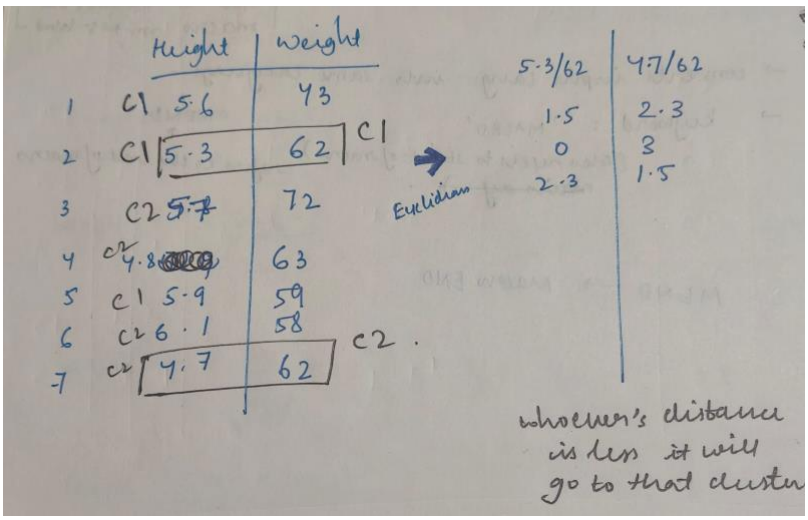
Unit 4:

K-means Clustering

K-means clustering is an unsupervised machine learning algorithm used to partition a dataset into k distinct clusters based on feature similarity. The algorithm works iteratively to assign data points to one of k clusters by minimizing the variance within clusters and maximizing the variance between clusters.



Working:

Step 1: Choose the number of clusters of k.

Step 2: Select k random points form the data as centroids.

Step 3: Assign all the points to the closest cluster centroid.

Step 4: Recompute the centroids of newly formed cluster.

Step 5: Repeat steps 3 and 4.

Advantages:

1. **Simplicity:** Easy to implement and computationally efficient for small to medium-sized datasets.

2. **Scalability:** Performs well with large datasets and is faster compared to other clustering algorithms like hierarchical clustering.
3. **Flexibility:** Works with various types of data if the appropriate distance metric is used.
4. **Interpretability:** Clusters are distinct and non-overlapping, making it easier to analyse results.
5. **Adaptability:** Can be easily adapted to a variety of feature types and dimensions.

Disadvantages:

1. **Choice of k:** Requires the number of clusters k to be predefined, which may not always be clear.
2. **Sensitivity to Initialization:** Random initialization of centroids can lead to different results; poor initialization may lead to suboptimal clustering.
3. **Sensitive to Outliers:** Outliers can distort the cluster centroids, leading to incorrect cluster assignments.
4. **Fixed Boundaries:** Assigns every point to a cluster, even if the point does not fit well with any cluster.
5. **High Dimensionality Issues:** Performance may degrade with very high-dimensional data due to the curse of dimensionality.

Numerical:

Question: Consider the following set of data given below, cluster 8 using K-means algorithm with the initial value of objects 2 and 5:

Objects	**X**	**Y**
1	2	4
2	4	6
3	6	8
4	10	4
5	12	4
6	7	5
7	6	9

Solution:

Data Points	(4, 6)	(12, 4)	cluster
(2, 4)	2.8	10	C1
C1 → (4, 6)	0	8.24	C1
(6, 8)	2.8	7.21	C1
(10, 4)	6.32	2	C2
C2 → (12, 4)	8.24	0	C2
(7, 5)	3.16	5.09	C1
(6, 9)	3.60	7.81	C1

$\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$

New centroid.

$\left(\frac{\Sigma x}{\text{Total no.}}, \frac{\Sigma y}{\text{Total no.}}\right)$

C1: (2,4), (4,6), (6,8), (7,5), (6,9) $\rightarrow \left(\frac{25}{5}, \frac{32}{5}\right) = (5, 6.4)$

C2: (10,4), (12,4) $\rightarrow$ (11, 4)

(New centroids calculated for next table)

Data Points	(5, 6.4)	(11, 4)	Cluster (previous table)	New cluster
(2, 4)	3.84	9	C1	C1
(4, 6)	1.07	7.28	C1	C1
(6, 8)	1.88	6.4	C1	C1
(10, 4)	5.54	1	C2	C2
(12, 4)	7.4	1	C2	C2
(7, 5)	2.44	4.12	C1	C1
(6, 9)	2.78	4.07	C1	C1

Fuzzy Clustering

Clustering is an unsupervised machine learning technique that divides the given data into different clusters based on their distances (similarity) from each other.

