3 - Unsupervised Models - DBSCAN

By: eng. Esraa Madhi

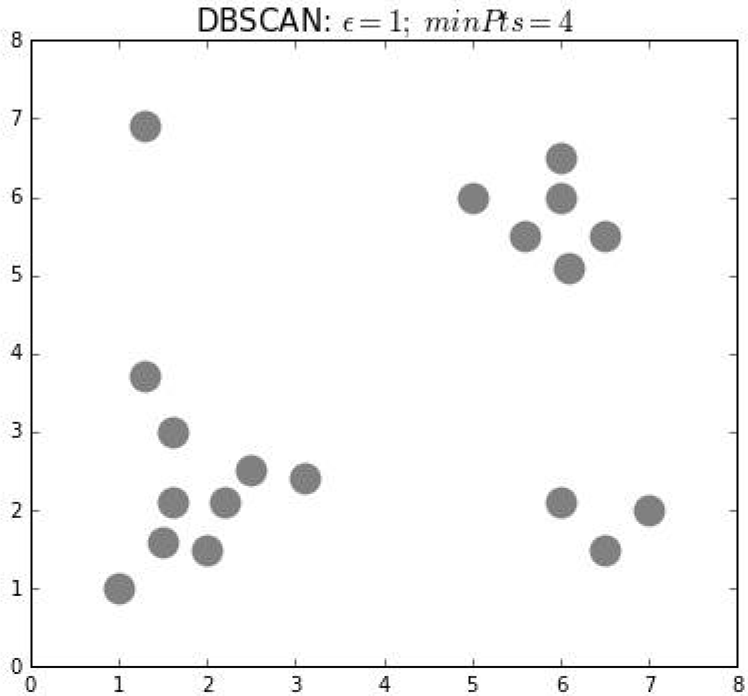
What is DBSCAN?

