Group 6: DBSCAN
Link2Slides
Github Repo
GoogleColabNB
Appendix

DBSCAN: **D**ensity-**B**ased **S**patial **C**lustering of **A**pplications with **N**oise

Some good resources to start with:
https://scikit-learn.org/stable/modules/clustering.html#dbscan
https://scikit-learn.org/stable/auto_examples/cluster/plot_dbscan.html#sphx-glr-auto-examples-cluster-plot-dbscan-py

What is DBSCAN?:
DBSCAN 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 Does DBSCAN Work?:
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 `min_samples` points within the distance defined by `eps` (the sub-section, *DBSCAN requires two parameters*, provides more detail on min_samples and eps).
- Border Point: A border point is not within the distance defined by `eps` to at least `min_samples` 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 be included in the cluster of the closest core point.
- Noise Point: A noise point is not within the distance defined by `eps` to any core points. As a result, it is not included in any cluster and can be considered an outlier.

DBSCAN requires two parameters (aka hyperparameters):

- eps - Short for epsilon, eps 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 (see *Insight on K-distance Graph* below)
- min_samples - commonly referred to also as minimum_neighbors, MinPts, or other similar variations. 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:
    - minimum_neighbors >= D + 1

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

- DBSCAN does not require one to specify the number of clusters beforehand.
- DBSCAN performs well with arbitrary shaped clusters.
- DBSCAN has a notion of noise, and is robust to outliers.
- Good efficiency on large databases

Disadvantages

- DBSCAN cannot cluster data-sets with large differences in densities well, since then the minPts-eps combination cannot be chosen appropriately for all clusters.
- Choosing a meaningful eps value can be difficult if the data isn't well understood.
- DBSCAN is not entirely deterministic. That's because the algorithm starts with a random point. Therefore border points that are reachable from more than one cluster can be part of either cluster.

Data Processing

- Your data should not have any null values, so use your preferred method of dealing with nulls (e.g. dropna, winsorize, impute, or any combination of those)
- Since the algorithm is grouping samples by distance, it is important that the data is scaled so as to not include the data values themselves in the parametrization of the model (e.g. comparing columns with data values represented in cm vs m would not yield the same since the model is simply looking at the numeric values)
- To make these columns/variables/features on the same scale, we'll need to scale the data using a scaling method of your choice (e.g. StandardScaler, MinMaxScaler, etc.). Note that StandardScaler is probably best so we can get all values into a standard-normal distribution
- Once the data is scaled, you can load into the algorithm and run the model (..and tune hyperparameters

Insight on K-distance Graph:
http://www.sefidian.com/2020/12/18/how-to-determine-epsilon-and-minpts-parameters-of-dbscan-clustering/
Calculate the distance of each point to its k-nearest neighbors. Plot the k-distances in ascending order. The aim is to determine the "knee", which defines the optimal `eps` value. A "knee" is a threshold where a sharp change occurs along the k-distance curve.

Histogram method:
https://stackoverflow.com/questions/43160240/how-to-plot-a-k-distance-graph-in-python

Resources:
https://www.geeksforgeeks.org/dbscan-clustering-in-ml-density-based-clustering/

https://scikit-learn.org/stable/modules/clustering.html#dbscan

https://scikit-learn.org/stable/auto_examples/cluster/plot_dbscan.html#sphx-glr-auto-examples-cluster-plot-dbscan-py

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