12-17

Contents

0.1 Part 1 2
0.1.1 Dataset1 - provided from course site 2
0.1.2 Dataset from internet - Dry bean dataset 2
0.2 Part 2 3
0.2.1 Implementation of methods 3
0.2.2 PCA 3
0.2.3 t-SNE 6
0.2.4 k-means 15
0.2.5 AHC 26
0.2.6 SOM 28
0.2.7 Conclusion 30

0.1 Part 1

0.1.1 Dataset1 - provided from course site

No changes to column values were made to the first dataset, only z-score normalization was done on the columns except the last one (class column) with simple jupyter notebook ”dataCleaningNotebooks.ipynb” which is situated in directory ”DataCleaning”. The file for this dataset is in directory ”datasets” and the name is ”A3-dataset1Modified.csv”.

0.1.2 Dataset from internet - Dry bean dataset

The second dataset was obtained from the kaggle.com website specifically [1]. This website consists of 16 variables, one class column and 2501 features. The database represents data that was obtained from a computer vision system where seven different dry beans were examined by a high resolution camera and over 13 000 images were taken. The website provides two csv files, one for testing and one for training, only the training csv file is used because the criterium for this task that the dataset needs to have at least 200 features and the training csv file has 2500 features which is enough.

This dataset class column values are categorical values (text) which represents different beans. There exist seven different beans names and they were changed for numbers. How the bean names were changes in the class column is shown in table 1. Z-score normalization was applied to every column because the columns had a lot of outliers, the last column, class, was not normalized. The modification and normalization was done with script ”dataCleaningNotebooks.ipynb” which is in directory ”DataCleaning”.

Old value	New value
SEKER	0
BARBUNYA	1
BOMBAY	2
CALI	3
DERMASON	4
HOROZ	5
SIRA	6

Table 1: Bean names change

0.2 Part 2

0.2.1 Implementation of methods

The code for the Implementation of methods is on this github repository [2].

0.2.2 PCA

Implementation of PCA is situated in directory "PCA" where a jupyter notebook file exist, "PCA.ipynb".

Scatter plot and scree plot for dataset 1 is visualized in figure 1 and 2 respectively. Plot for dataset2 is visualized in figure 3 and 4.

The PCA doesn't do a good job in classifying the different classes in dataset 1. The scatter plot is spread over the graph and it's hard to see the clusters. Scatter plot for dataset 2 on the other hand is much better and the clusters are visible.

Dataset1

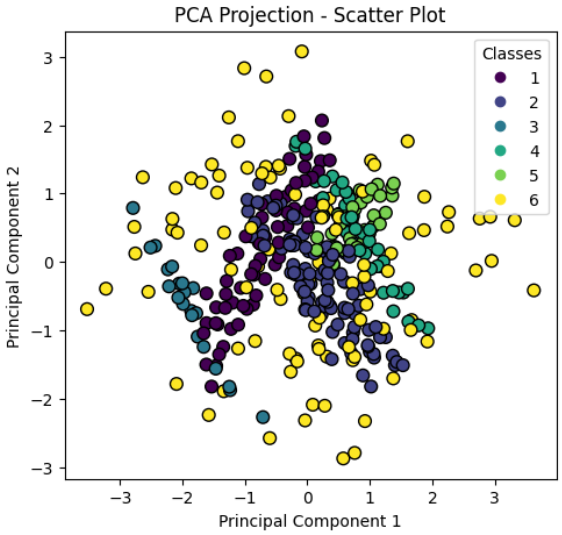


Figure 1: Dataset1 scatter plot

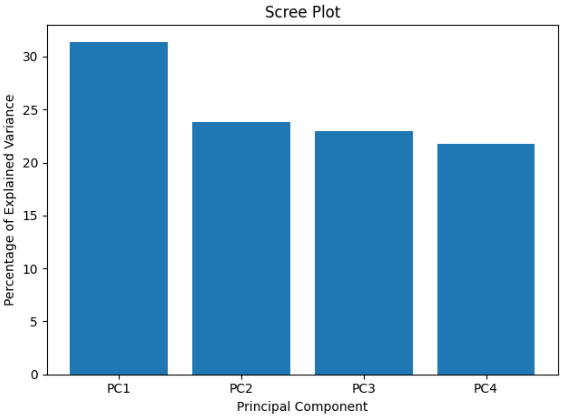


Figure 2: Dataset1 scree plot

Dataset2

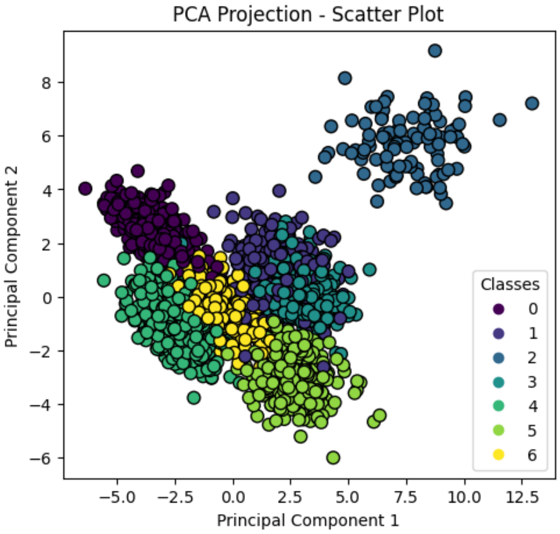


Figure 3: Dataset2 scatter plot

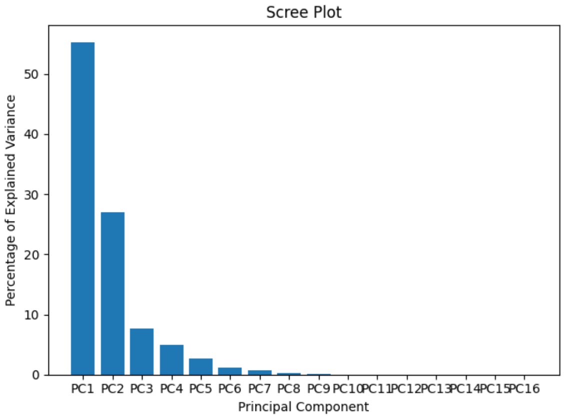


Figure 4: Dataset2 scree plot

0.2.3 t-SNE

Documentation for t-SNE in python was accessed by the official scikit learn website [4]. According to the documentation the important parameters that can be tuned are following:

- perplexity - values should be between 5 and 50. Is connected to number of nearest neighbours, for bigger datasets this parameter should be bigger.
- learning_rate - How fast the model classifies the data, the value for this parameter is given by: $\frac{\frac{Numberofsamples}{early\_exaggeration}}{4}$
- n_iter - number of iteration

Other parameters in t-SNE have either value "None" or have no impact on the classification if the parameter changes. These are the other paramters:

- early_exaggeration - value: the program changes this parameter auomatically. Defines how much space will be between clusters
- n_components - value: default=2
  sets the dimension of embedded space, the task description says that the t-SNE projection should be found in 2D.
- n_iter_without_progress - value: default=300
  Maximum number of iterations without progress. Decreasing and increasing the value doesn't affect the output plot.
- min_grad_norm - value: default=1e-7
  Decreasing and increasing the value doesn't affect the output plot.
- metric - default='euclidean'
  Method to calculate the distance between points, other methods have no change in the output plot.
- metric_params - value: default=None
  Additional keyword arguments for the metric function. Hard to find other specification for this parameter
- init - value: default="pca"
  Other value for this paramter is "random" and it spreads out the clusters and changes the value so the output plot is not better.

- verbose - value: 1
  If set to 1, gives more information what happens at every step of the iterations.

- random_state - value: default=None
  Randomize the data, doesn't have an effect on the output plot.