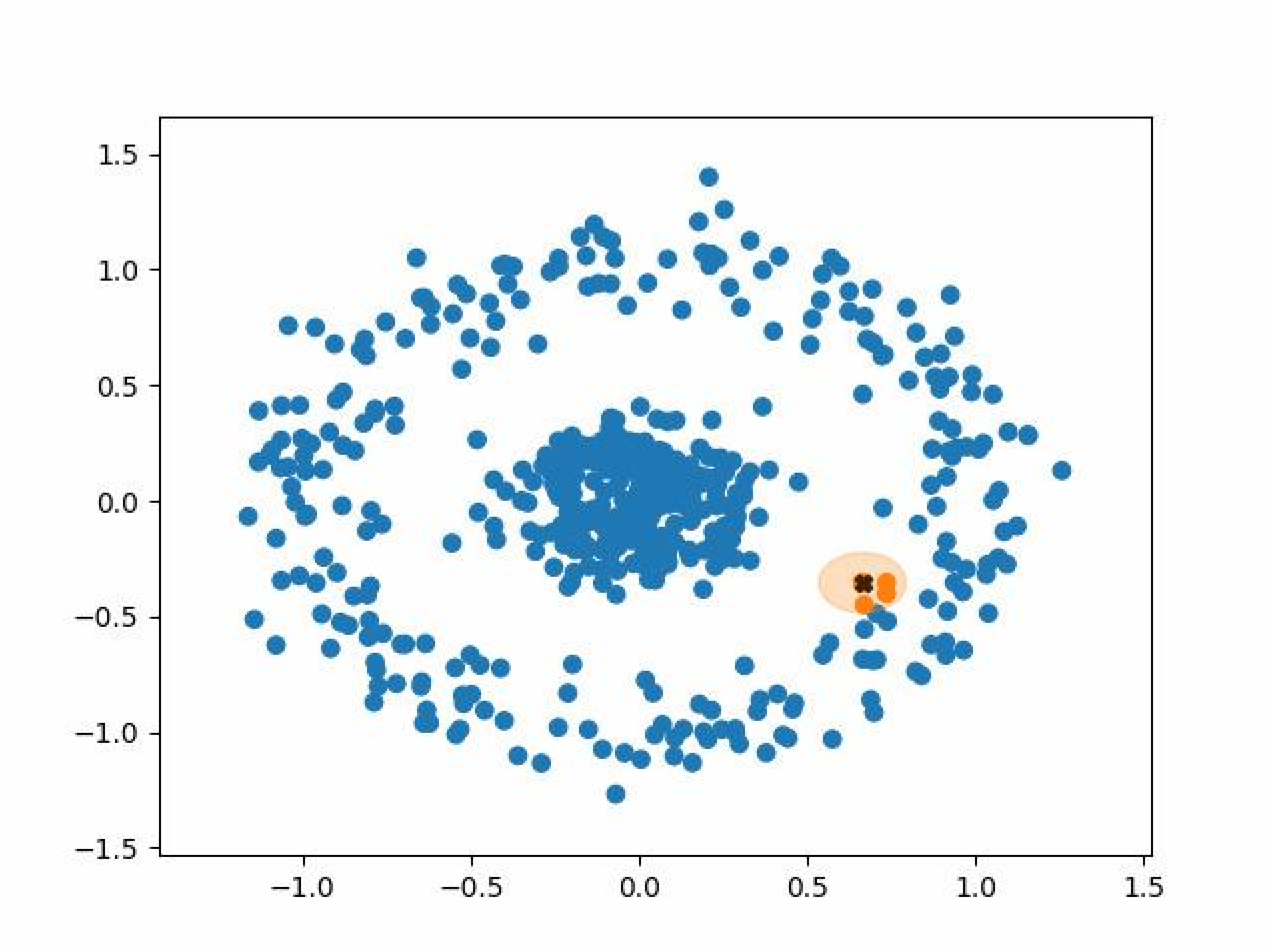
https://youtu.be/sJQHz97sCZ0

https://youtu.be/sJQHz97sCZ0

DBSCAN is a density-based clustering algorithm that **finds clusters by grouping data points that are closely packed together, based on a specified distance and density threshold**. DBSCAN is particularly useful for identifying clusters of defined shapes and for detecting noise in the data.

How DBSCAN works?





In Steps:

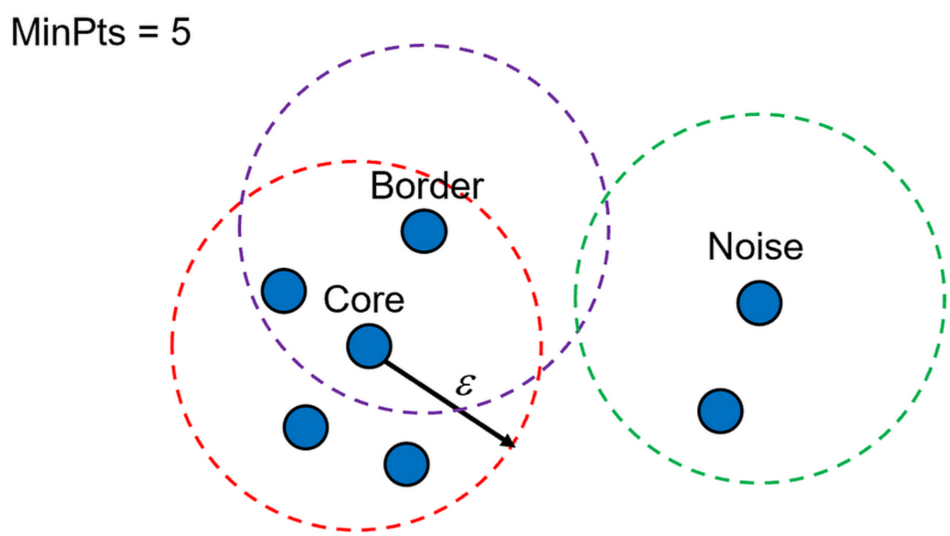
1. For each unvisited data point, determine its neighborhood based on a specified distance threshold (eps).
2. If the number of neighbors is greater than or equal to a specified density threshold (min_samples), create a new cluster and add the data point and its neighbors to the cluster.
3. Expand the cluster by recursively checking the neighbors' neighbors, following the same density criteria.
4. Repeat steps 1 to 3 for all unvisited data points.

The algorithm works by computing the distance between every point and all other points. We then place the points into one of three categories:

- **Core point:** A point with at least ***min_samples*** points whose distance with respect to the point is below the threshold defined by epsilon.

- **Border point:** A point that isn't in close proximity to at least ***min_samples*** points but is close enough to one or more core point. Border points are included in the cluster of the closest core point.

- **Noise point:** Points that aren't close enough to core points to be considered border points. Noise points are ignored. That is to say, they aren't part of any cluster.



Example:

****** Minpts = 3, Eps = 1.5, Objective function = Euclidean distance ******

A
dist(A,B) = 0.71
dist(A,C) = 5.66
dist(A,D) = 3.61
dist(A,E) = 4.24
dist(A,F) = 3.20
A is a noise point

B
dist(B,A) = 0.71
dist(B,C) = 4.95
dist(B,D) = 2.92
dist(B,E) = 3.54
dist(B,F) = 2.50
B is a noise point

C
dist(C,A) = 5.66
dist(C,B) = 4.95
dist(C,D) = 2.24
dist(C,E) = 1.41
dist(C,F) = 2.50
C is a border point

D
dist(D,A) = 3.61
dist(D,B) = 2.92
dist(D,C) = 2.24
dist(D,E) = 1.00
dist(D,F) = 0.50
D is a core point

E
dist(E,A) = 4.24
dist(E,B) = 3.54
dist(E,C) = 1.41
dist(E,D) = 1.00
dist(E,F) = 1.12
E is a core point

F
dist(F,A) = 3.20
dist(F,B) = 2.50
dist(F,C) = 2.50
dist(F,D) = 0.50
dist(F,E) = 1.12
F is a core point

Demo: https://www.naftaliharris.com/blog/visualizing-dbscan-clustering/

The key parameters for DBSCAN are:

- **eps:** The maximum distance between two data points to be considered neighbors.
- **min_samples:** The minimum number of data points required to form a dense region (cluster).

