FDA
Unsupervised Learning
Report
SS 21

Mohammad Mahdi Fallah
01428941
mahdyfalah@gmail.com

Task 1: PCA

In this task I have implemented a **PCA** function which can be found in the second block as, PCA_impl(X, num_components).
This implementation receives the data sets and the number of components as parameters and returns the resulting PCA matrix.
Further info on the pseudo code can be found via :
https://www.askpython.com/python/examples/principal-component-analysis

Then I have calculated explained variance and explained variance ratio for both my implementation and sklearn PCA which results in the following:

For my Implementation:

Explained variance :
```
[1.07970946e+01 2.03805505e+00 7.34206725e-02
1.26838629e-02
 2.74215844e-03 1.57629129e-03 2.90232256e-05]
```

Explained variance ratio:
```
[8.35326268e-01 1.57675836e-01 5.68025183e-03
9.81297676e-04
 2.12149385e-04 1.21951096e-04 2.24540616e-06]
```

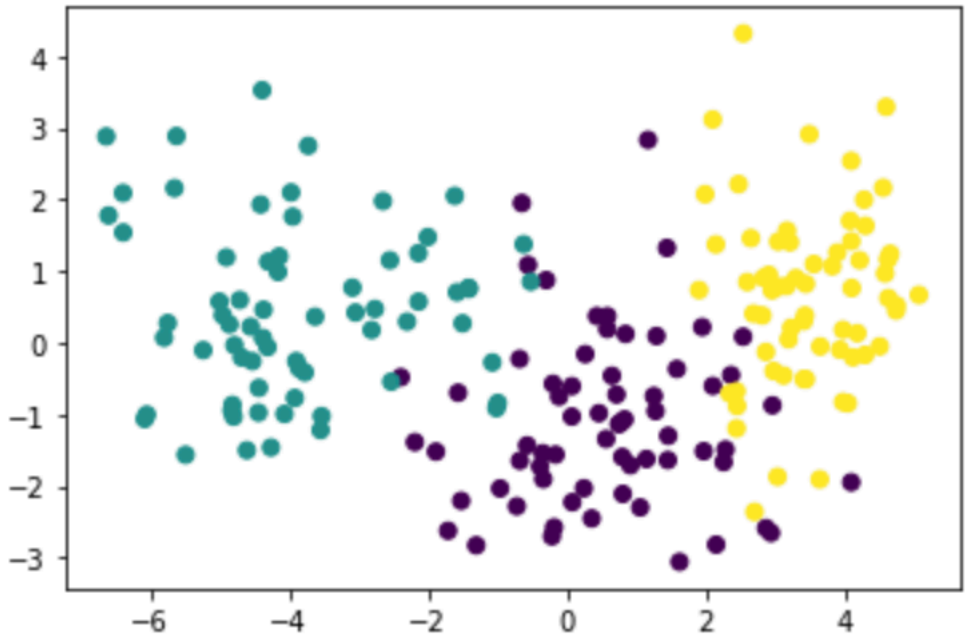
For SKlearn Implementation:

Explained variance :
```
[1.08516254e+01 2.04834826e+00 7.37914840e-02
1.27479228e-02
 2.75600772e-03 1.58425236e-03 2.91698076e-05]
```

