# Assignment 3: Clustering Algorithm Self-Study

Deadline: Friday, February 28th 2025

## 1. Algorithm Overview (20%)

The DBSCAN algorithm assigns points to clusters based on the density of points in a region. High density regions are identified using the epsilon parameter, which defines how close points must be to be considered neighbors. A point becomes a core point if it has at least min_samples neighbors. Clusters are formed by linking core points that are neighbors, while non-core points must be neighbors with a core point to be included in the cluster. Points with too few neighbors remain unclustered and are marked as outliers. Due to the iterative addition of core points to a cluster, DBSCAN works well at identifying irregularly shaped clusters and nested clusters but is computationally expensive for large datasets. DBSCAN struggles with varying density in a dataset and its results are highly sensitive to epsilon and min_samples.

## 2. Algorithm Comparison (40%)

DBSCAN outperforms k-means clustering and hierarchical clustering when clusters are nested and irregularly shaped (Fig. 1). Using the make_moons dataset, it was shown that DBSCAN was able to identify clusters that overlap in the x and y axis (Fig. 1). DBSCAN is also useful when the number of clusters is unknown, as both k-means and agglomerative clustering require that the number of clusters is defined prior to executing the algorithm.

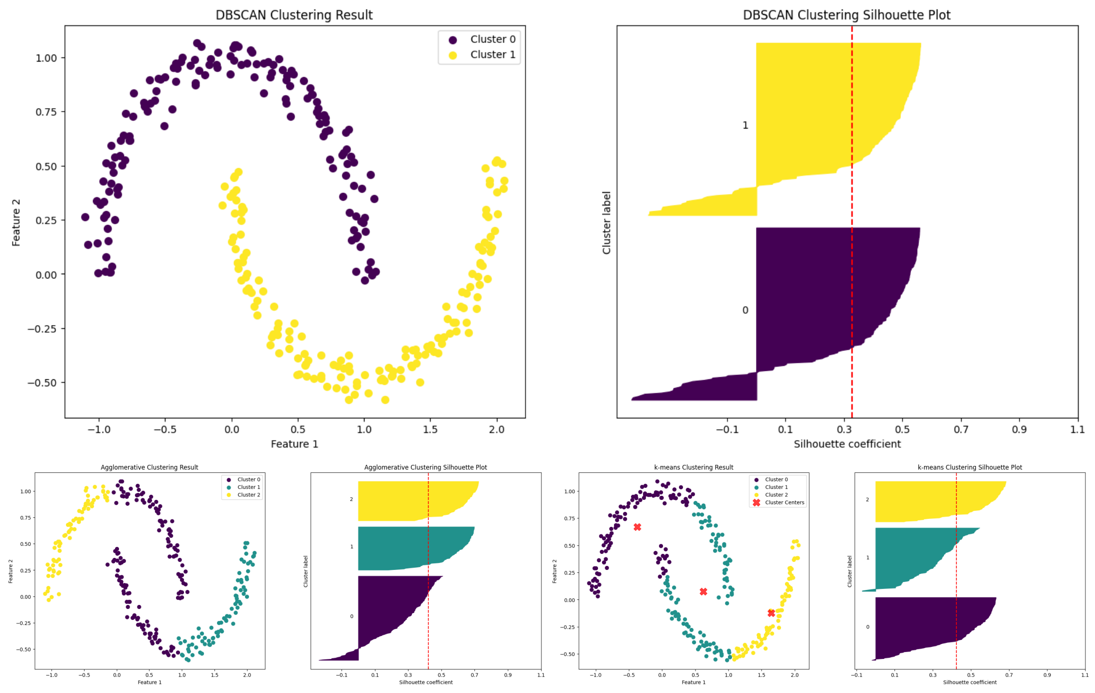

**Figure 1. Performance of clustering algorithms on the make_moons dataset.** Top: DBSCAN clustering results. The silhouette score plot indicates that the clusters overlap, but the scatterplot shows DBSCAN successfully attributed points to one of two distinct clusters despite the irregular shape of the data (score = 0.33). Left: Agglomerative clustering results. Points have been assigned to clusters, but the purple cluster being spread out reduces its silhouette score. Right: k-means clustering result. Similarly to agglomerative clustering, points

are clustered but do not capture the moon shape. The irregular shape of the blue cluster results in a lower silhouette score.

DBSCAN is outperformed by agglomerative clustering and k-means clustering when the data has variable density. The DBSCAN scatterplot (Fig. 2.) shows that clusters were poorly formed, as the data is not well separated by either feature.