- method - value: default='barnes_hut'
  Gradient calculation algorithm. The other option for this parameter is "exact", the program is slower and doesn't show any improvement in the output plot.

- angle - value: default=0.5
  According to the documentation, this variable doesn't have big effect on the output.

- n_jobs - value: default=None
  Parallelization of the program. Not relevant for this assignment.

Dataset 1

Dataset1	Test 1	Test 2	Test 3	Test 4	Test 5	optimal values
Perplexity	20	5	5	5	20	5
Learning_rate	2	30	2	5	30	2
n_iter	1000	1000	1500	500	1000	1000

Table 2: Dataset 1 parameter values

The different values for t-SNE parameters are shown in table 2, five columns are for test values and one column for optimal values for dataset 1. The output plot from this tests are visualized in the following figures:

- Increased perplexity - figure 5b
- Increased learning rate - figure 5a
- Increased and decreased number of iterations - figure 6
- Increased both learning rate and perplexity - figure 7

The output plot from the optimal parameter values is shown in figure 8. Perplexity was chosen to be the lowest value 5, because the dataset 1 is a small database and according to documentation the value for this parameter should be small with small datasets. Learning rate value is obtain by the following equation:

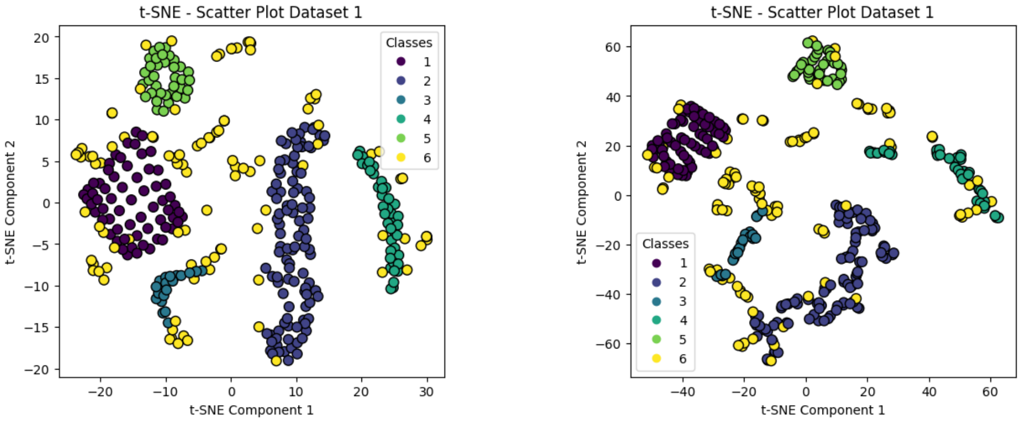
$$\frac{\frac{Number of samples}{early\_exaggeration}}{4}$$

early_exaggeration is chosen automatically by the program and the value was 61 and the final learning rate was calculated to be:

$$\frac{\frac{361}{61}}{4} \approx 2$$

Number of iteration was set to 1000, increasing or decreasing it more didn't give better output, according to figure 6.

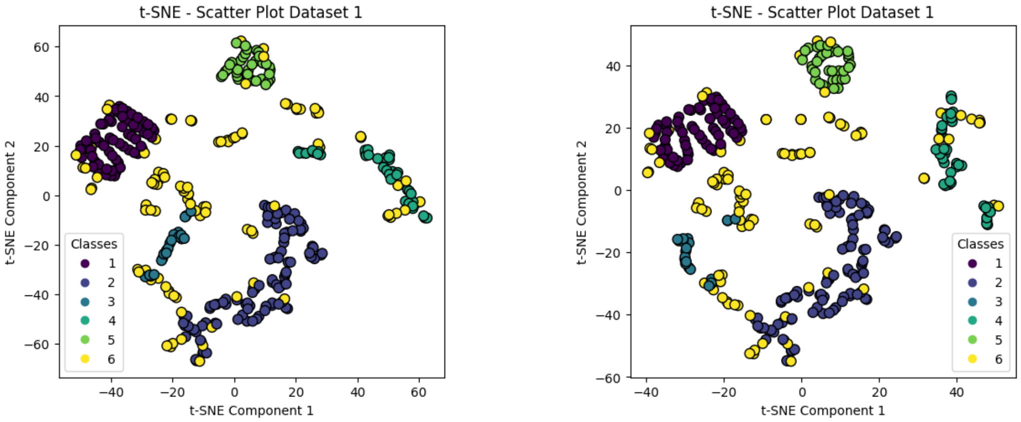
The t-SNE program seems to do a pretty good job in clustering almost all data points except the class 6 datapoints. Wasn't able to find parameter values that could handle all classes including the sixth one.



(a) Increased learning rate

(b) Increased perplexity

Figure 5: Output plot for increased learning rate and perplexity for dataset 1



(a) Increased number of iterations

(b) Decreased number of iterations

Figure 6: Increased and decreased number of iterations output plot

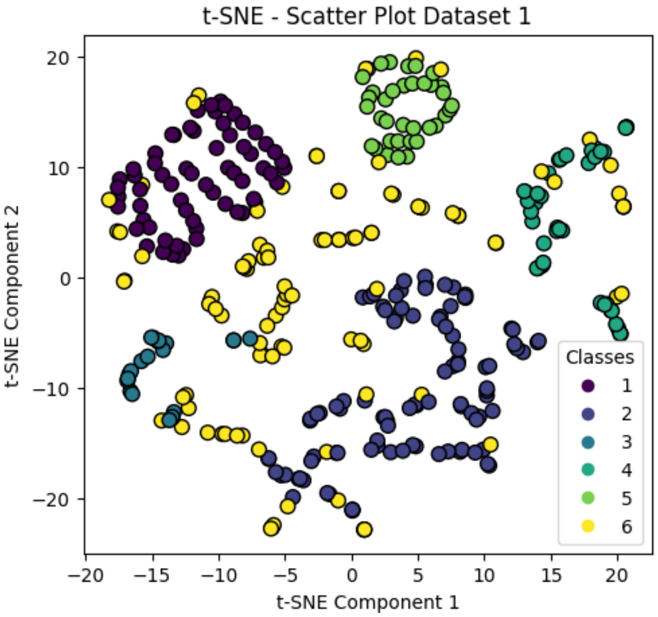


Figure 7: Dataset 1 increased learning rate and perplexity

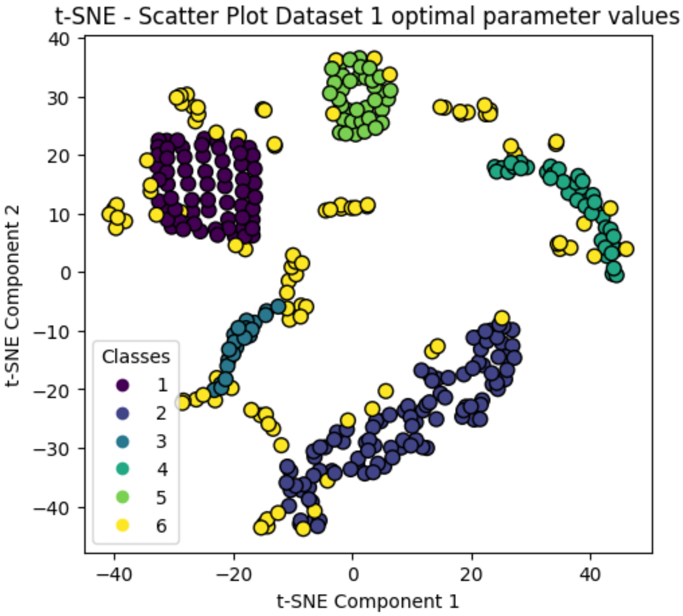


Figure 8: Dataset 1 optimal parameter values plot

Dataset 2 - dry beans
The different parameter values that were tested are shown in table 3.

