Ensemble Clustering

Project By: Tete Jordy Mensah

Initial Goal

Build an ensemble clustering algorithm

- Incorporate K-Means and DBSCAN
- Leverage their strengths and weaknesses
- Algorithm that would perform “better” than the individual algorithms

Ensemble Clustering Algorithm Using Similarity Matrix

- Model was created by João Pedro and was published in "Towards Data Science"
- How the model works

  - Step 1: Run the dataset through 128 (user specified) instances of K-Means

  - Step 2: Store each of those outputs and build a similarity matrix (option to convert to a distance matrix) based on cluster frequency of a point in a cluster.

  - Step 3: Use the matrix as an input for running your DBSCAN

- Main Benefit: 128 different K-Means runs minimized the sensitivity of initial centroids

What Happened?

Something was missing

- The model seemed to perform “better” visually than the independent clustering algorithms
- The Problem: The method still required the input of parameters by the user

  - K-Means and DBSCAN are both sensitive to their parameters

  - Wrong values impact accuracy

  - Knowing optimal values for parameters required domain knowledge and some experimenting

New Goal

- Build an ensemble clustering algorithm that incorporated both K-Means and DBSCAN
- Autonomously finds optimal parameter values
- Learns from individual clustering algorithm outputs
- The model has to be dynamic (has to be able to work on many different datasets)
- Relatively “good” performance
- Essentially became, can K-Means output be used to find most optimal parameters for DBSCAN

Hypothesis

- Model could perform well but not as well as manually finding optimal values

First Method (Model 1)

How Does It Work?

- First, run the dataset through K-Means (Starting with K=2 or user specified K)
- Output from K-Means determines DBSCAN parameters

  - EPS = average distance of each centroid from one another,

  - MinPts = total data points, divided by the amount of clusters(K) multiplied by 2 (ex: 300 data points, 2 clusters = 300/(2*2)

- Next, run the dataset through DBSCAN using the estimated parameter values
- Evaluate the results using silhouette score and store the results
- Loop back to the start of the function using 2 rules

  - If the current DBSCAN output has more clusters than the current K-Means iteration, the next loop begins with 1 more cluster(K) than the amount of clusters DBSCAN identified

  - If the DBSCAN output identifies the same or a lesser amount of clusters than the current iteration of K-Means, then the next loop begins with 1 more than the current amount of clusters the current K-Means run began with

- Model converges when the total number of iterations(user specified) are completed

Model 1 Evolutions

- Initial Centroids: At the start, the function would first perform a specified number (user specified) of K-Means runs and the one with the best inertia, would be chosen for the function. This was done to mitigate the sensitivity to initial centroids

- Convergence: Initially, the model was suppose to converge when there were 2 consecutive DBSCAN outputs where the silhouette score was lower than previous, and would return the best run before the consecutive low silhouette scores. This was changed as a result to fluctuating silhouette scores, as well as silhouette scores not being super reliable especially for DBSCAN

Model 1 Issues

- Visually wasn't performing to my expectations
- Low silhouette scores
- One model did not even return a DBSCAN result (because of very high eps)
- EPS estimates were very high
- Method for EPS and minPts estimation were not very good

Second Method: Model 2

How it works

- Works identically to model 1: K-Means output is used to estimate EPS and minPts

- Differences

  - K-Means is ran multiple times(user specified) until an optimal number of clusters is found based on the highest silhouette score

  - The optimal number of clusters (K) identified by the best K-Means output is used as k in knn to estimate EPS : calculates the distance to the k-th nearest neighbors for each point in the dataset

  - Elbow method: Using the plotted distances, the function determines the elbow point: where the rate of increase sharply changes

- DBSCAN is run with that estimated EPS and minPts is set to the optimal number of clusters (K)

Model 2.5

- Very similar to model 2
- Only difference is the minPts estimation, it is set to a percentage of the average number of points in each cluster found in the K-Means runs (15%)

Datasets Used

Dataset 1

- Created using make blobs package
- Suppose to be 4 clusters but because of high standard deviation, it visually looks more like 3 or 2
- High Standard deviation causes for less distinct clusters

Dataset 2

- Created using make blobs package
- 3 clusters
- Low standard deviation so more distinct and easily identifiable clusters

Dataset 3 (created by João Pedro)

