Scikit-learn: DBSCAN

Cole Ballard, Christian Rodriguez, James Miller, and Temesgen Fekadu

What is DBSCAN?

DBSCAN - or Density-Based Spatial Clustering of Applications with Noise - is an unsupervised learning algorithm for clustering. It typically handles noise and outliers better than partitioning methods (such as K-means) and hierarchical methods. Whereas these methods are suitable for compact clusters that are well separated, DBSCAN can also handle clusters with arbitrary shapes (because it works by identifying clusters as areas of high density separated by areas of low density).

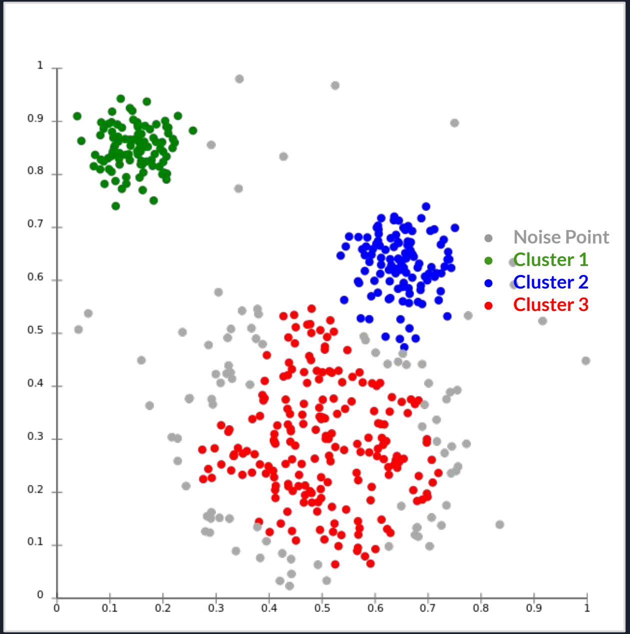
How DBSCAN Works

In a nutshell, the algorithm works by computing the distance from each individual data point to all other data points, then places all of the points into one of three categories:

Core Point: A core point is a data point that has at least n points within a specified distance.

Border Point: A border point is not within the defined distance defined to at least n points, however, it is close enough to at least one core point to be considered part of its cluster. A border point may be close to multiple core points, but it will only be included in the cluster of the closest core point.

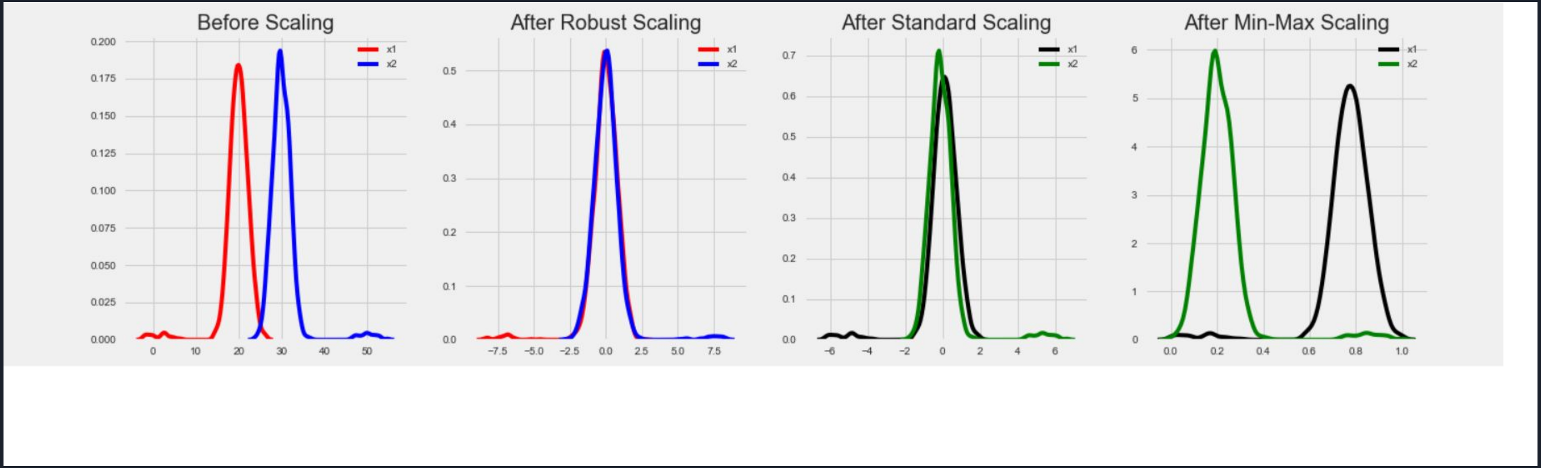
Noise Point: A noise point is not within the specified distance to any core points. As a result, it is not included in any cluster and can be considered an outlier.