All the parameter values gave almost the same output plot of the different classes, the best one was the one with perplexity 30, learning rate 10 and number of iterations 1000, which is shown in figure 9 but can even say that the perplexity: 50 was good as well. The worst was the one with perplexity 20 because the clusters are more spread.

As shown in figure 10, number of iteration has a slightly influence on the clusterring. Figure 11 shows different tests with different values for perplexity and for this dataset higher value for perplexity seems to be an advantage. Learning rate on the other hand, as shown in figure 12, doesn't have big effect on the t-SNE output.

The optimal values where obtained as follows:
Learning rate, as in the previous dataset, was obtained by this equation:

$\frac{\frac{Number of samples}{early\_exaggeration}}{4}$

The value for optimal learning was calculated to be:

$\frac{\frac{2500}{68}}{4} \approx 10$

The perplexity was set to be higher than for the dataset 1 because dataset 2 is larger and perplexity should be larger when database increases. Number of iterations was set to 1000 because increasing more didn't give any better output plot.

Dataset1	Test 1	Test 2	Test 3	Test 4	Test 5	Test 6	Test 7	Optimal values
Perplexity	20	35	50	30	30	30	30	30
Learning_rate	10	10	10	30	100	10	10	10
n_iter	1000	1000	1000	1000	1000	500	1500	1000

Table 3: Dataset 2 parameter values

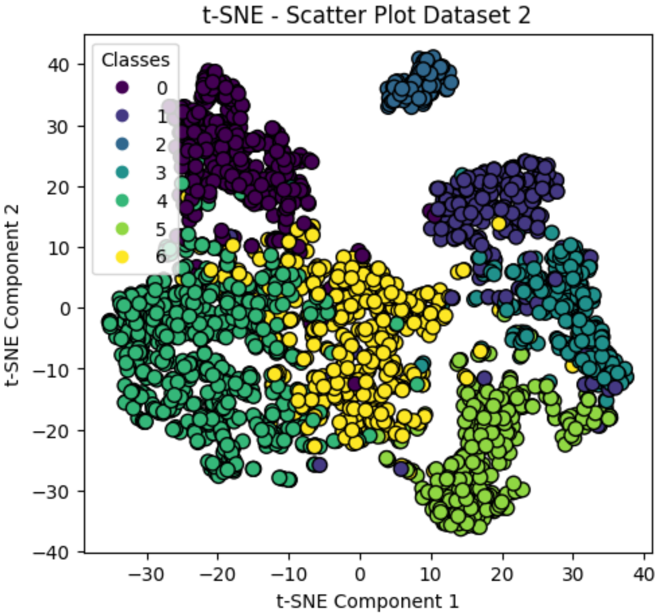
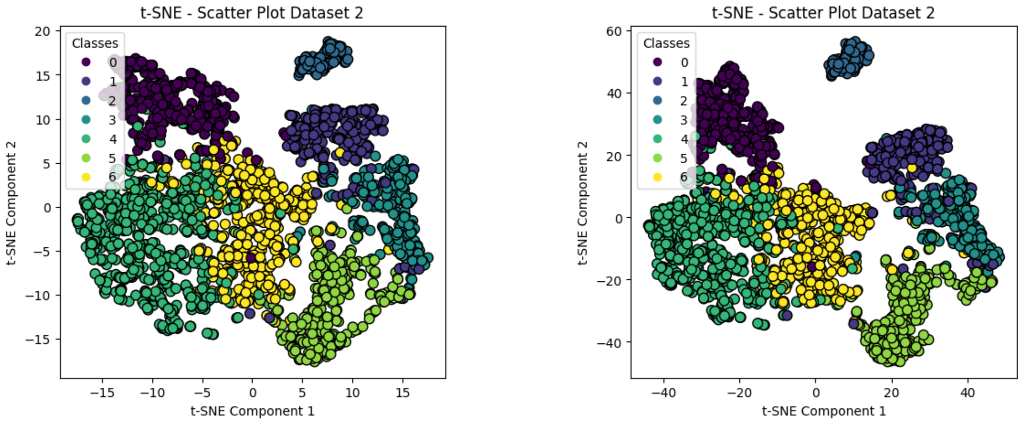


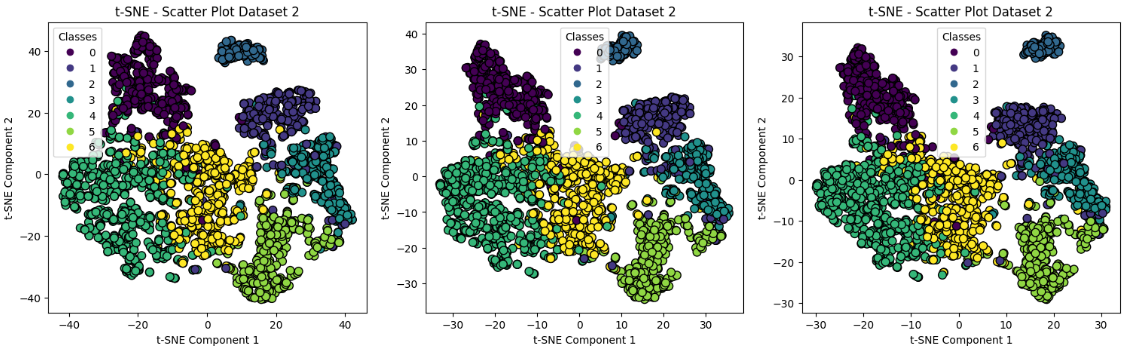
Figure 9: Dataset 2 optimal values: perplexity: 30, learningRate: 10, nIter: 1000



(a) Number of iteration: 500

(b) Number of iteration: 1500

Figure 10: Database 2: Different number of iterations with perplexity: 30, learningRate: 10

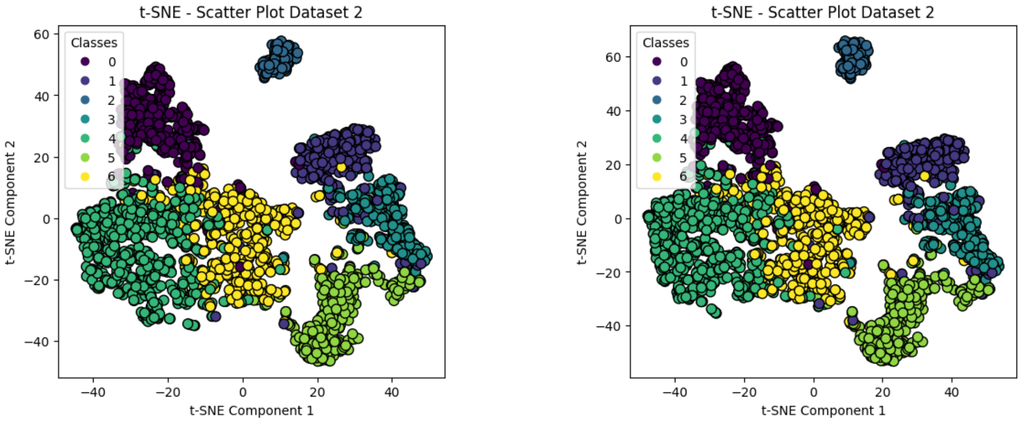


(a) Perplexity = 20

(b) Perplexity: 35

(c) Perplexity: 50

Figure 11: Database 2: Different perplexities, learningRate: 10, nr_iter = 1000



(a) learningRate: 30

(b) learningRate: 100

Figure 12: Database 2: Different learningRate, perplexity:30, nr_iter = 1000

0.2.4 k-means

Dataset 1:

Script for k-means, k-means.ipynb, algorithm is situaited in directory ”K-means”.