- Created using make blobs and make moons package
- Less spherical clusters, more noise
- Should be more difficult for the models

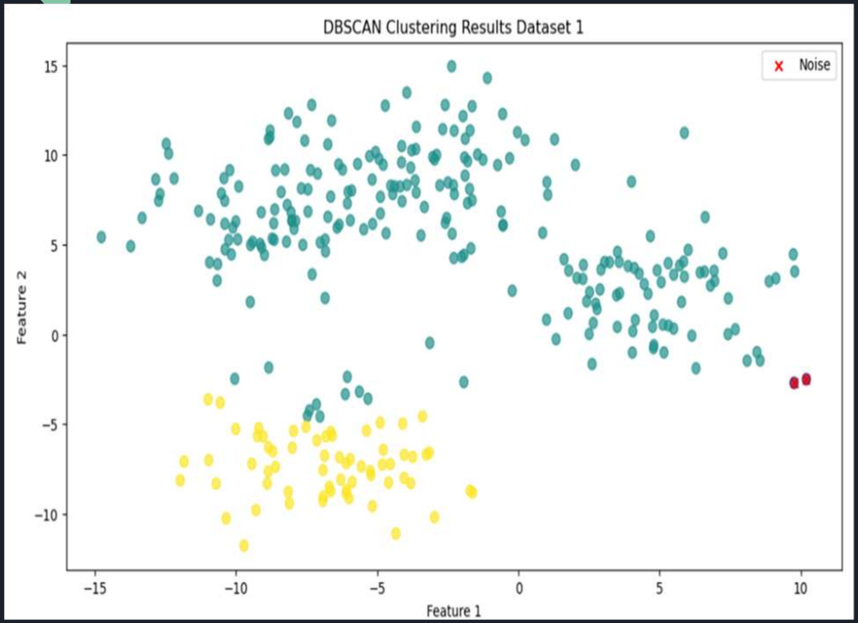
Dataset 4 (Kaggle Dataset)

- Real world wine dataset
- Has many (13) features
- Because of the many dimensions, harder to visualize so PCA is used

Results
(Visual Evaluation)

