FDA - Lab Assignment #2

Christian Wiskott

University of Vienna — January 13, 2021

Task 1: Dimensionality Reduction & PCA

The first task was to implement PCA from scratch in Python. This was accomplished using a custom class called *PCA_custom* and by only utilizing the *numpy* package. The corresponding file is called *lab2_PCA.py* and the only thing it requires is the file *seeds.csv* to be in the same folder as the py-file. The ouput consists of the PCA-plot and the print statements which show the calculated explained variance and their respective ratios for each principal component (PC).

The top part of fig. 1 shows that the most important two principal components separate the data nicely into three distinct groups. The corresponding table 1 showcases the performance metrics for the computed PC's. It can be seen that the first PC explains around 80% of the variance within the data, while the second PC explains roughly 17%. In absolute number that is 10.941 and 2.25, respectively.

This shows, that together, those two PC's explain more than 95% of the variance. For this reason, to reduce the complexity and thus the computational cost without sacrificing to much gain, only those two PC's were chosen thus reducing the dimensionality. Since less than 5% of the variance is unaccounted for, this was deemed acceptable. This is a classic case of the "curse of dimensionality" as there is always a trade-off between performance and computational cost.

Table 1 also compares the results of the custom PCA with the pre-implemented *sklearn* PCA. Since the results match perfectly, the custom implementation can be considered correct.

However when looking at the bottom part of fig. 1, the plotted points of the custom PCA seem to be only identical to the sklearn PCA results, if the whole plot is mirrored along the x-axis. This might be due to a different implementation within sklearn and not necessarily a coding error of the custom PCA. Since the metrics in table 1 are identical, this would support this assumption.

	Expl. Variance	**Expl. Variance Ratio**
(1) Custom PCA		
PC1	10.941	0.801
PC2	2.250	0.165
(2) Sklearn PCA		
PC1	10.941	0.801
PC2	2.250	0.165

Table 1: Comparison of the explained variance and the explained variance ratio of the principal components of the custom- and the sklearn PCA. The results are identical and show that PC1 explains around 80 % of the variance while PC2 explains 16.5 %.

Task 2: Clustering

The given data sets were evaluated using *DBSCAN*, *K-Means*, *Gaussian Mixture* and *Agglomerative Clustering*. Each method was applied to the each data set, resulting in 20 plots, showing the predicted labels and

how well the respective algorithm recreated the true labels. The data was pre-processed by normalizing it via sklearn's *StandardScaler*, as this increased the methods performances. In order to measure performance, the Normalized Mutual Information (NMI) and the (adjusted) Rand index were computed for each method and are presented in table 2. Fig. 2 to fig. 6 show the results of the algorithms on the five data sets.

The parameters for each method were the following:

1. DBSCAN: eps=0.3, min_samples=30. These values were chosen as they maximized the performance across all data sets.
2. K-Means: n_clusters was set to the respective number of clusters of each data set.
3. Gaussian Mixture: n_components was set to the respective number of clusters of each data set.
4. Agglomerative Clustering: n_clusters was set to the respective number of clusters of each data set.

	NMI	**Rand Index**
Dataset 1		
DBSCAN	**0.974**	**0.978**
K-Means	0.829	0.715
Gaussian Mixture	0.948	0.948
Agglomerative Clustering	0.848	0.701
Dataset 2		
DBSCAN	0.527	0.245
K-Means	0.716	0.559
Gaussian Mixture	**0.779**	**0.602**
Agglomerative Clustering	0.698	0.501
Dataset 3		
DBSCAN	0.445	0.434
K-Means	**1.0**	**1.0**
Gaussian Mixture	0.75	0.845
Agglomerative Clustering	0.735	0.797
Dataset 4		
DBSCAN	**0.997**	**0.999**
K-Means	0.374	0.471
Gaussian Mixture	0.387	0.486
Agglomerative Clustering	0.623	0.646
Dataset 5		
DBSCAN	0.716	0.733
K-Means	0.197	0.259
Gaussian Mixture	**0.962**	**0.984**
Agglomerative Clustering	0.014	0.019

Table 2: Performance metrics for each data set and method. The best performing method was highlighted for each data set. It can be seen that each method performs best on different use cases.

Fig. 2 shows seven localized clusters. Judging from the plot as well as the NMI (0.97) and the Rand index (0.97), *DBSCAN* performed the best. However the other methods performed also reasonably well.