To get the optimal values for k, elbow method was used as describe in this source [3]. The optimal value for k is 4 for dataset 1 because according to the elbow method plot in figure 17, the ”elbow” appears when k = 4. Because the datasets has more than two features, the datasets were dimensionally reduced with PCA.

As an additional evaluation of the k-means performance ARI (Adjusted Rand Index) and NMI (Normalized Mutual Information) scores was used. These scores ranges between 0 and 1 where 0 means worst classification and 1 means perfect classification. The ARI and NMI scores are shown in table 4.

For dataset 1 this set of k values was used: 2, 3, 4, 5, 6. The output from the program is visualized in the following plots:

- k = 2,3 scatter plot: figure 13, confusion matrix plot: figure 15
- k = 4,5,6 scatter plot: figure 14, confusion matrix plot: figure 16

K-means method is not doing a good job in clustering the data. Scatter plots for all k values contains very spread data and the ARI and NMI values doesn’t improve when k is increased. Confusion matrix is showing a lot of values that are not classified correctly, the diagonal of every confusion matrix has small values which means that the model is not able to classify the data correctly.

K	ARI	NMI
2	0.196	0.345
3	0.280	0.416
4	0.225	0.408
5	0.228	0.406
6	0.230	0.408

Table 4: Dataset 1 k-means ARI and NMI values

Dataset 2:

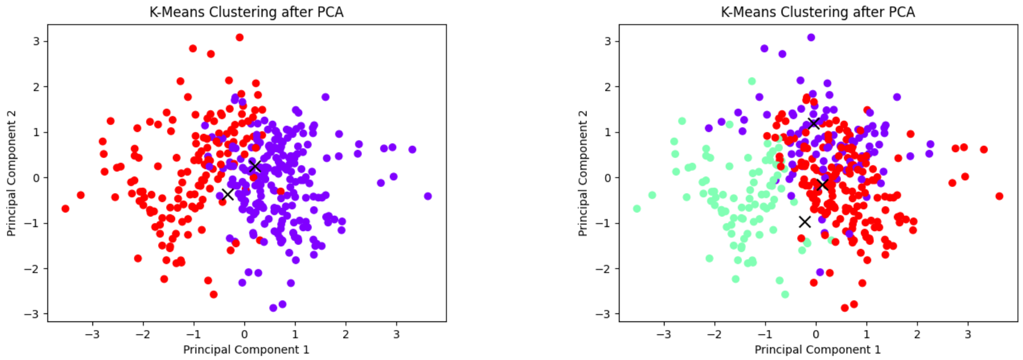
For dataset 2 this set of k values was used: 2, 3, 4, 5, 6, 7, 8. The output is visualized in the following plots:

- k = 2,3 scatter plot: figure 18, confusion matrix plot: figure 21
- k = 4,5,6 scatter plot: figure 19, confusion matrix plot: figure 22
- k = 7,8 scatter plot: figure 20, confusion matrix plot: figure 23
- Elbow method plot 24

For the dataset 2, k-means is doing a better job than for dataset 1. The plot from elbow method says that the optimal value for k is 6 and it is even visible on the ARI and NMI score in table 5 that the ARI and NMI scores increase until k is equal to 6 and further increase, decreases the scores. Scatter plot for k = 6 shows clear clusters and confusion matrix for the same k value has a decent amount of values on the diagonal cells.

K	ARI	NMI
2	0.0352	0.1695
3	0.3030	0.4987
4	0.3945	0.5990
5	0.5597	0.6960
6	0.6849	0.7327
7	0.6661	0.7113
8	0.5612	0.6646

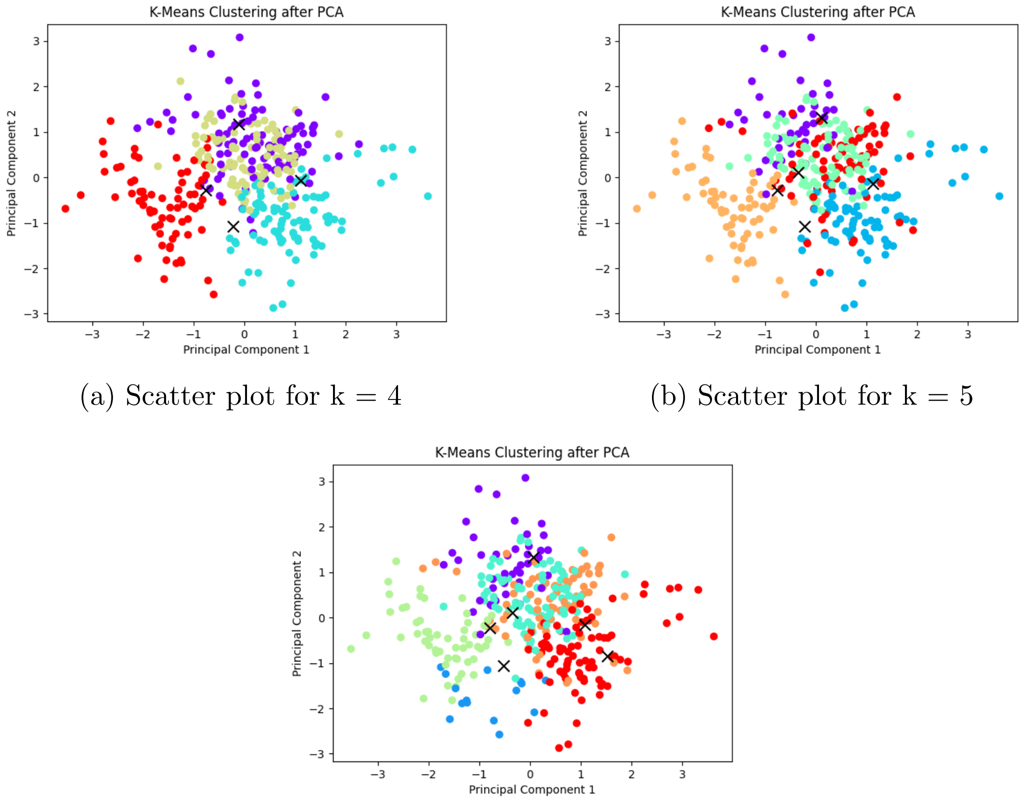
Table 5: Dataset 2: ARI and NMI Scores for Different Values of K