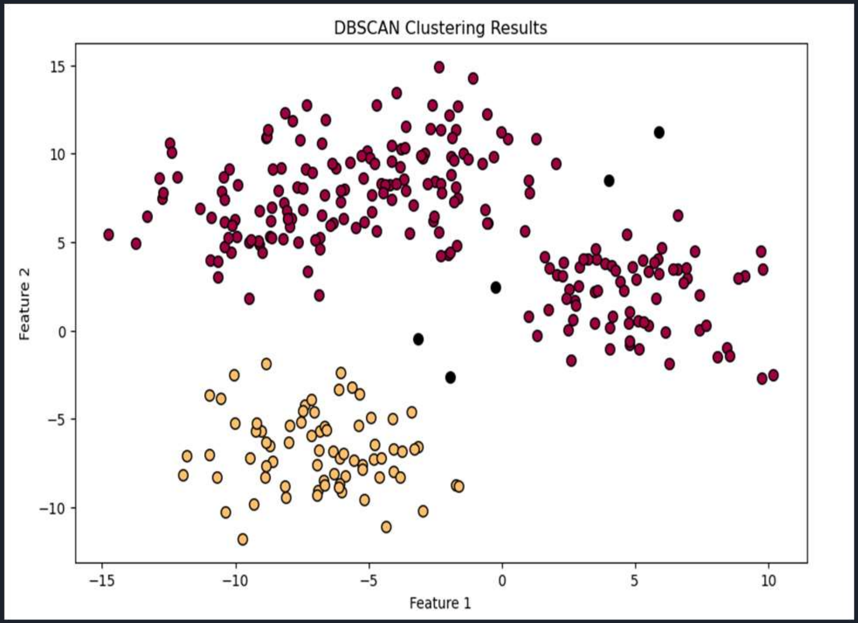
Dataset 1

Model 1



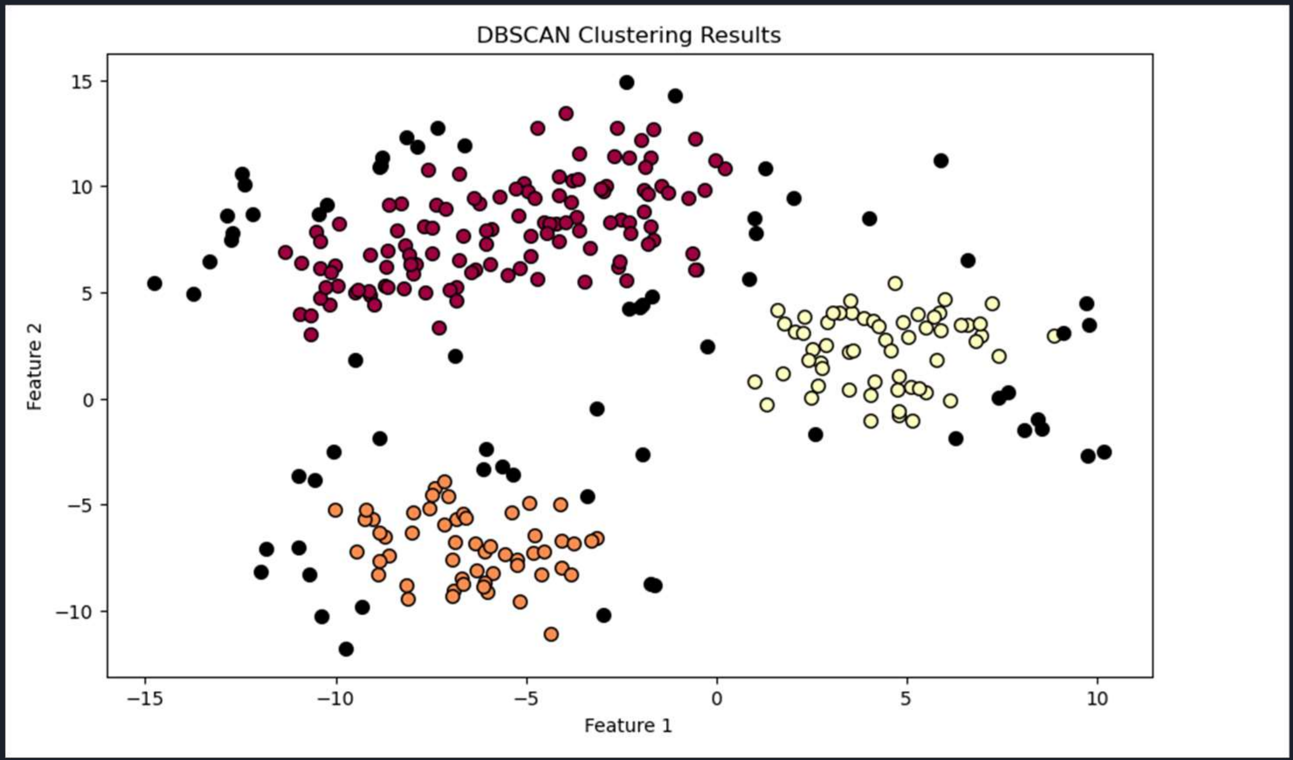
Did a decent job, the clusters look visually ok, however could have benefited from another cluster

Model 2



Seems like it did a better job when considering silhouette score, however take the score with a grain of salt, visually looks very similar to how model 1 clustered dataset. Overall a tough dataset because of how closely packed some of the clusters are

Model 2.5



Impressive results, suppose to be 4 clusters but when I look at it, it makes sense why there are 3 clusters. Visually looks more like 3 clusters. Could have handled noise points more accurately