Fig. 3 shows six clusters which are partly very localized and partly spread out. Judging from the plot as well as the NMI (0.78) and the Rand index (0.6), *Gaussian Mixture* performed the best. However K-Means performed also reasonably well.

Fig. 4 shows two linearly separable clusters. Judging from the plot as well as the NMI (1.0) and the Rand index (1.0), *K-Means* separated the clusters perfectly. All other methods except DBSCAN performed also reasonably well.

Fig. 5 shows two half-circle shaped clusters. Judging from the plot as well as the NMI (0.99) and the Rand index (0.99), *DBSCAN* performed almost perfectly. However no other method could separate the clusters sufficiently.

Fig. 6 shows two almost linearly separable, irregularly shaped clusters. Judging from the plot as well as the NMI (0.96) and the Rand index (0.98), *Gaussian Mixture* performed the best. However no other method performed equally well.

It is clear, that each method has its advantages and disadvantages, as no single method could perform best across all data sets.

Task 3: Apriori Algorithm

The implemented algorithm can be found in the file *apriori.py*. The files *oneItems.txt* and *patterns.py* can also be found in the same location. It contains all frequent itemsets of the data set.

In order to summarize the result, the only frequent itemset that contains five books is the Harry Potter books from part 1-5. It has an absolute support of 50. The rest of the results can be found in the corresponding files.

Part c of the third task was not completed, thus it is neither included in this report nor in the Python-file.