The unsupervised k-means clustering algorithm gives the values of any point lying in some particular cluster to be either as 0 or 1 i.e., either true or false. But the fuzzy logic gives the fuzzy values of any particular data point to be lying in either of the clusters. Here, in fuzzy c-means clustering, we find out the centroid of the data points and then calculate the distance of each data point from the given centroids until the clusters formed become constant. Suppose the given data points are {(1, 3), (2, 5), (6, 8), (7, 9)}

Fuzzy Clustering is a type of clustering algorithm in machine learning that allows a data point to belong to more than one cluster with different degrees of membership. Unlike traditional clustering algorithms, such as k-means or hierarchical clustering, which assign each data point to a single cluster, fuzzy clustering assigns a membership degree between 0 and 1 for each data point for each cluster.

Applications in several fields of Fuzzy clustering:

1. **Image segmentation:** Fuzzy clustering can be used to segment images by grouping pixels **with similar properties together, such as color or texture.**
2. **Pattern recognition:** Fuzzy clustering can be used to identify patterns in large datasets by grouping similar data points together.
3. **Marketing:** Fuzzy clustering can be used to segment customers based on their preferences and purchasing behaviour, allowing for more targeted marketing campaigns.
4. **Medical diagnosis:** Fuzzy clustering can be used to diagnose diseases by grouping patients with similar symptoms together.
5. **Environmental monitoring:** Fuzzy clustering can be used to identify areas of environmental concern by grouping together areas with similar pollution levels or other environmental indicators.
6. **Traffic flow analysis:** Fuzzy clustering can be used to analyse traffic flow patterns by grouping similar traffic patterns together, allowing for better traffic management and planning.

7. **Risk assessment:** Fuzzy clustering can be used to identify and quantify risks in various fields, such as finance, insurance, and engineering.

Advantages of Fuzzy Clustering:

1. **Flexibility:** Fuzzy clustering allows for overlapping clusters, which can be useful when the data has a complex structure or when there are ambiguous or overlapping class boundaries.
2. **Robustness:** Fuzzy clustering can be more robust to outliers and noise in the data, as it allows for a more gradual transition from one cluster to another.
3. **Interpretability:** Fuzzy clustering provides a more nuanced understanding of the structure of the data, as it allows for a more detailed representation of the relationships between data points and clusters.

Disadvantages of Fuzzy Clustering:

1. **Complexity:** Fuzzy clustering algorithms can be computationally more expensive than traditional clustering algorithms, as they require optimization over multiple membership degrees.
2. **Model selection:** Choosing the right number of clusters and membership functions can be challenging, and may require expert knowledge or trial and error.

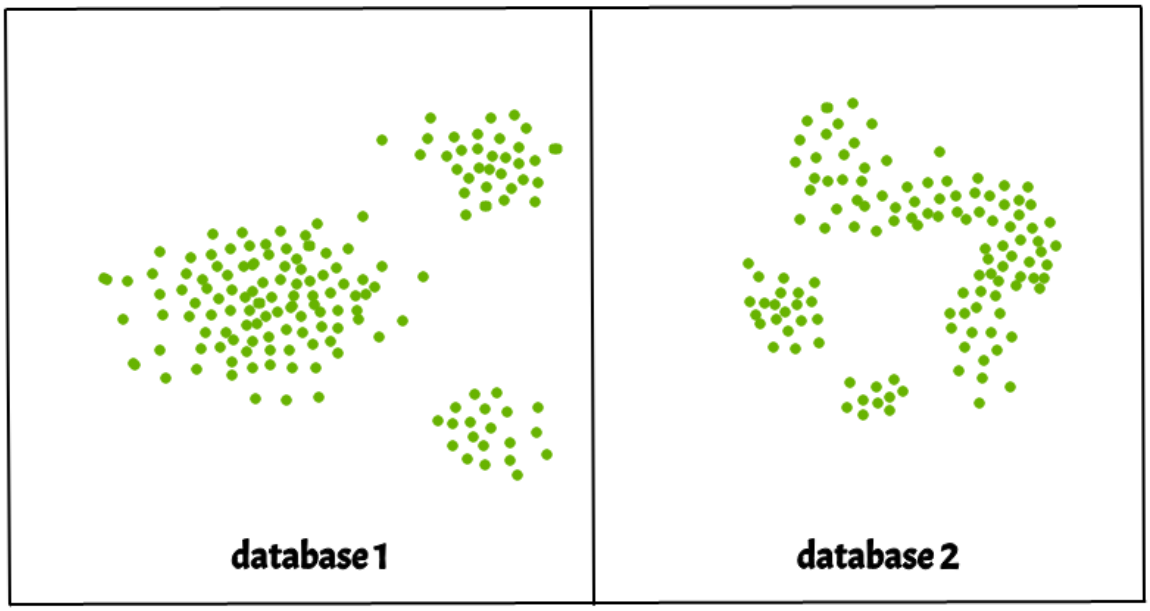
DBSCAN Clustering

Clustering analysis or simply Clustering is basically an Unsupervised learning method that divides the data points into a number of specific batches or groups, such that the data points in the same groups have similar properties and data points in different groups have different properties in some sense. It comprises many different methods based on differential evolution. E.g. K-Means (distance between points), Affinity propagation (graph distance), Mean-shift (distance between points), DBSCAN (distance between nearest points), spectral clustering (graph distance), etc.