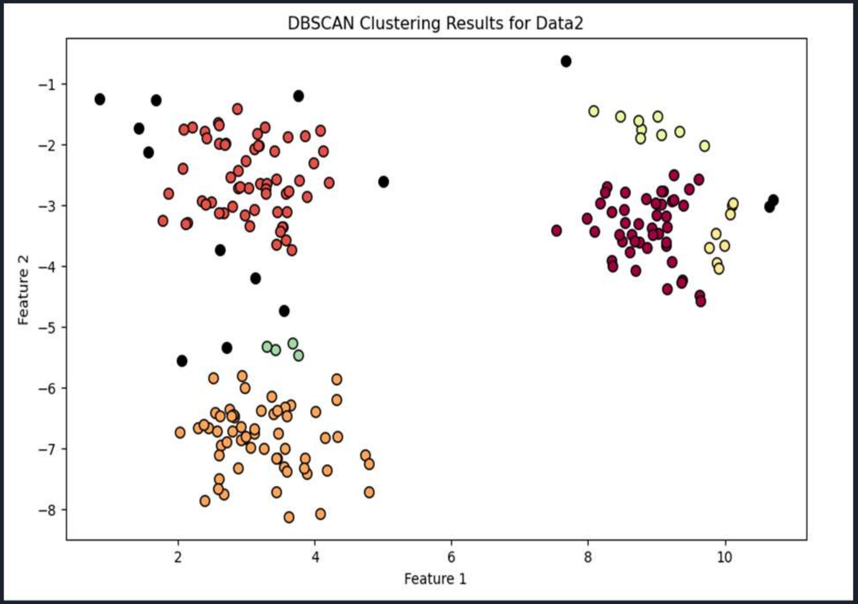
Dataset 2

Model 1

N/A

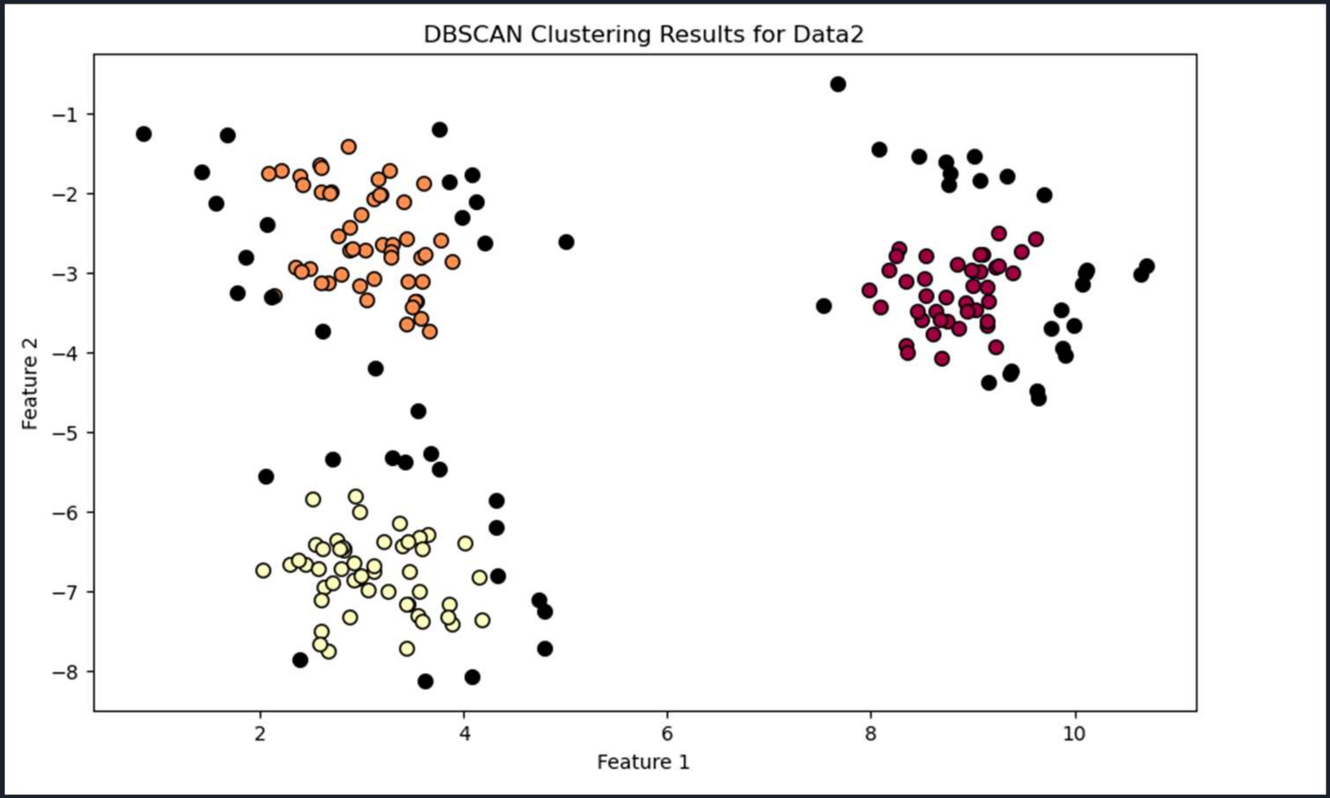
Had a very difficult time getting a DBSCAN output. I believe the problem might have come from EPS and minPts estimation. The problem comes from extremely high eps and minPts estimations by model 1.