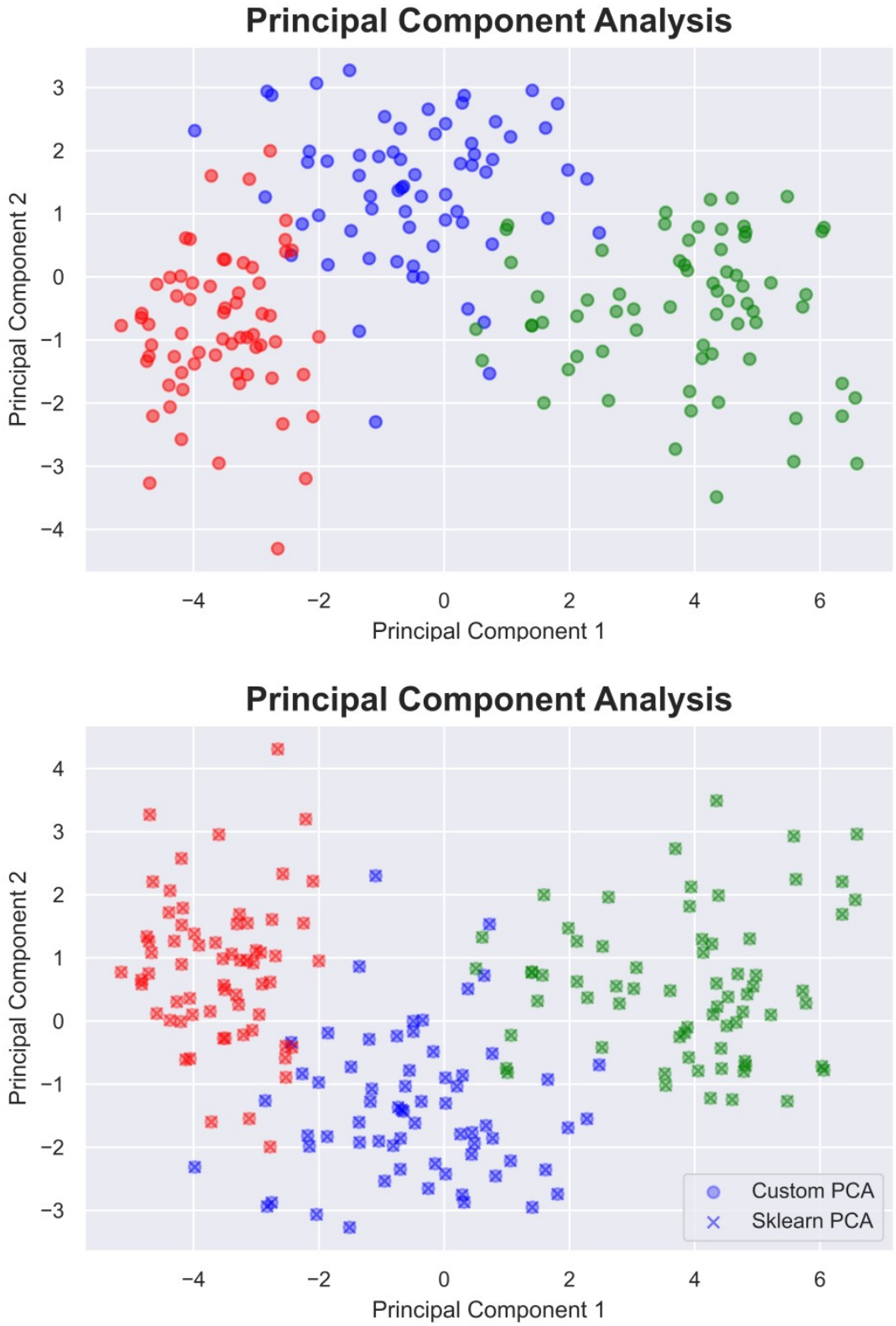

Figure 1: **Top**: PCA plot using the two most important PC's. Data is separated into three well-defined groups. The blue dots correspond to the label 1 of the original data, green to label 2 and red to label 3. PC1 and PC2 explain over 95% of the variance within the data. With PC1 (80.1 %) and PC2 (16.5 %). **Bottom**: Comparison of the results of the custom and the sklearn PCA. If the custom plot is mirrored along the x-axis the results of both methods are identical, which can be seen by the crosses (sklearn) and the dots (custom) overlapping while exhibiting the same color i.e. label.