(a) Scatter plot for k = 2

(b) Scatter plot for k = 3

Figure 13: Dataset 1: Scatter plot with value k = 2,3

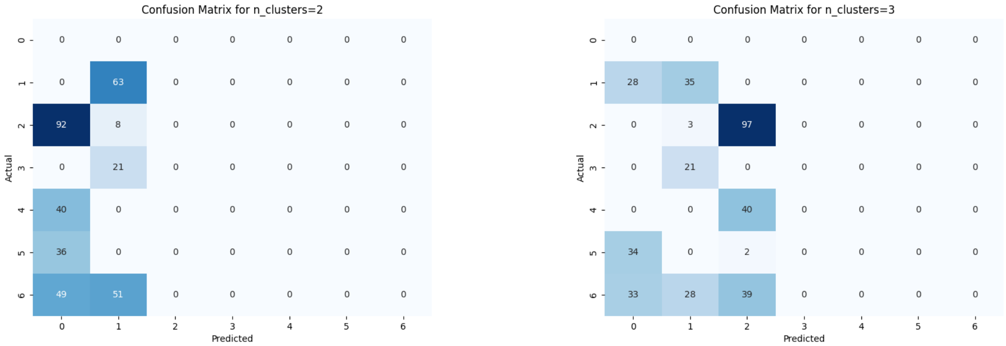


(a) Scatter plot for k = 4

(b) Scatter plot for k = 5

(c) Scatter plot for k = 6

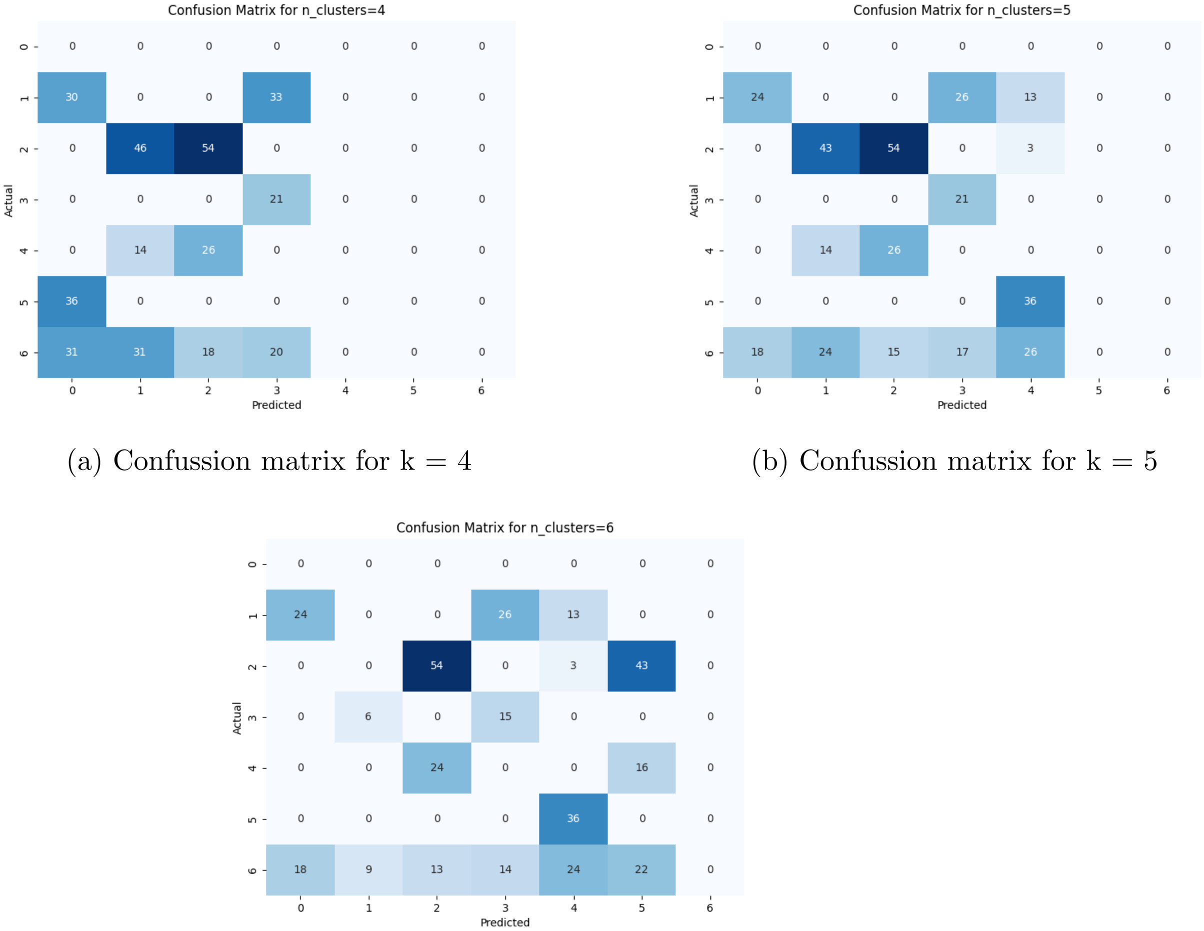
Figure 14: Dataset 1: Scatter plot with value k = 4, 5, 6



(a) Confussion matrix for k = 2

(b) Confussion matrix for k = 2

Figure 15: Dataset 1: Scatter plot with value k = 2,3



(a) Confussion matrix for k = 4

(b) Confussion matrix for k = 5

(c) Confussion matrix for k = 6

Figure 16: Dataset 1: Scatter plot with value k = 4, 5, 6

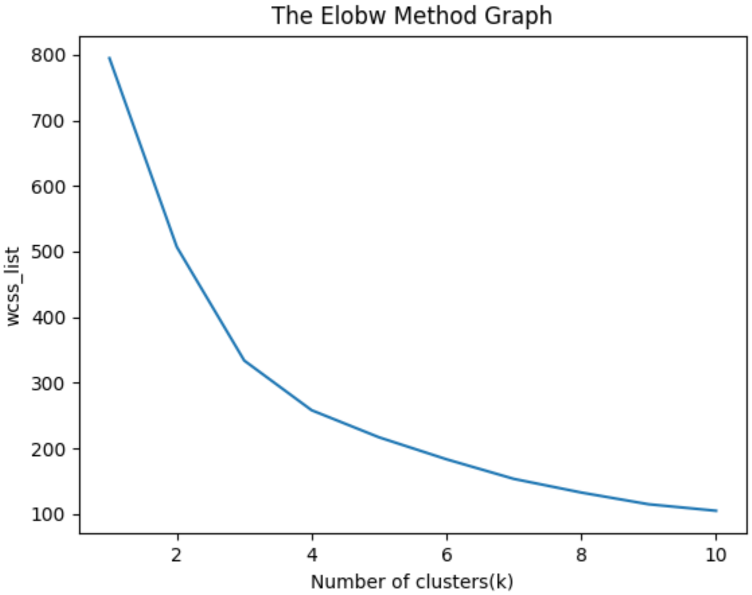
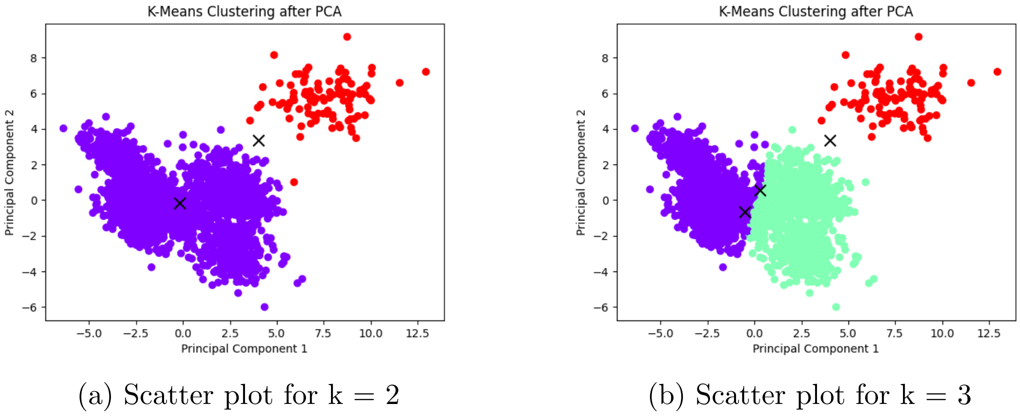