Model 2



Visually, the clustering looks very good, it should have only been 3 clusters realistically, but by looking at the clusters created the 3 biggest clusters are what we expected. Silhouette score not great but decent.

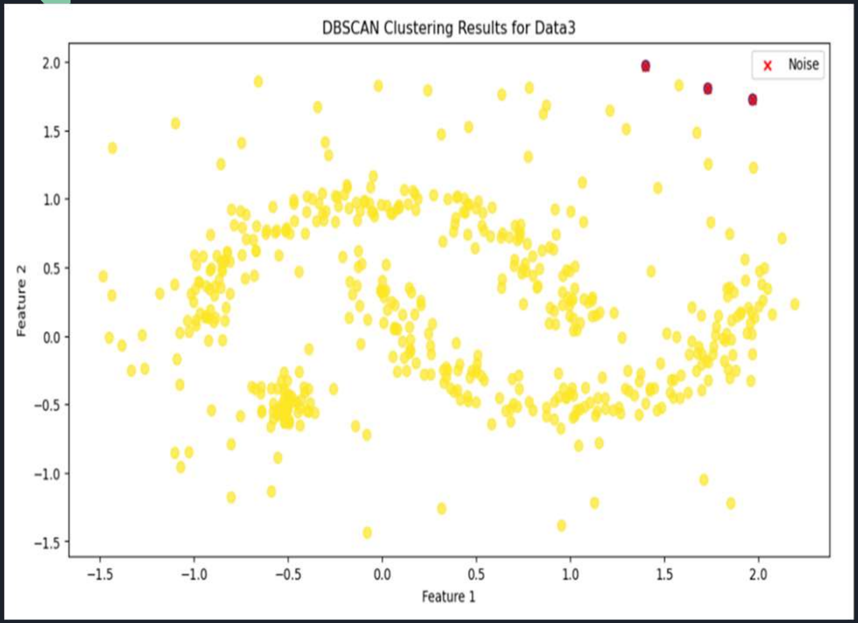
Model 2.5



Very impressive performance by model 2.5 on this dataset, noise points could be a little better but it definitely found the correct amount of clusters