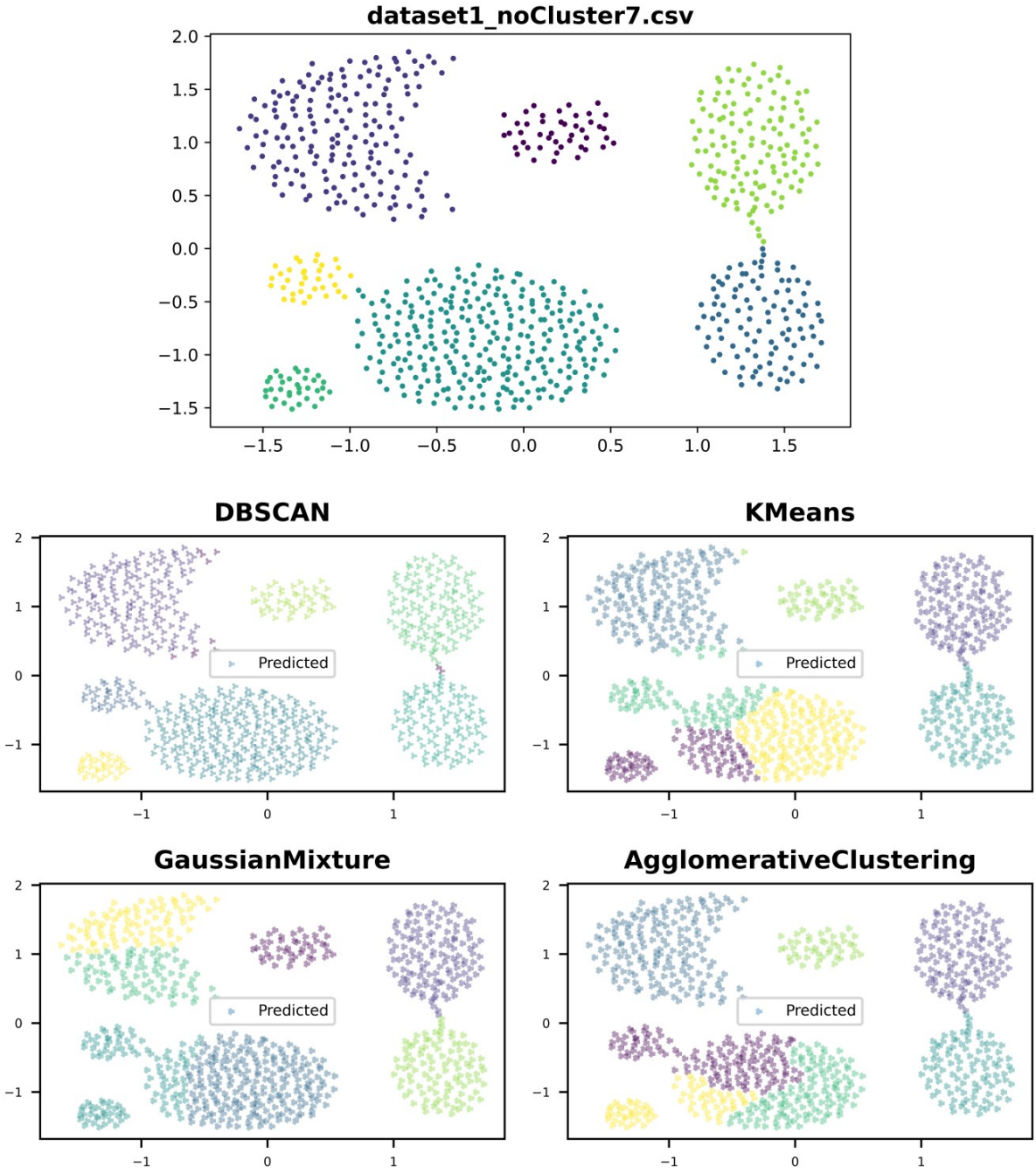

Figure 2: When comparing the plot and the results from table 2, DBSCAN performed the best.