How to choose Eps, min_samples parameters?

There is no automatic way to determine the `eps` & `min_samples` value for DBSCAN. They are crucial parameters and **their values depend heavily on the dataset.**

Minimum Samples ("MinPts"):

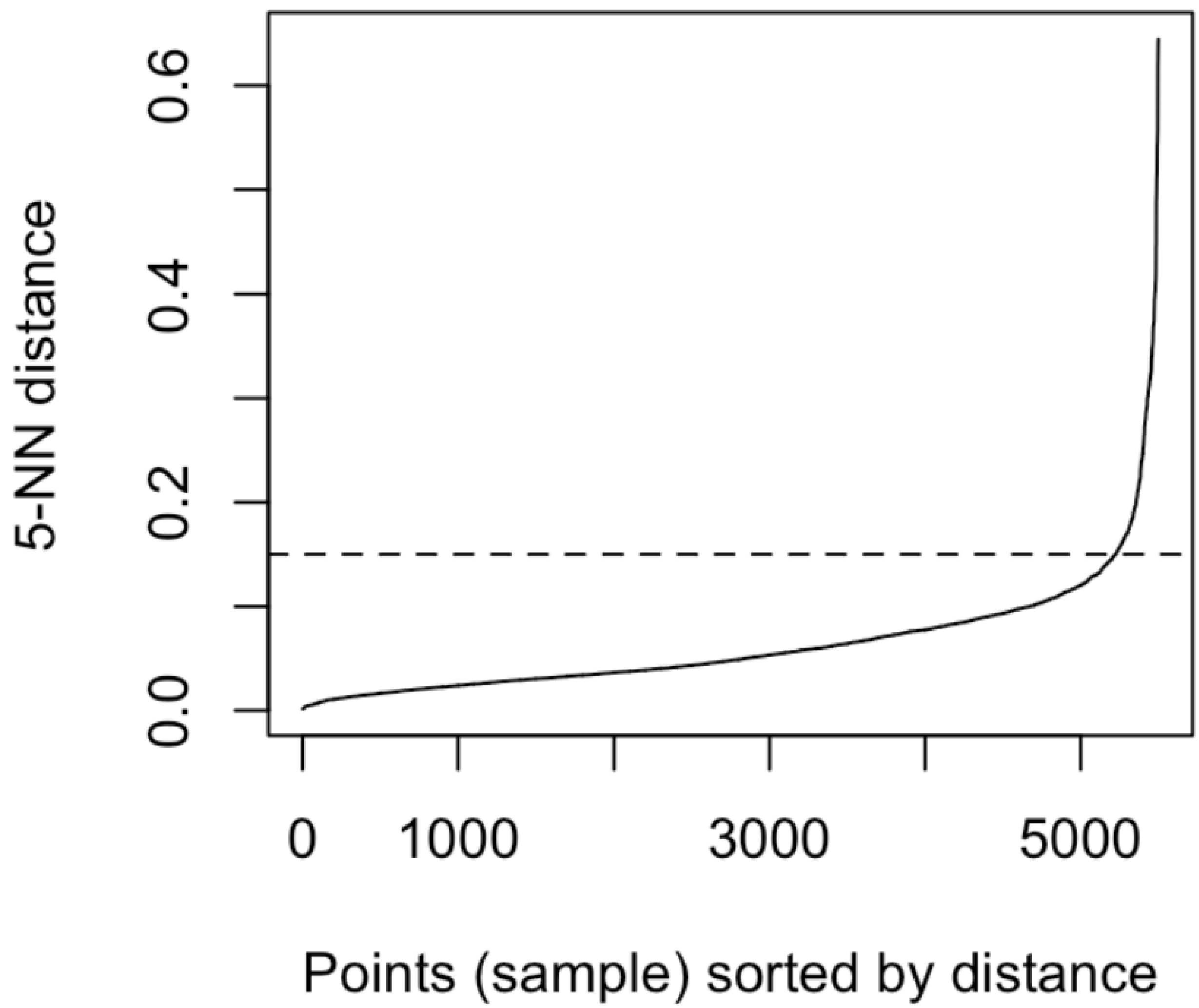
Here are a few rules of thumb for selecting the MinPts value:

- The larger the data set, the larger the value of MinPts should be
- If the data set is noisier, choose a larger value of MinPts

- Generally, MinPts should be greater than or equal to the dimensionality of the data set

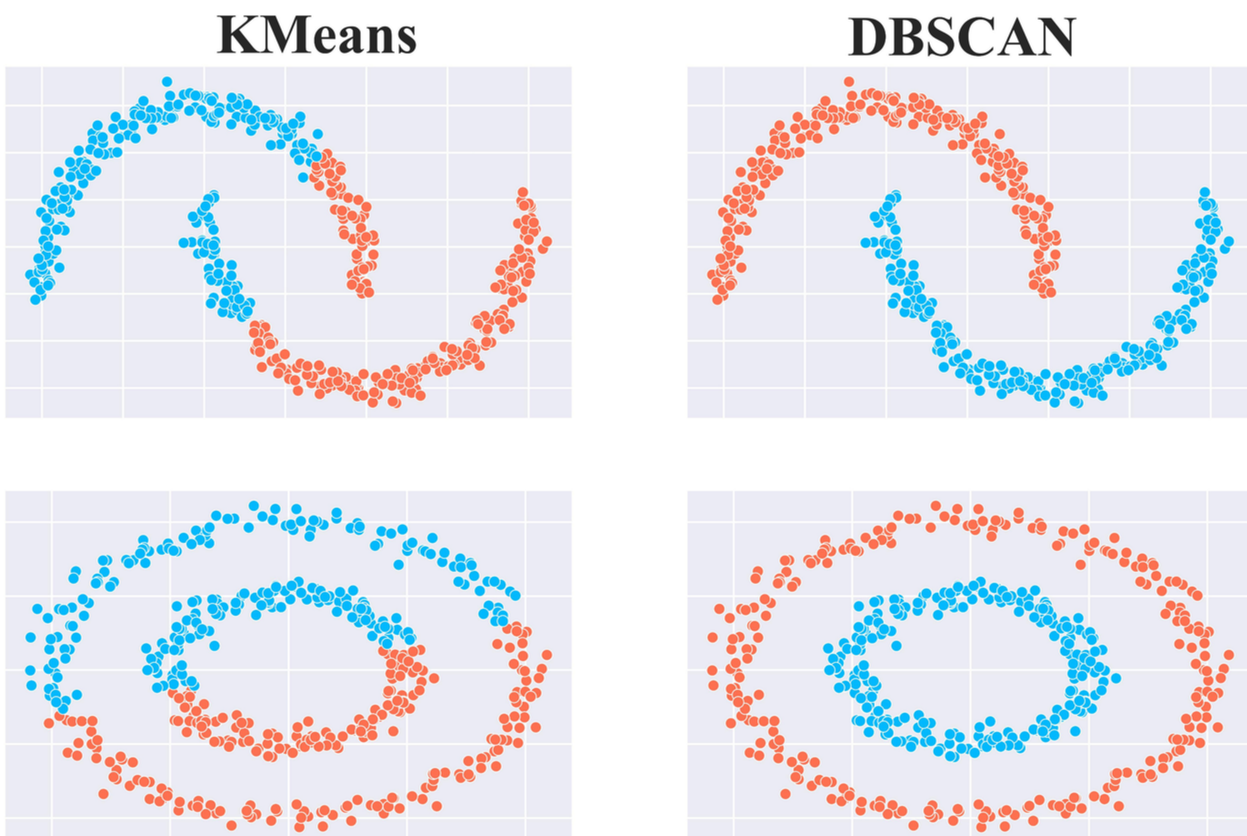
Epsilon (ε)

After you select your MinPts value, you can move on to determining ε. One technique to automatically determine the optimal ε value is described in this paper. This technique calculates the average distance between each point and its k nearest neighbors, where k = the MinPts value you selected. The average k-distances are then plotted in ascending order on a k-distance graph. You'll find the optimal value for ε at the point of maximum curvature (i.e. where the graph has the greatest slope).



Pros:

- It can find clusters of defined shapes, unlike K-means, which tends to work best with spherical and equally sized clusters.



- It can identify noise points, which can be useful for detecting outliers or anomalies.
- It does not require specifying the number of clusters in advance.

Cons:

- It is sensitive to the choice of parameters (eps and min_samples). Selecting appropriate values for these parameters can be challenging and may require domain knowledge or trial and error.
- It may not perform well with clusters of varying densities or with high-dimensional data.
- The algorithm's performance may degrade as the size of the dataset increases.

Resources:

- https://github.com/christianversloot/machine-learning-articles/blob/main/performing-dbscan-clustering-with-python-and-scikit-learn.md