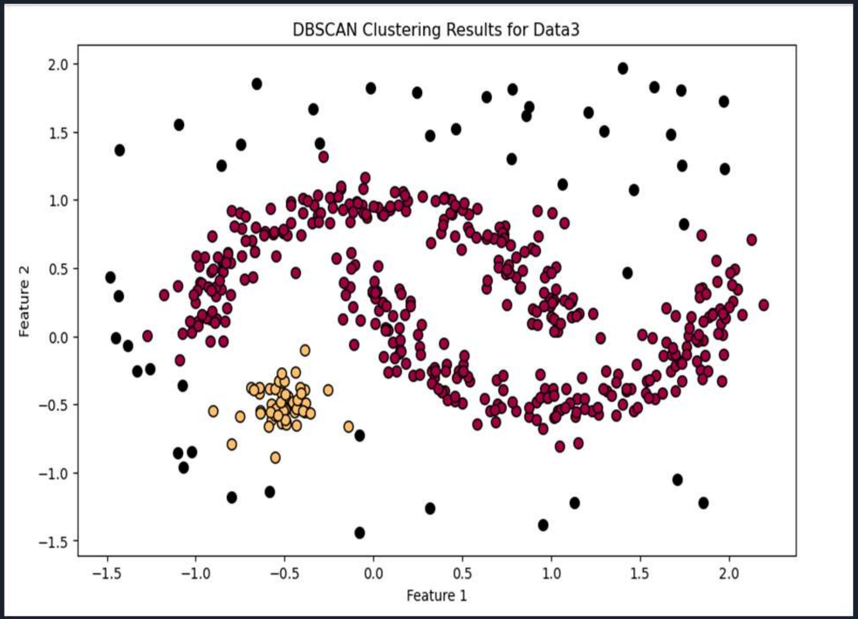
Dataset 3

Model 1



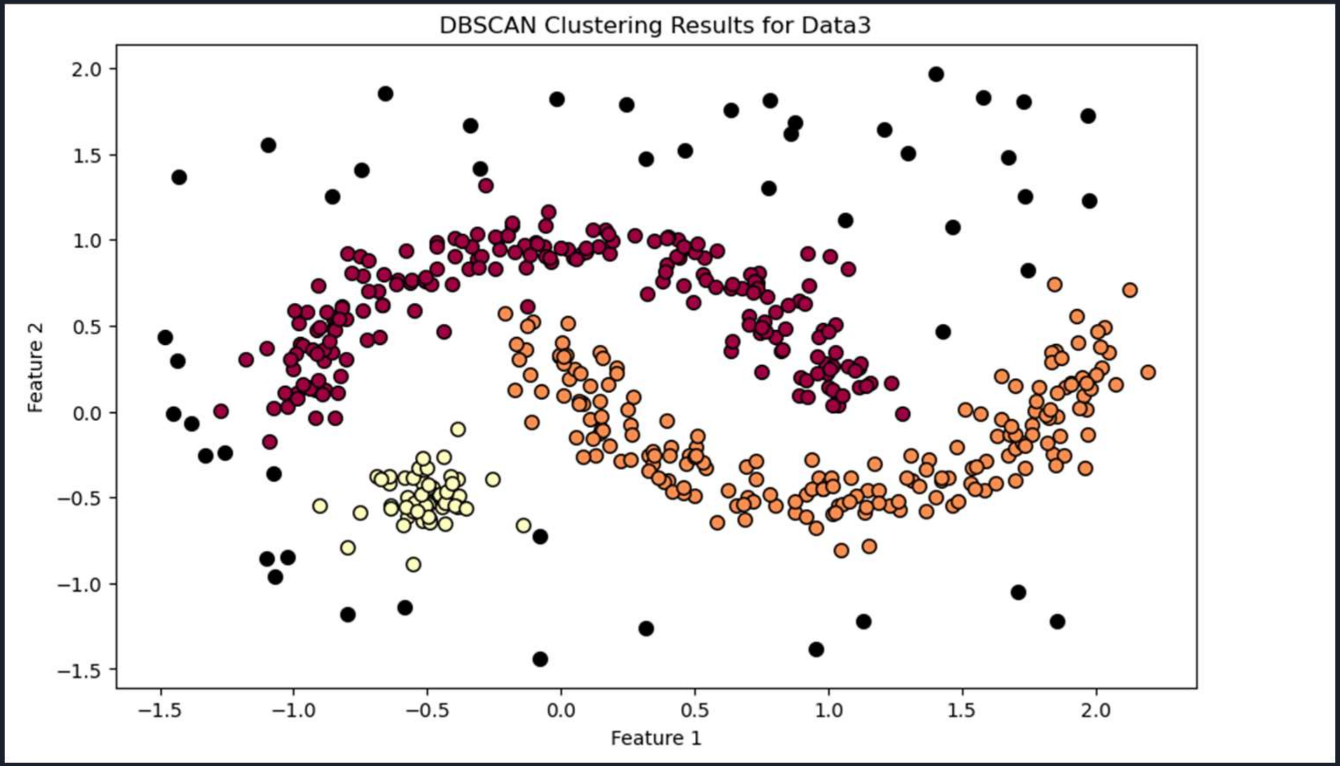
Horrible performance by this model on this dataset. Clustered almost every point into the same cluster except for outliers. Don't know exactly what I expected but it was a bad performance.

Model 2



Much better performance than model 1. Was able to find multiple clusters but visually, it handled the noisy points much much better