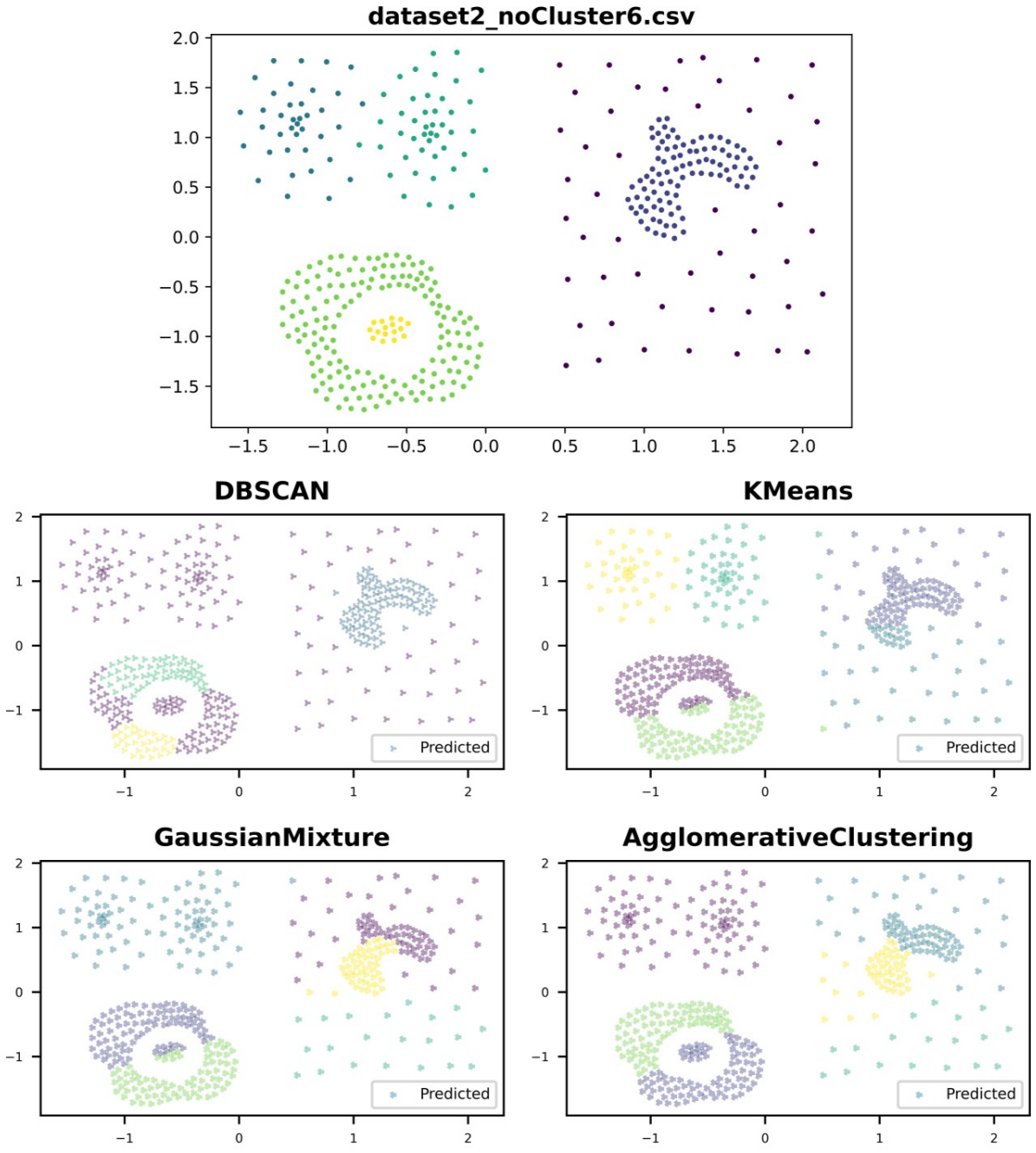

Figure 3: When comparing the plot and the results from table 2, Gaussian Mixture performed the best.

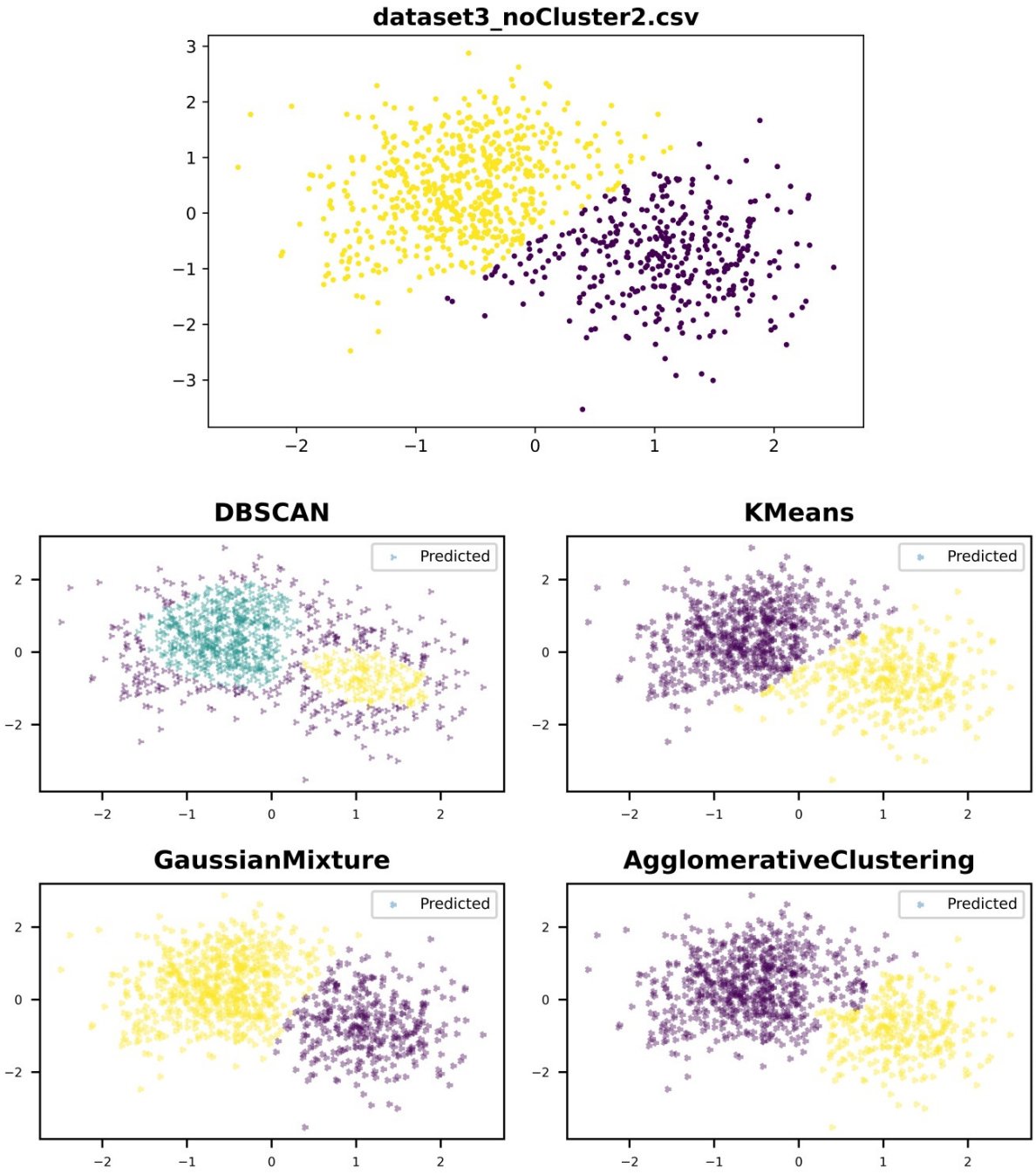

Figure 4: When comparing the plot and the results from table 2, K-Means performed the best.