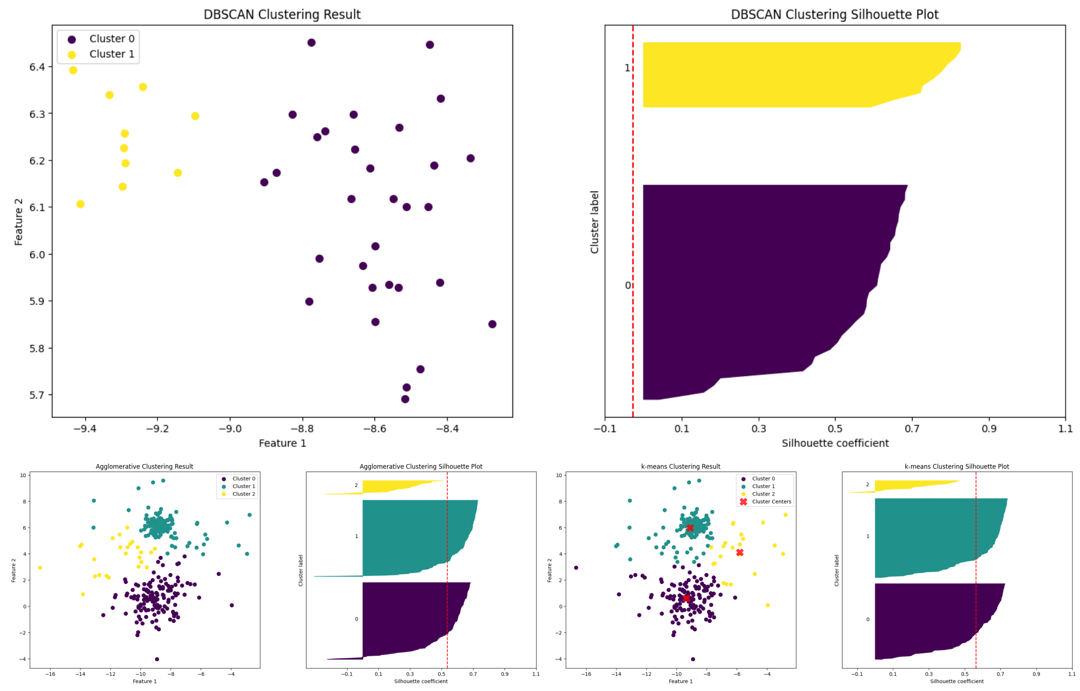

**Figure 2. Performance of clustering algorithms on the make_blobs dataset.**
Top: DBSCAN clustering results. The average silhouette score being close to 0 (red line, score = -0.03) indicates that the algorithm struggles to place data in one cluster versus the other. The scatterplot shows that the clusters are sparse and poorly defined. Left: Agglomerative clustering results. Points have been assigned to clusters with moderate success (score = 0.54), despite some overlap of clusters. Right: k-means clustering result. Points are assigned to clusters with moderate success and slightly more effectively (score = 0.56) than agglomerative clustering.

Overall, this analysis shows that DBSCAN, k-means clustering, and agglomerative clustering are all viable clustering algorithms, and the choice of algorithm should be guided by the nature of the dataset. When the data has consistent density, the identification of outliers is important, clusters may be irregularly shaped, and the number of clusters is not known prior to clustering, DBSCAN is ideal. Real world applications of clustering that would work well with DBSCAN include image recognition, where irregularly shaped clusters of pixels may be used to identify objects, or the large-scale analysis of sensor readings, where the identification outliers can identify faulty equipment.

## 3. Table Update (20%)

| **Feature** | **k-Means** | **Hierarchical Clustering** | **DBSCAN** |
|---|---|---|---|
| **Definition** | Partitioning algorithm that assigns points to k clusters based on centroids | Builds a hierarchy of clusters using distance metrics | Density-based clustering of points |
| **Approach** | Iteratively minimizes variance within k clusters | Agglomerative (bottom-up) or divisive (top-down) | Assignment of core and non-core points, followed by iterative expansion of clusters |
| **Number of Clusters** | Requires predefined k | Can be determined from dendrogram but subjective | Can be determined from the resultant plot |
| **Cluster Shape** | Prefers spherical clusters | Works well with various shapes but can be unstable | Works well with irregular shapes and nested clusters |
| **Initialization** | Randomly selects k initial centroids | No initialization needed | Assigns core points and non-core points based on density |
| **Result** | Hard assignments—each point belongs to a single cluster | Hierarchical structure (tree/dendrogram) | Points may be assigned to a cluster, or they may be left out of all clusters if they are outliers |
| **Interpretability** | Moderate—cluster assignments but no hierarchy | High—dendrogram can be analyzed | Moderate – data is clearly assigned to clusters |
| **Strengths** | Simple, fast and efficient on large datasets | Can capture hierarchical relationships | Can identify nested and irregularly shaped clusters that may be subdivided by other methods (e.g., k-means) |
| **Limitations** | Sensitive to initial centroids and k choice | Computationally expensive for large datasets | Computationally expensive, sensitive to epsilon and min_points |

## 4. Code Documentation & Submission Quality (20%)

https://github.com/r-shadoff/BINF5507/tree/55772ac495f2882a0083fa18f4cf8f706c55d0ac/Assignment3