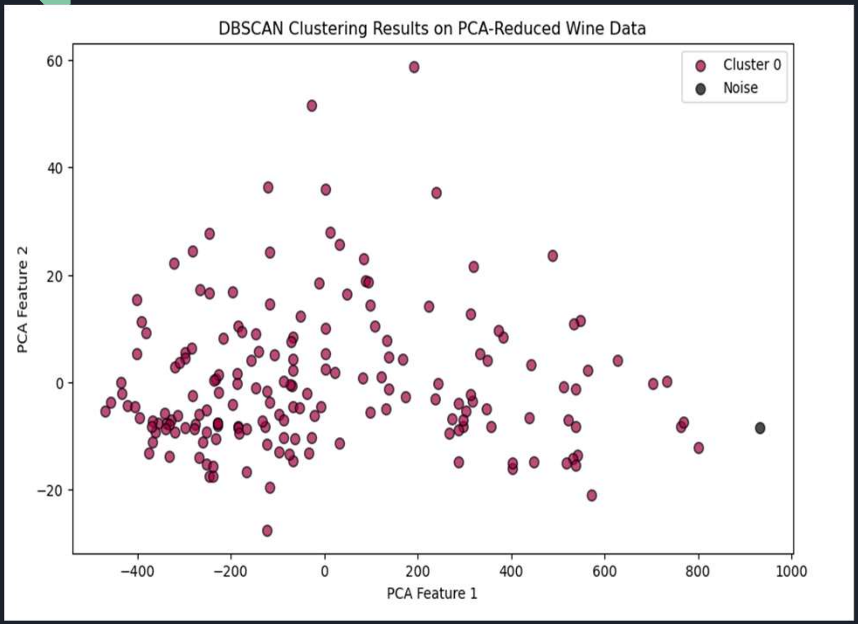
Model 2.5



I believe of all 3 models, on this dataset, this might have been the best performance because the amount of clusters seem to be correct. The noise points are somewhat correct as well.