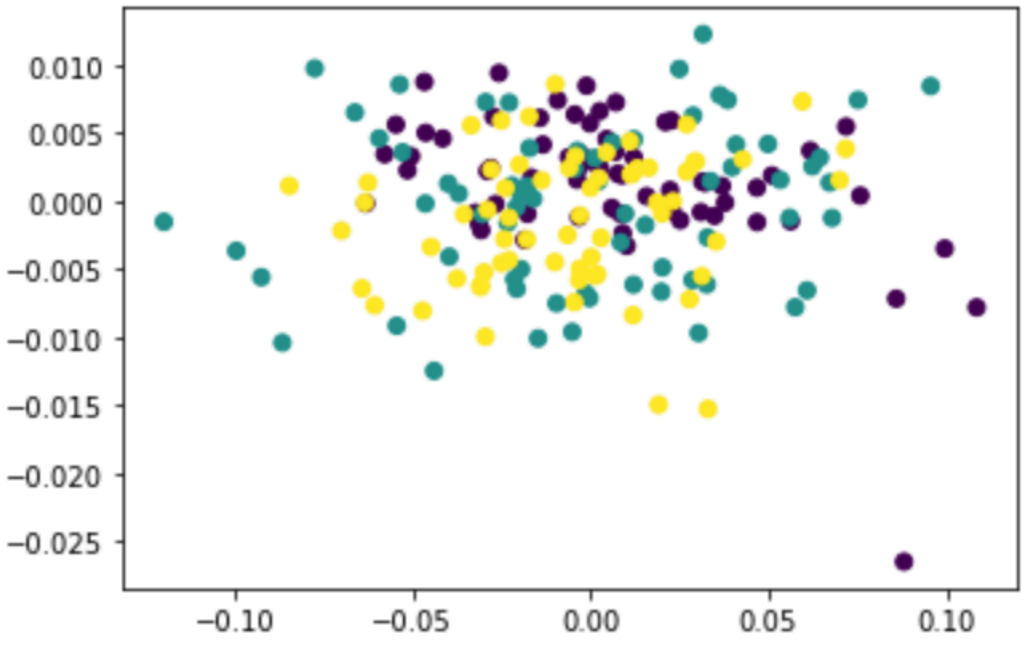
Explained variance ratio:
```
[8.35326268e-01 1.57675836e-01 5.68025183e-03
9.81297676e-04
 2.12149385e-04 1.21951096e-04 2.24540616e-06]
```

And at last scatter plot the two principal components corresponding to the the dimension with the most and least variances can be found, which shows the clear difference:

The 2 highest variances:



The 2 lowest variances:

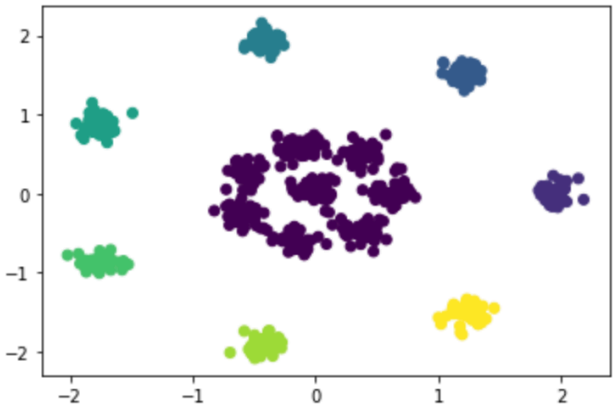


Task 2: Clustering

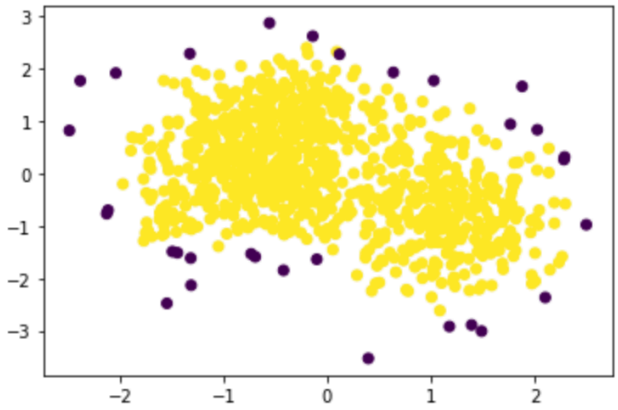
In the first block we import the needed libraries from sklearn such as DBSCAN, KMeans, StandardScaler etc.
Then in the second block we read the data using np.genfromtxt()
Then in the following blocks we calculate the clustered data and then we plot the results:

DBSCAN:

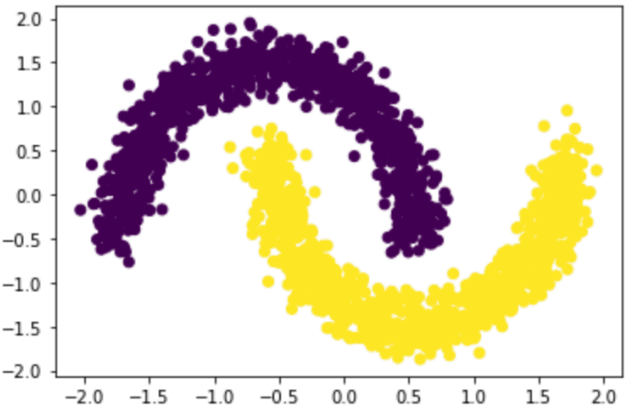
data1



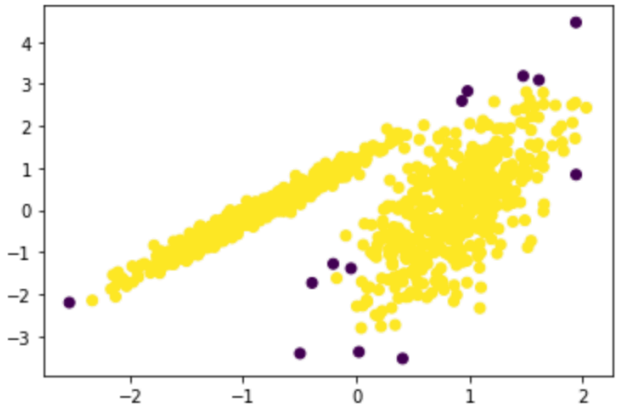
data2



data3

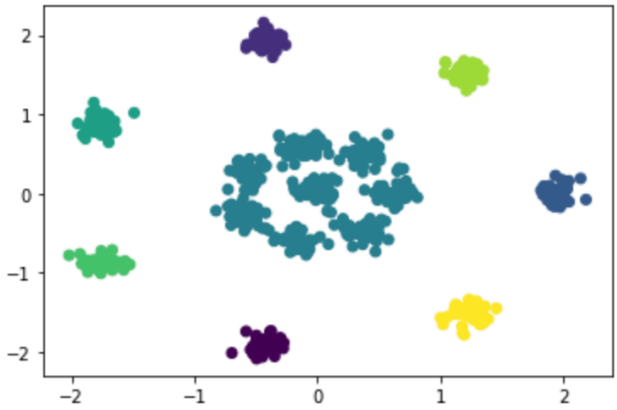


data4

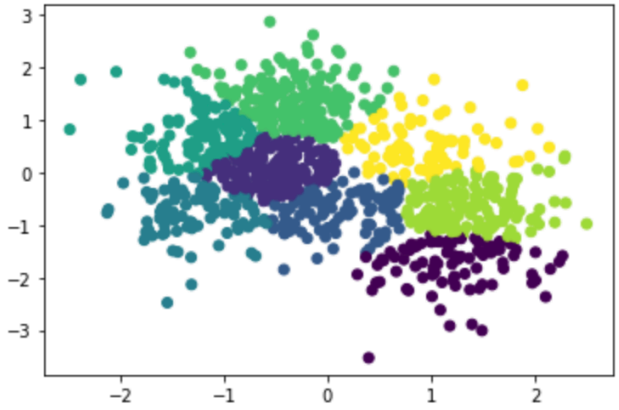


KMeans:

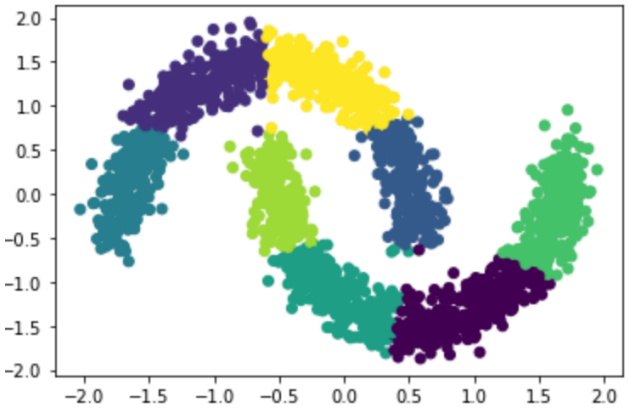
data1



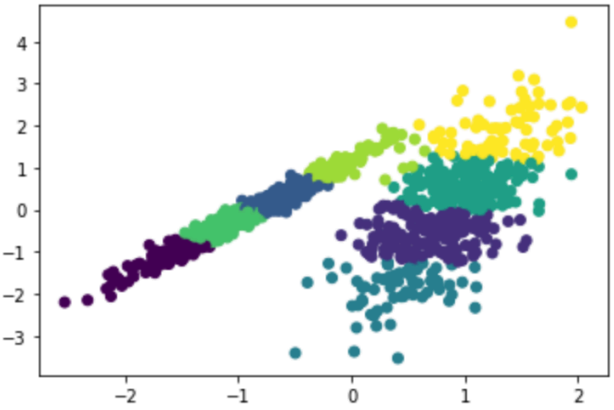
data2



data4

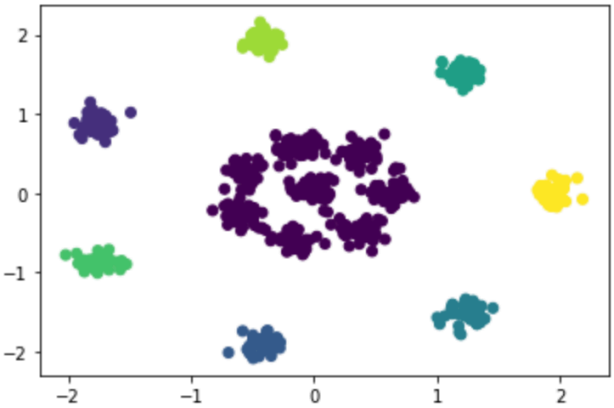


data4

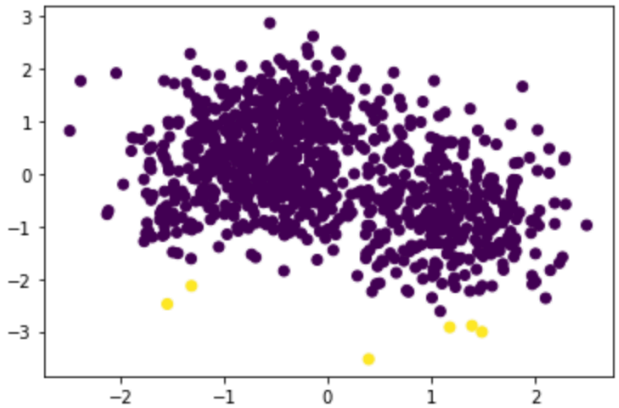


Average Link:

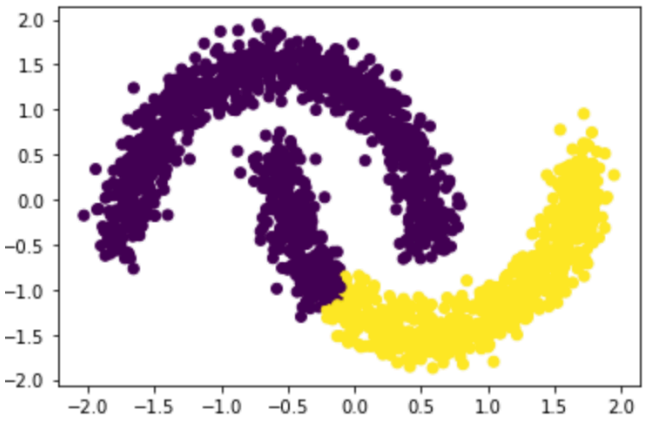
data1



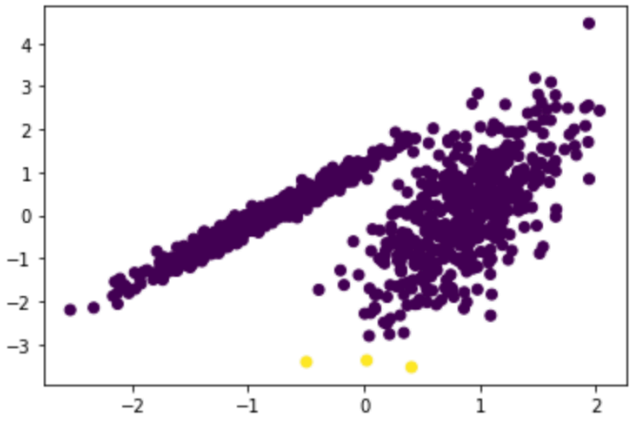
data2



data3

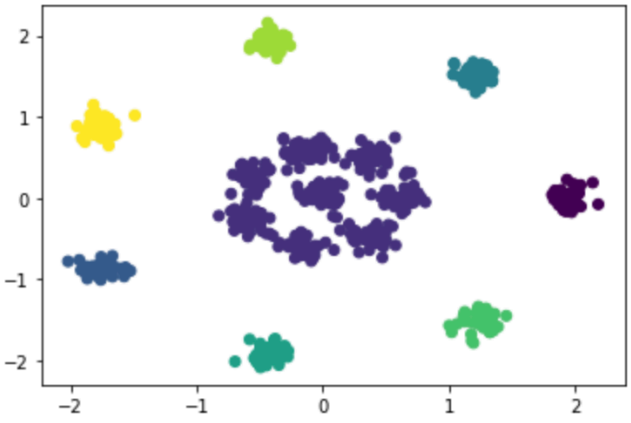


data4

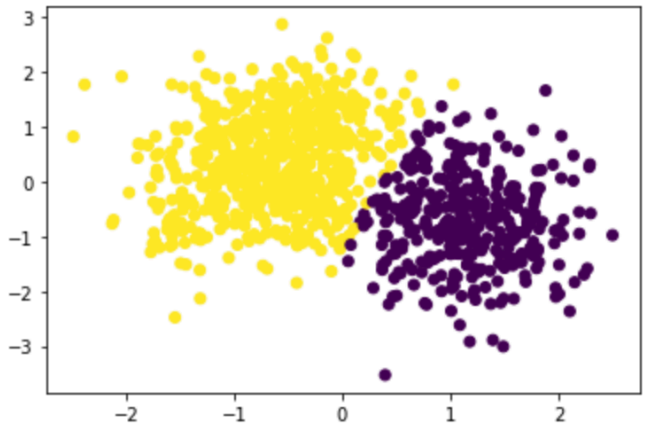


Expectation Maximization:

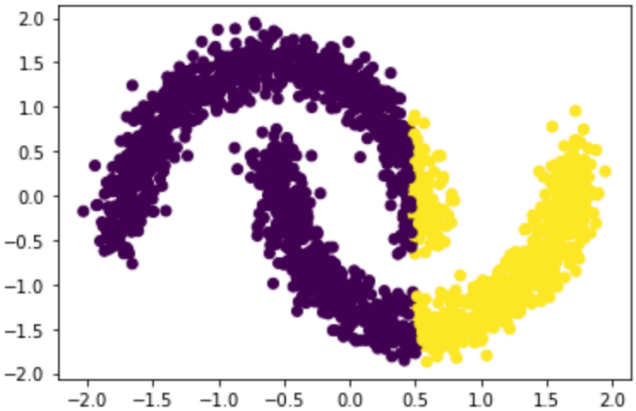
data1



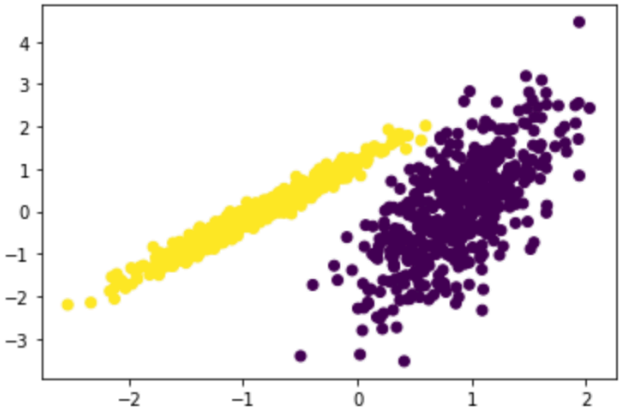
data2



data3



data4



Now we compare the results to get an understanding of which algorithm is better suited for which type of data:
I believe every algorithm has similar results for clusters of data 1
But clearly DBSCAN did the better job for data 3, and Expectation Maximization had the best results for clusters of data 2, and data 4.