Data (Pre)Processing

You’ll want to:

- Handle all null values using your preferred method
- Scale your data since the algorithm is grouping samples by distance
- Load the data into algorithm after scaling (... and tune hyperparameters)



Tunable hyperparameters

DBSCAN requires two tunable hyperparameters:

- eps (epsilon)- Defines the “neighborhood” surrounding a data point. Where the distance between 2 data points is less than or equal to eps, those data points are considered to be neighbors. Choosing an eps value is important to ensure that you are limiting the amount of outliers while also avoiding the merging of clusters. A k-distance graph is commonly used to find an appropriate eps value (explained in next slide)

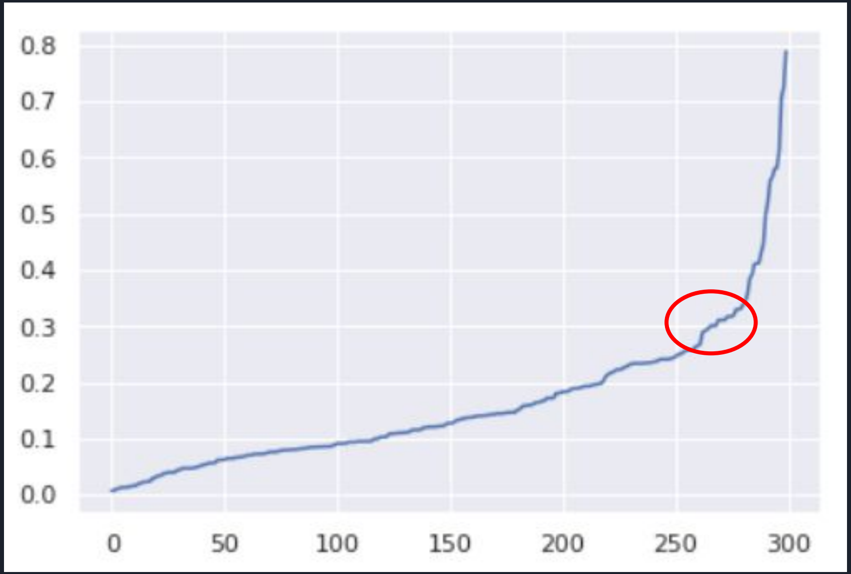
- min_samples (minimum_neighbors, MinPts, etc.) - This parameter specifies the minimum number of neighbors that must exist within a radius equal to eps. Generally, you can derive the minimum neighbors from the number of dimensions, D, in the dataset. Larger datasets require larger values of minimum_neighbors. You may use the inequality:

    - min_samples >= D + 1

K-Distance Graph

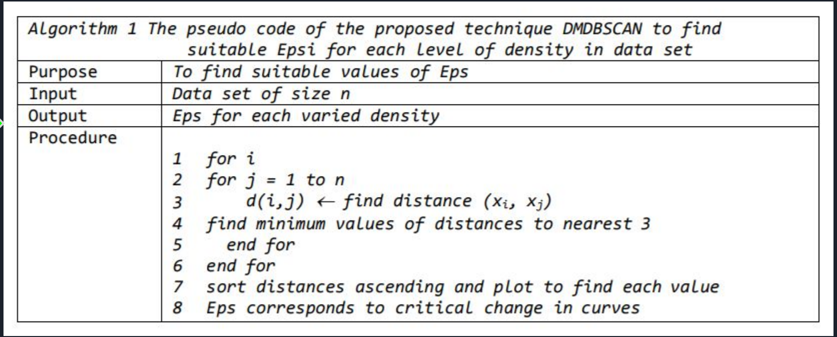
One of the best ways to determine the optimal eps value is to produce a K-distance graph

- Calculate the distance from each point to its k-nearest neighbor
- Sort the k-distances in ascending order and plot the results
- Locate the “knee” (sharp change in values) to determine the optimal eps value. In the example on the right, the knee is located around 0.3, suggesting an optimal eps value of about 0.3.



Tuning the Hyperparameters: eps

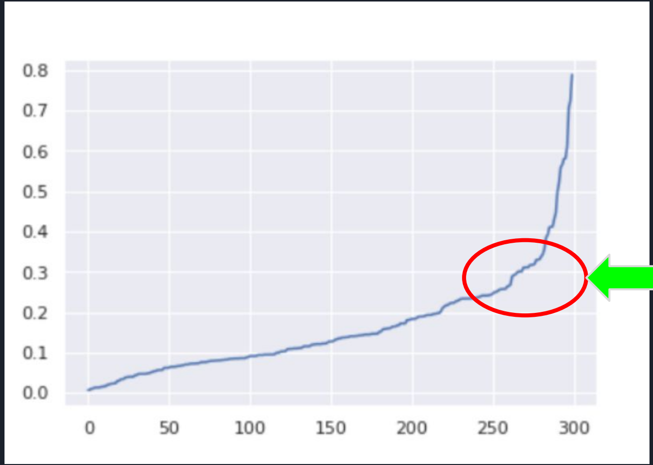
This algorithm will help you produce an optimal value for eps:



Algorithm 1 The pseudo code of the proposed technique DMDBSCAN to find suitable Epsi for each level of density in data set

Purpose	To find suitable values of Eps
Input	Data set of size n
Output	Eps for each varied density
Procedure	see below

```
1  for i
2  for j = 1 to n
3      d(i,j) ← find distance ($x_i$, $x_j$)
4  find minimum values of distances to nearest 3
5    end for
6  end for
7  sort distances ascending and plot to find each value
8  Eps corresponds to critical change in curves
```



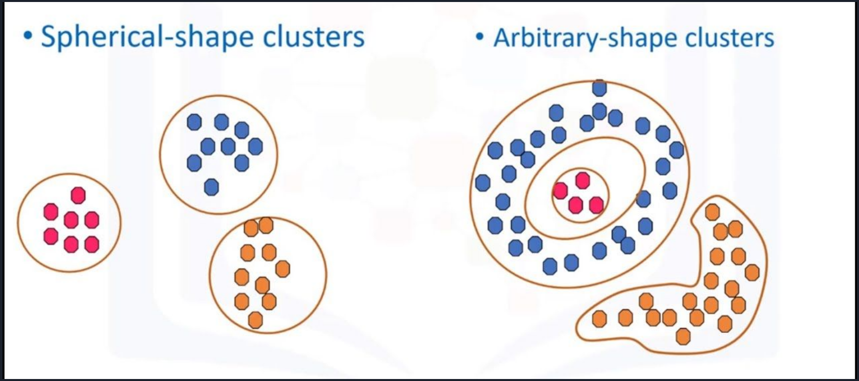
https://iopscience.iop.org/article/10.1088/1755-1315/31/1/012012/pdf

The optimal eps value typically corresponds to the point of greatest curvature in the plot produced by the above algorithm

https://towardsdatascience.com/machine-learning-clustering-dbscan-determine-the-optimal-value-for-epsilon-eps-python-example-3100091cfbc

Advantages

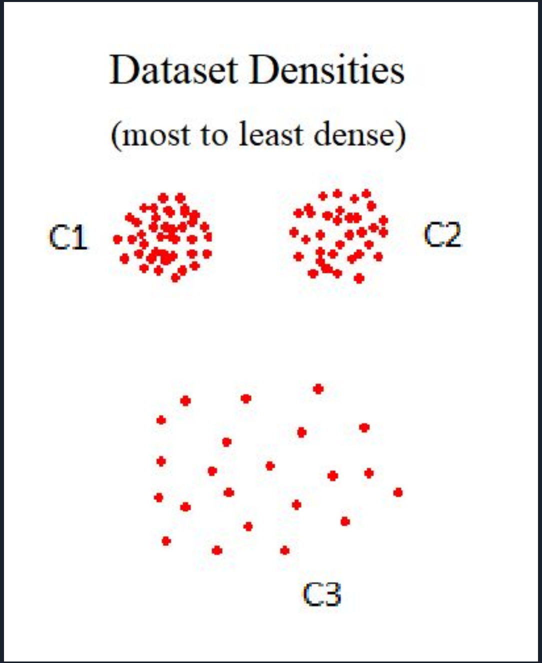
- DBSCAN does not require a specified number of clusters beforehand
- DBSCAN performs well with arbitrary shaped clusters
- DBSCAN has a notion of noise, and works well if the data includes outliers



https://ml-explained.com/blog/dbscan-explained#advantages

Disadvantages

- DBSCAN doesn’t cluster datasets with large differences in densities very well
- choosing a meaningful **eps value** (how close points should be to each other to be considered a part of a cluster) can be difficult if the data isn't well understood
- DBSCAN is not entirely deterministic, which means there are data points (usually residing on the border of a cluster) that may be considered part of more than one cluster, although this situation is very rare



https://ml-explained.com/blog/dbscan-explained#disadvantages

DBScan vs K-Means

DBScan Clustering

Comparison

- discards objects that it defines as noise
- uses density clustering - determines cluster based on density of points
- good at handling clusters of varying size and structure, and is not heavily influenced by noise and outliers
- bad at parsing high-dimensional data - Euclidean definition of density doesn’t operate well for high-dimensional data

K-Means Clustering

Process

- K centroids are randomly placed, one for each cluster
- distance of each point from each centroid is calculated
- each data point is assigned to its closest centroid, forming a cluster
- the position of K centroids are recalculated

Comparison

- generally clusters all objects
- uses prototype clustering - determines cluster based on distance of points from the prototype point
- has difficulty with non-globular clusters and clusters of multiple sizes
- good at parsing high-dimensional data, eg. a file

Conclusion

- DBSCAN is an unsupervised learning algorithm for clustering.
- Requires just two tunable hyperparameters, min_samples & eps, that are used to identify clusters.
- Can identify arbitrary shaped clusters, handles noise and outliers well.