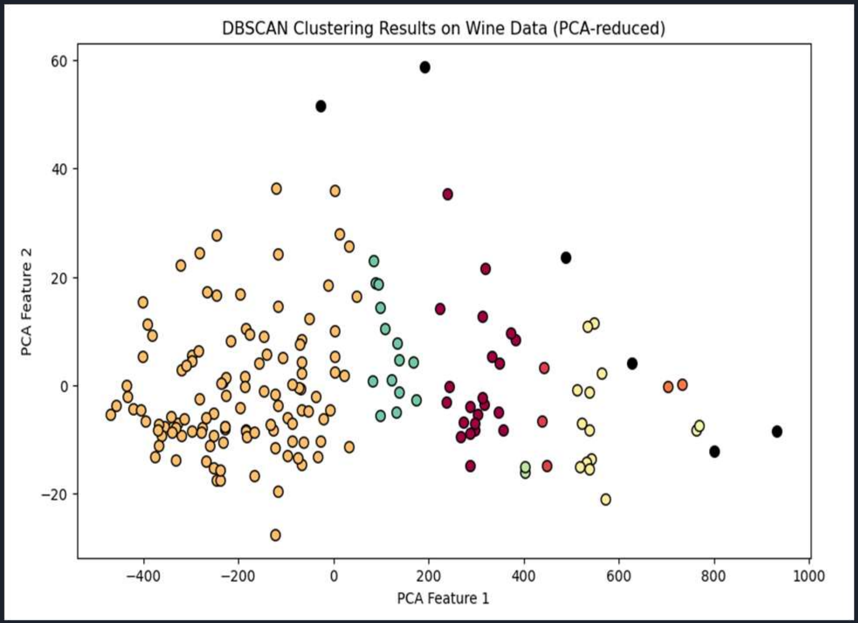
Dataset 4

Model 1



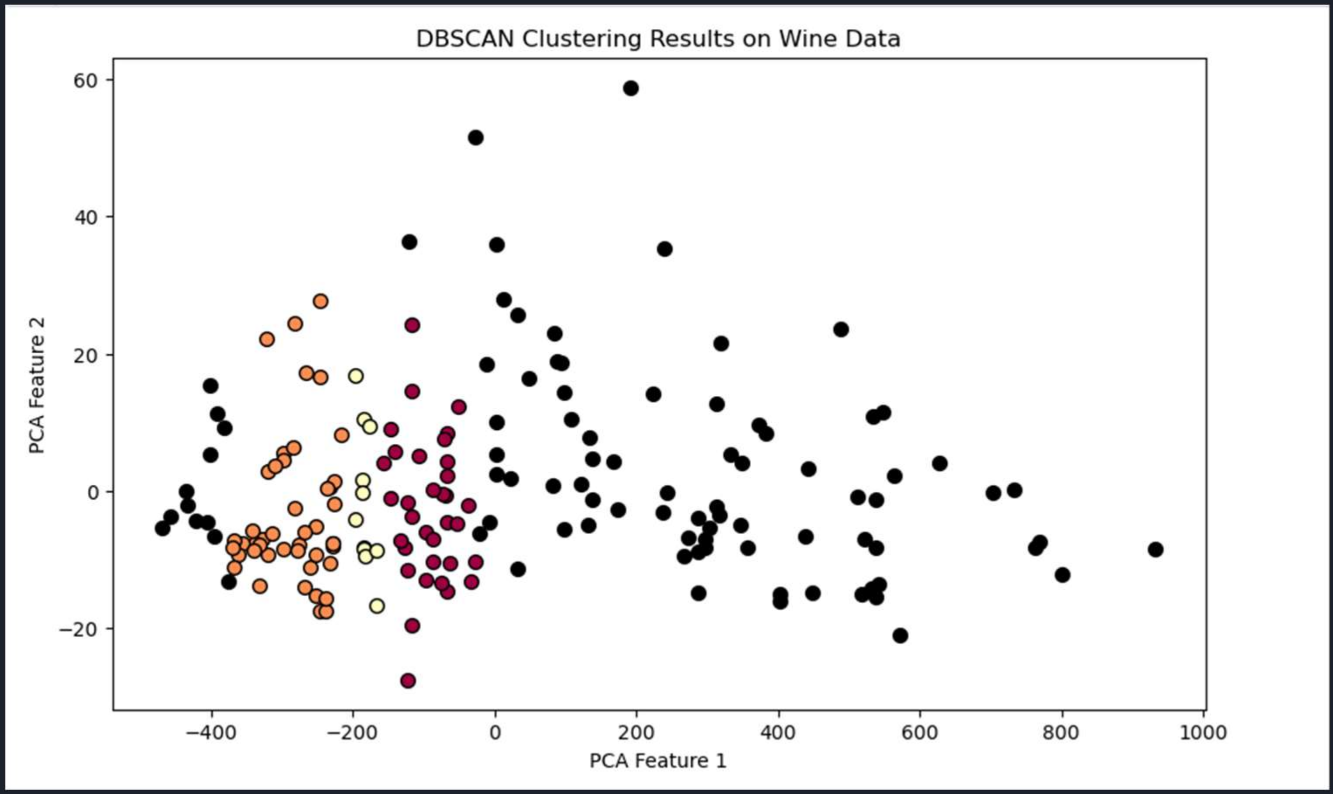
Much like the performance in dataset 3, it performed horribly. It also clustered most points into one single cluster. Again not sure what I was exactly expecting but not much clustering was done here.

Model 2



I was actually very pleasantly surprised by this models performance on this dataset. 8 Clusters were found, some bigger than others. From the main clusters just taking a look at them visually, they seem to make sense.

Model 2.5



Feels like a worse performance by model 2.5, most points labeled as noise, however some clusters found. It could be because of high dimensionality.

Conclusion

- **Performance:** Models 2 and 2.5 were the better performing models by far. Better performance by 2.5 could be from a bias in datasets 1-3 (scale and cluster shapes)
- **EPS estimation:** Not completely satisfied with the method used to determine K in knn but seems to work decently on the datasets in this project
- **MinPts Estimation:** Model 2 minPts estimation could use some tuning, shows an improvement when it was better tuned in model 2.5, but would need better estimation methods
- **Poor Efficiency:** Overall all models seem computationally inefficient, however model 2 and 2.5 seem to be more efficient
- **Outcome:** Don't think I necessarily failed but did not achieve a model that performed how I would like it to. I plan on continuing to work on this project (had to find a good place to pause)

Works Cited

- Pedro, J. (2022, May 19). How to ensemble clustering algorithms. Medium. https://towardsdatascience.com/how-to-ensemble-clustering-algorithms-bf78d7602265
- Saji, B. (2023, September 20). Elbow method for finding the optimal number of clusters in K-means. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2021/01/in-depth-intuition-of-k-means-clustering-algorithm-in-machine-learning/
- Selecting the number of clusters with silhouette analysis on kmeans clustering. scikit. (n.d.). https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_silhouette_analysis.html
- Bhardwaj, A. (2020, May 27). Silhouette coefficient : Validating clustering techniques. Medium. https://towardsdatascience.com/silhouette-coefficient-validating-clustering-techniques-e976bb81d10c