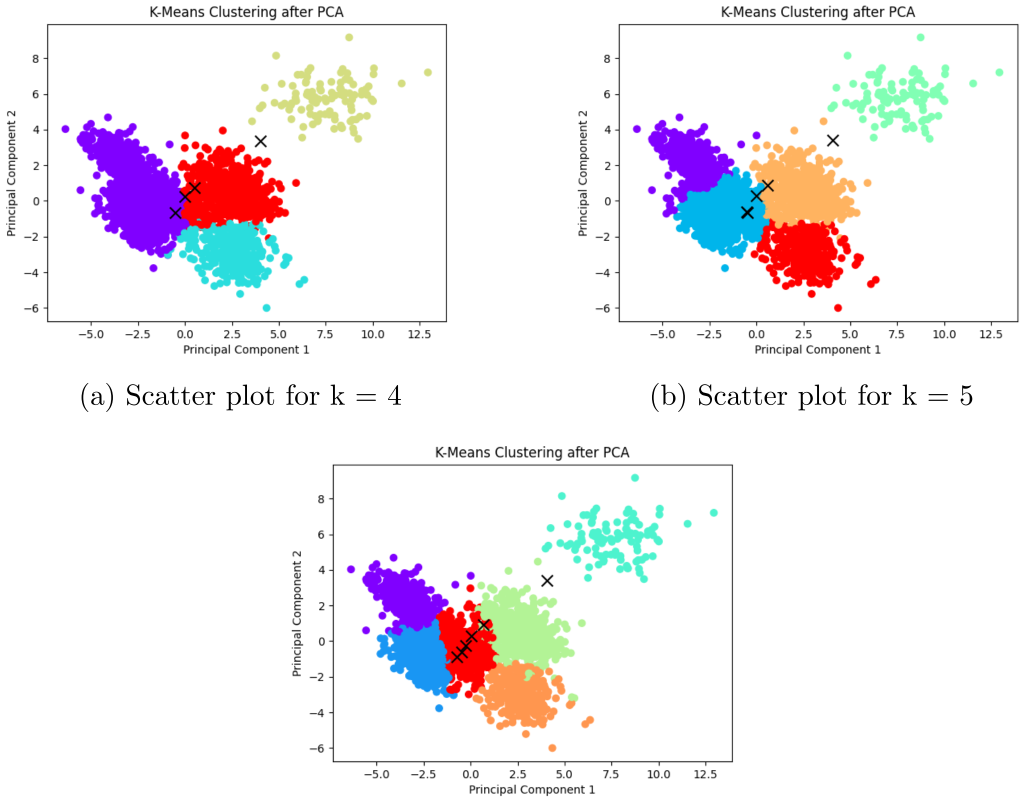
Figure 17: Dataset 1 ElbowMethod output plot



(a) Scatter plot for k = 2

(b) Scatter plot for k = 3

Figure 18: Dataset 2: Scatter plot with value k = 2,3

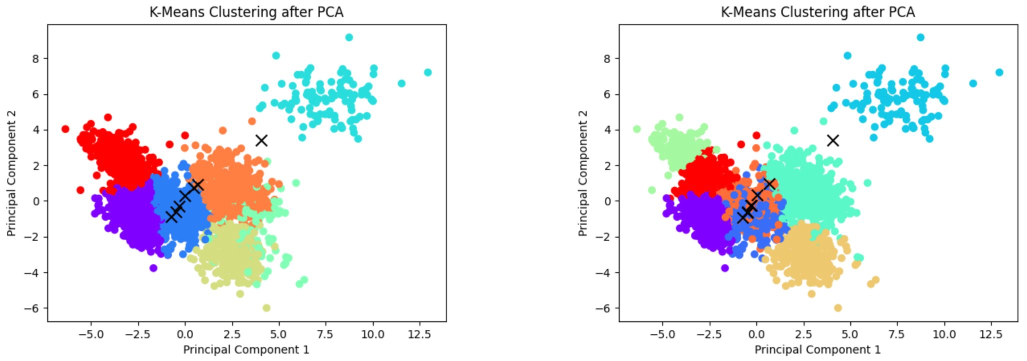


(a) Scatter plot for k = 4

(b) Scatter plot for k = 5

(c) Scatter plot for k = 6

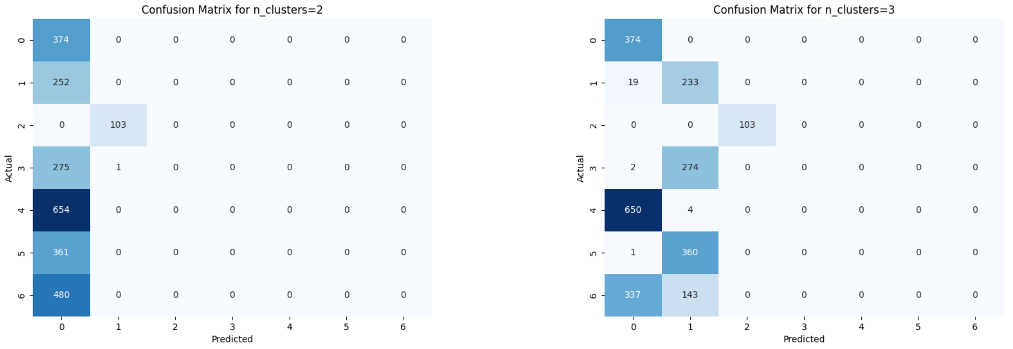
Figure 19: Dataset 2: Scatter plot with value k = 4, 5, 6



(a) Scatter plot for k = 7

(b) Scatter plot for k = 8

Figure 20: Dataset 2: Scatter plot with value k = 7,8



Confusion Matrix for n_clusters=2

Actual \ Predicted	0	1	2	3	4	5	6
0	374	0	0	0	0	0	0
1	252	0	0	0	0	0	0
2	0	103	0	0	0	0	0
3	275	1	0	0	0	0	0
4	654	0	0	0	0	0	0
5	361	0	0	0	0	0	0
6	480	0	0	0	0	0	0