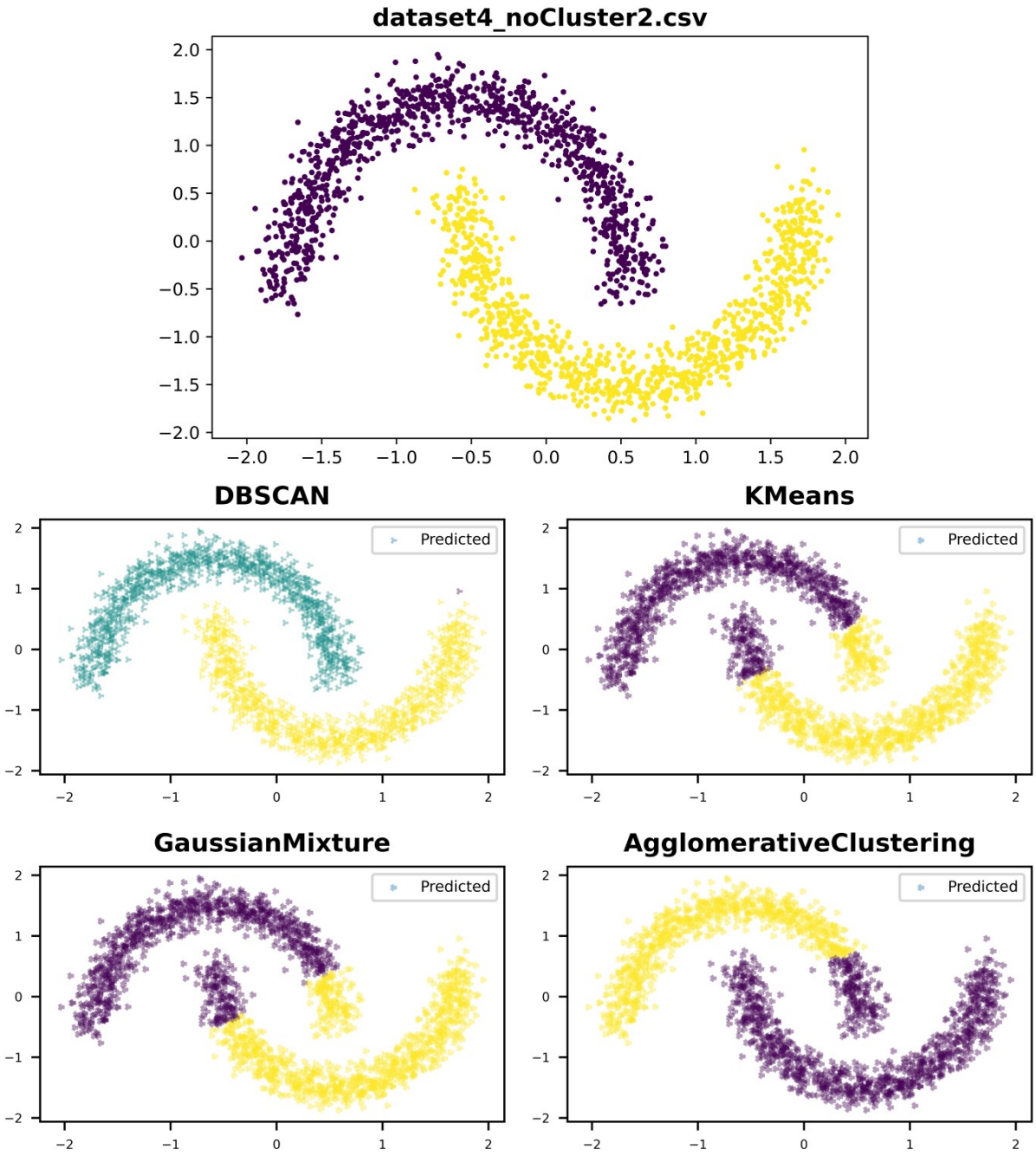

Figure 5: When comparing the plot and the results from table 2, DBSCAN performed the best.

