**Assignment 3: Clustering Algorithm Self-Study**

**Student:** Aryan Rathod

[aryanrathod23/dbscan\_clustering\_assignment](https://github.com/aryanrathod23/dbscan_clustering_assignment)

### 1. Algorithm Overview (20%)

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is a density-based clustering algorithm commonly used when data contains clusters of varying shapes and noise. Unlike partitioning methods such as k-Means that assume spherical cluster shapes or hierarchical methods that require linkage decisions, DBSCAN forms clusters based on the density of points in a region.

#### Cluster Identification

DBSCAN defines clusters as areas of high point density, distinguishing them from areas of low point density (noise). It uses two main concepts:

* Core Points: A point is a core point if there are at least min\_samples points (including itself) within a radius of eps.
* Border Points: A point that is within eps of a core point but does not have enough neighbors to be a core point itself.
* Noise Points: Points that are not within eps of any core point.

DBSCAN starts from an arbitrary point and expands clusters by recursively visiting all density-reachable points (i.e., points that can be reached from a core point within the given eps).

#### Key Parameters

* eps (epsilon): The maximum distance between two points for them to be considered as in the same neighborhood. A smaller value results in smaller, tighter clusters, while a larger eps may merge distinct clusters.
* min\_samples: The minimum number of points required to form a dense region. Commonly set to values between 3 and 10, depending on dataset size and dimensionality.

#### Strengths

* Can find clusters of arbitrary shape.
* Effectively handles noise and outliers.
* No need to specify the number of clusters in advance.
* Works well for spatial datasets and with non-spherical structures.

#### Limitations

* Sensitive to the choice of eps and min\_samples.
* Struggles with datasets having clusters of varying densities.
* Less efficient on very large or high-dimensional datasets.

### 2. Algorithm Comparison (40%)

To visually assess the performance of DBSCAN against k-Means and Hierarchical Clustering, we applied all three algorithms on two datasets generated from sklearn.datasets:

* Dataset 1: make\_moons(n\_samples=300, noise=0.05)
  + Characteristics: Two interlocking half-moon shapes. Ideal for evaluating non-spherical cluster detection.
  + DBSCAN successfully identified the two curved clusters and handled noise points accurately. This dataset highlights DBSCAN’s ability to work with arbitrarily shaped clusters.
  + k-Means failed to capture the non-linear shape due to its assumption of spherical clusters.
  + Hierarchical Clustering performed moderately better than k-Means but still struggled to separate the curved shapes cleanly.
* Dataset 2: make\_blobs(n\_samples=300, centers=3, cluster\_std=[1.0, 2.5, 0.5])
  + Characteristics: Three Gaussian blobs with different densities. Useful for testing algorithm sensitivity to density variations.
  + DBSCAN struggled to detect all three clusters correctly due to the varying densities. Some clusters were merged, and several points were marked as noise incorrectly.
  + k-Means performed well, correctly identifying the three clusters despite their differences in density spread.
  + Hierarchical Clustering also produced good results, though slightly more sensitive to linkage method used.

#### Analysis (20%)

Performance Comparison: - DBSCAN outperforms both k-Means and Hierarchical Clustering on datasets with non-spherical shapes (e.g., make\_moons). - k-Means performs well when clusters are spherical and similarly sized. - Hierarchical Clustering provides a good compromise and is especially insightful when a dendrogram is needed.

Failure Cases: - DBSCAN underperforms when clusters have significantly different densities, as it uses global eps and min\_samples values. - k-Means struggles with non-spherical and overlapping clusters. - Hierarchical Clustering is computationally expensive and may require trial-and-error with linkage methods.

Trade-offs: - DBSCAN eliminates the need to pre-specify the number of clusters but is sensitive to its parameters. - k-Means is scalable and efficient but rigid in assumptions. - Hierarchical Clustering is interpretable but slower and prone to instability depending on the linkage strategy.

### 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 algorithm that groups points based on local density and marks noise |
| Approach | Iteratively minimizes variance within k clusters | Agglomerative (bottom-up) or divisive (top-down) | Expands clusters from core points based on density threshold (eps) |
| Number of Clusters | Requires predefined k | Can be determined from dendrogram but subjective | Automatically determined by density and connectivity |
| Cluster Shape | Prefers spherical clusters | Flexible, can detect various shapes | Arbitrary shapes (non-linear, complex) |
| Initialization | Requires k random centroids | No initialization required | No initialization required |
| Result | Hard assignments (one cluster per point) | Hierarchical structure (tree/dendrogram) | Hard assignments + identifies noise (label -1) |
| Interpretability | Moderate | High (dendrogram analysis possible) | Moderate |
| Strengths | Simple, fast, efficient on large datasets | Reveals hierarchy, no need to define k initially | Handles noise, no need to set k, good for spatial data |
| Limitations | Sensitive to k and centroid initialization | Computationally expensive for large datasets | Sensitive to eps/min\_samples, poor with varying densities |

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

The clustering code is written in Python using libraries such as scikit-learn, matplotlib, seaborn, and pandas. The code includes inline comments explaining each step and uses functions for plotting and dataset generation to ensure modularity. Plots are generated with appropriate labels and saved in a plots/ directory.

All required clustering plots (6 in total) were produced: - DBSCAN, k-Means, and Hierarchical Clustering on make\_moons - DBSCAN, k-Means, and Hierarchical Clustering on make\_blobs

The script successfully compares all three algorithms across datasets and outputs visual insights along with a comprehensive comparison table.

A graph with yellow and purple dots

AI-generated content may be incorrect.A graph with different colored dots

AI-generated content may be incorrect.A graph of a clustering diagram

AI-generated content may be incorrect.A graph with different colored dots

AI-generated content may be incorrect.A graph with yellow and purple dots

AI-generated content may be incorrect.A graph with different colored dots

AI-generated content may be incorrect.

#### GitHub Repository

[aryanrathod23/dbscan\_clustering\_assignment](https://github.com/aryanrathod23/dbscan_clustering_assignment)