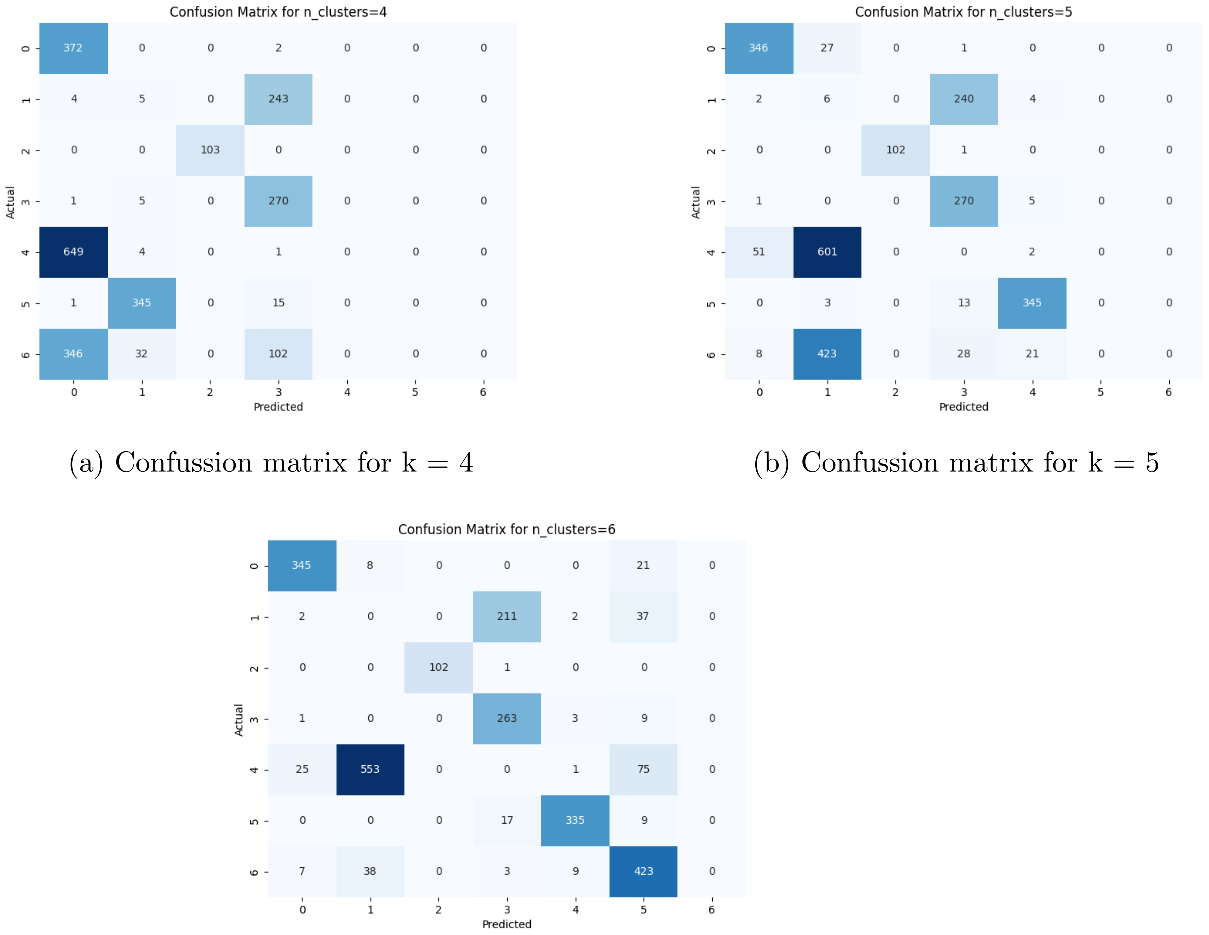
Confusion Matrix for n_clusters=3

Actual \ Predicted	0	1	2	3	4	5	6
0	374	0	0	0	0	0	0
1	19	233	0	0	0	0	0
2	0	0	103	0	0	0	0
3	2	274	0	0	0	0	0
4	650	4	0	0	0	0	0
5	1	360	0	0	0	0	0
6	337	143	0	0	0	0	0

(a) Confussion matrix for k = 2

(b) Confussion matrix for k = 2

Figure 21: Dataset 2: Scatter plot with value k = 2,3



Confusion Matrix for n_clusters=4

	0	1	2	3	4	5	6
0	372	0	0	2	0	0	0
1	4	5	0	243	0	0	0
2	0	0	103	0	0	0	0
3	1	5	0	270	0	0	0
4	649	4	0	1	0	0	0
5	1	345	0	15	0	0	0
6	346	32	0	102	0	0	0

(a) Confussion matrix for k = 4

Confusion Matrix for n_clusters=5

	0	1	2	3	4	5	6
0	346	27	0	1	0	0	0
1	2	6	0	240	4	0	0
2	0	0	102	1	0	0	0
3	1	0	0	270	5	0	0
4	51	601	0	0	2	0	0
5	0	3	0	13	345	0	0
6	8	423	0	28	21	0	0

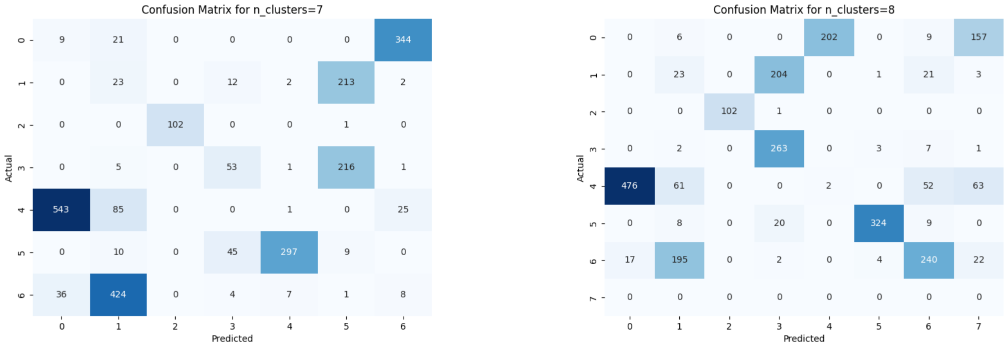
(b) Confussion matrix for k = 5

Confusion Matrix for n_clusters=6

	0	1	2	3	4	5	6
0	345	8	0	0	0	21	0
1	2	0	0	211	2	37	0
2	0	0	102	1	0	0	0
3	1	0	0	263	3	9	0
4	25	553	0	0	1	75	0
5	0	0	0	17	335	9	0
6	7	38	0	3	9	423	0

(c) Confussion matrix for k = 6

Figure 22: Dataset 2: Scatter plot with value k = 4, 5, 6



(a) Confussion matrix for k = 7

(b) Confussion matrix for k = 8

Figure 23: Dataset 2: Scatter plot with value k = 7,8

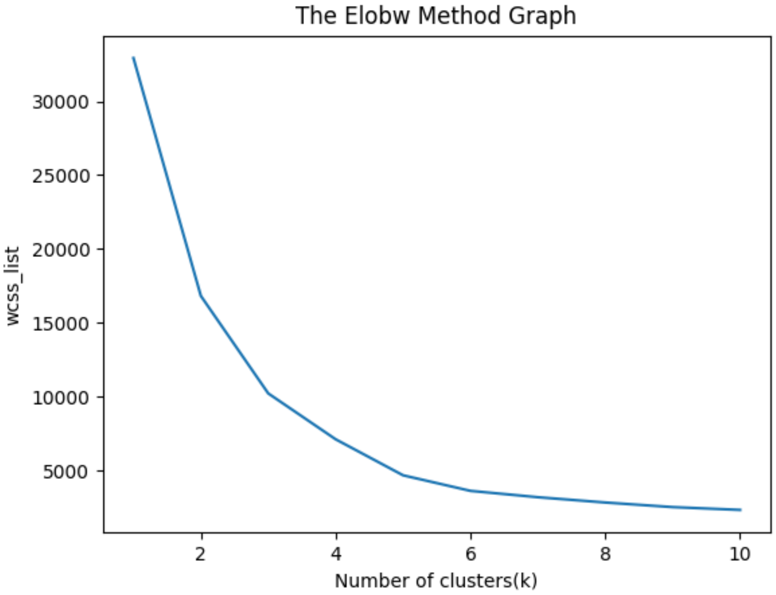


Figure 24: Dataset 2 ElbowMethod output plot

0.2.5 AHC

Script for this method is situated in directory AHC and the name of the script is AHC.ipynb.

The dendrograms are shown in figure 25 and figure 26. Distance threshold was set to visualize the different clusters that are formed from the AHC method. For dataset 1 distance threshold was set to 5 and 3 for complete linkange (figure 25a) and UPGMA linkage (figure 25b) respectively. For dataset 2 distance threshold was set to 11 and 4.1 for complete linkange (figure 26a) and UPGMA linkage (figure 26b) respectively.

Class	1	2	3	4	5	6
Data count	63	100	21	40	36	100

Table 6: Dataset 1 Amount of points for each class

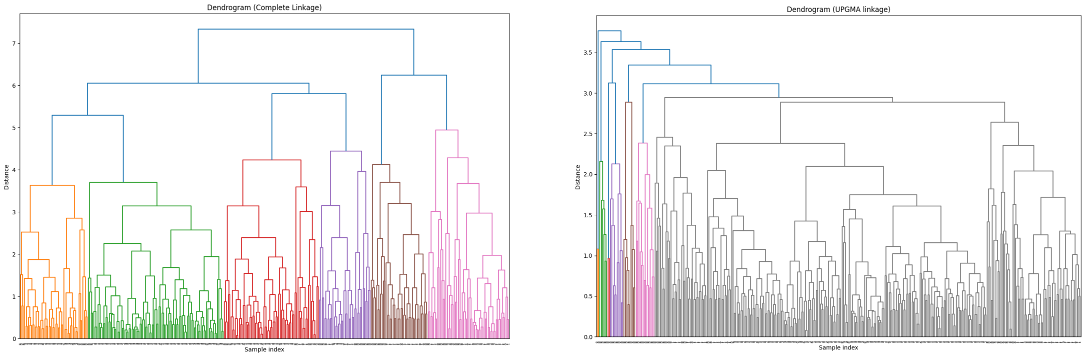
Class	0	1	2	3	4	5	6
Data count	374	252	103	276	654	361	480

Table 7: Dataset 2 Amount of points for each class

By visual inspection of the dendrograms, for dataset 1, complete linkage is much better at representing the amount of datapoints in each clusters. Looking at table 6 where it shows how many datapoints belongs to each class and comparing the complete linkage figure 25a, the dataset is much better represented than using UPGMA. UPGMA is just assiging a big chunk of the datapoints to one class.