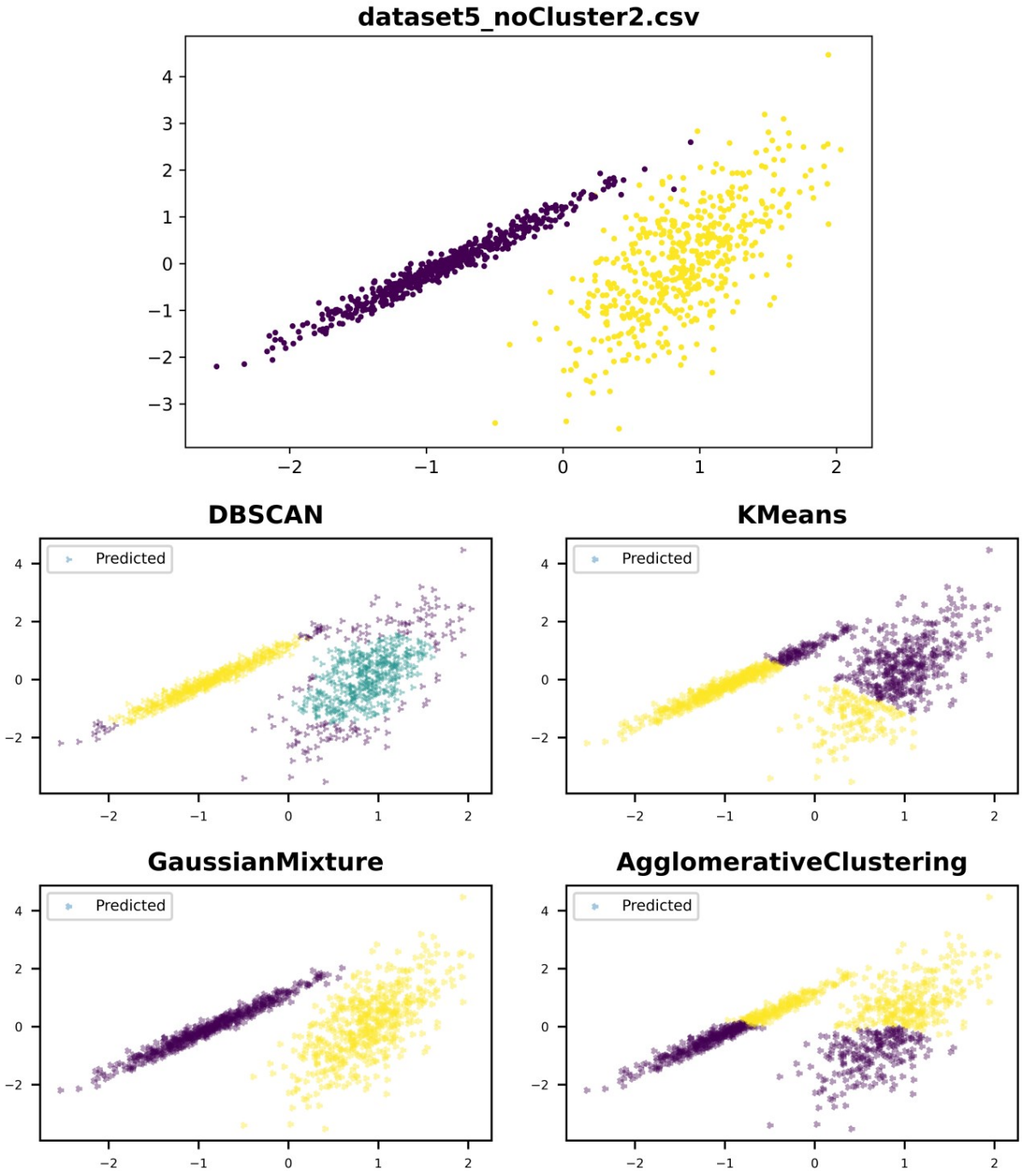

Figure 6: When comparing the plot and the results from table 2, Gaussian Mixture performed the best.