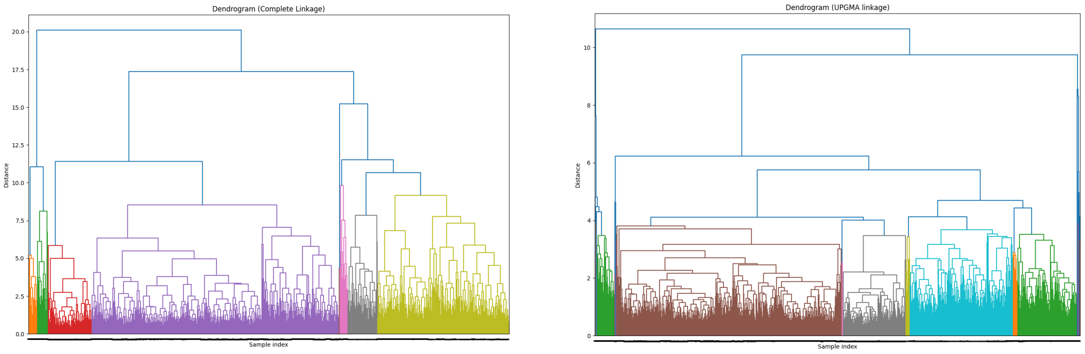
The same conclusion can be made for dataset 2. In figure 26a where complete linkage dendrogram is shown and table 7 which shows how many datapoints belongs to each class, the dataset is much better represented than using UPGMA. The purple cluster can be the datapoints that belongs class 4, which contains the most datapoints.

This method is slow, takes a lot of time to create the dendrograms espacially the complete linkage for dataset 2, takes approximatly 1 minute. This method seems to be more expensive resources wise than the other methods.



(a) Dataset 1 complete linkage dendrogram (b) Dataset 1 UPGMA linkage dendrogram

Figure 25: Dataset 1 dendrograms



(a) Dataset 2 complete linkage dendrogram (b) Dataset 2 UPGMA linkage dendrogram

Figure 26: Dataset 2 dendrograms

0.2.6 SOM

The different parameters have the following influence on the output:

- grid size - the grid size needs to be atleast 100 according to assignment description, but some source recommends grid size $5 * sqrt(N)$ where N is the number of samples.

- sigma - defines radius of neighborhood function. If to small the model will not fully represent the multi dimensional data onto the two dimensional plot. If too big then the model will overfit the data.

- learning rate - how much weights are updated during each iteration.

- Number of iterations - For dataset 1, 10 000 iterations was enough and for dataset 2, 20 000 iterations was needed for the model to visualize the clusters.

With this method only the best plot is visualized. All other output can be accessed on the github repo in directory SOM/Output.

Dataset 1

After numerous tests on the parameters, the best plot was obtained with test 1. The different values of parameters that were tested for dataset 1 are visualized in table 8. The plot is visualized in figure 27a, the darker colours shows where the clusters are identified and different figures in the plot shows different clusters. The SOM method doesn't quite identify many clusters in dataset 1. Even after playing with the parameter values it's hard for the model to identify the clusters.

Dataset 2

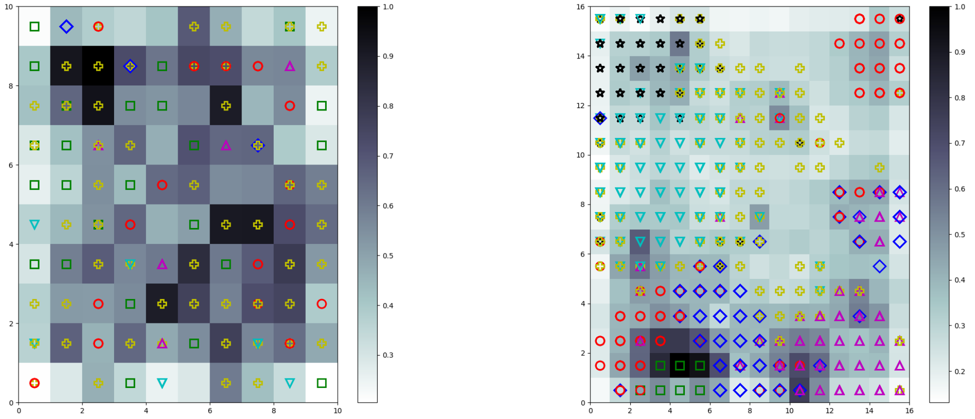
The different tests that were made are listed in table 9. There was more tests done but there is no need to include all of them. All the tests are included in the directory SOM/Output on the github repo. Surprisingly this method had difficulty in clustering the dataset 2. Can be the case that the parameters wasn't adjusted properly. The best plot was obtained with test 8 and the output plot is visualized in figure 27b.

Test nr:	1	2	3	4	5	6	7	8	9	10
Grid size	100	100	100	100	100	100	100	100	144	144
Sigma	0.2	0.2	0.5	0.5	0.5	0.5	0.5	0.7	0.2	0.5
Learning rate	0.5	0.5	0.2	0.5	0.5	0.7	0.7	0.5	0.5	0.5
Nr of iterations	10000	15000	10000	10000	15000	10000	15000	10000	15000	10000

Table 8: Dataset 1: parameter values that were tested

Test nr:	1	2	3	4	5	6	7	8	9	10
Grid size	256	256	256	256	256	256	256	256	144	144
Sigma	0.9	0.9	0.9	0.8	0.8	0.8	0.7	0.7	0.2	0.5
Learning rate	0.4	0.2	0.1	0.6	0.2	0.2	0.4	0.2	0.5	0.2
Nr of iterations	10000	10000	10000	10000	15000	10000	15000	20000	15000	10000

Table 9: Dataset 2: parameter values that were tested



(a) Dataset 1 best output plot

(b) Dataset 2 best output plot

Figure 27: Dataset 1 and 2 best output plots

0.2.7 Conclusion

All the different methods had difficulties to cluster the dataset 1. Visually PCA and AHC did the best job in clustering and classifying the dataset 1. For dataset 2 almost all methods did a very good job in clustering this dataset. The worst clustering for dataset 2 was SOM and the best was k-means and AHC.

Dataset 1's poor classification performance across all methods likely cames from its limited sample size of only 360 instances. In contrast, Dataset 2, with 2500 samples, has more classes as well but the methods still handles the classification effectively. This shows the importance of having an adequate sample size for machine learning. Increasing Dataset 1's sample size could improve its classification accuracy across different methods.

Bibliography

[1] Dry bean classification. https://www.kaggle.com/datasets/gauravduttakiit/dry-bean-classification. Accessed 2023-12-08.

[2] Github repository. https://github.com/Pawel712/NEC-Assignment3.

[3] K-means clustering algorithm. https://www.javatpoint.com/k-means-clustering-algorithm-in-machine-learning. Accessed 2023-12-13.

[4] Sklearn.manifold.tsne. https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html. Accessed 2023-12-11.