Fundamentally, all clustering methods use the same approach i.e. first we calculate similarities and then we use it to cluster the data points into groups or batches. Here we will focus on the **Density-based spatial clustering of applications with noise** (DBSCAN) clustering method.

Density-Based Spatial Clustering of Applications with Noise (DBSCAN)

Clusters are dense regions in the data space, separated by regions of the lower density of points. The ***DBSCAN algorithm*** is based on this intuitive notion of "clusters" and "noise". The key idea is that for each point of a cluster, the neighbourhood of a given radius has to contain at least a minimum number of points.

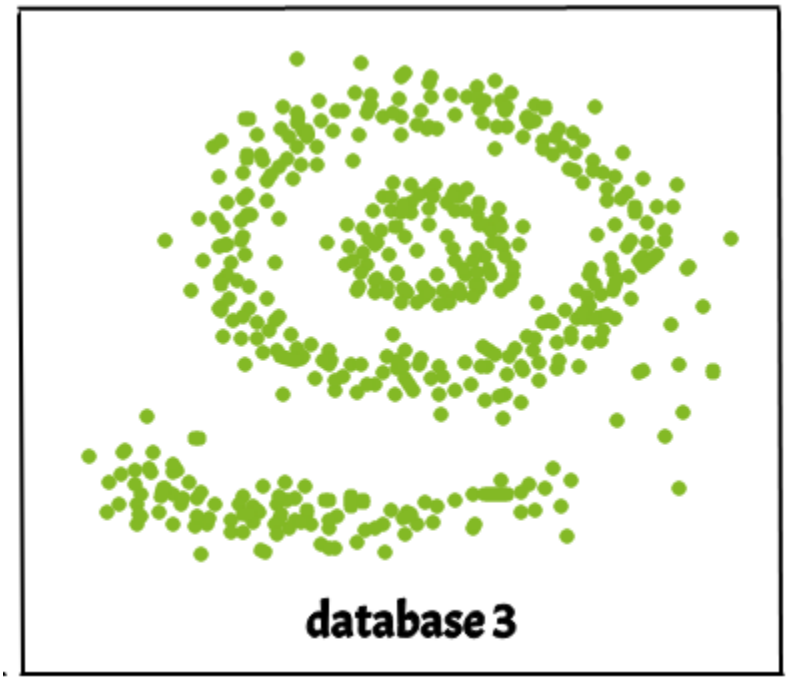


Why DBSCAN?

Partitioning methods (K-means, PAM clustering) and hierarchical clustering work for finding spherical-shaped clusters or convex clusters. In other words, they are suitable only for compact and well-separated clusters. Moreover, they are also severely affected by the presence of noise and outliers in the data.

Real-life data may contain irregularities, like:

1. Clusters can be of arbitrary shape such as those shown in the figure below.
2. Data may contain noise.



The figure above shows a data set containing non-convex shape clusters and outliers. Given such data, the k-means algorithm has difficulties in identifying these clusters with arbitrary shapes.

Parameters Required for DBSCAN Algorithm

1. **eps**: It defines the neighbourhood around a data point i.e. if the distance between two points is lower or equal to 'eps' then they are considered neighbours. If the eps value is chosen too small then a large part of the data will be considered as an outlier. If it is chosen very large then the clusters will merge and the majority of the data points will be in the same clusters. One way to find the eps value is based on the ***k-distance graph***.

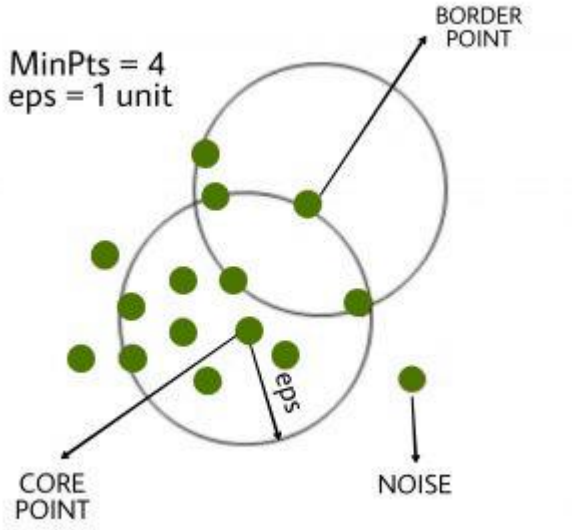
2. **MinPts**: Minimum number of neighbors (data points) within eps radius. The larger the dataset, the larger value of MinPts must be chosen. As a general rule, the minimum MinPts can be derived from the number of dimensions D in the dataset as, MinPts >= D+1. The minimum value of MinPts must be chosen at least 3.

In this algorithm, we have 3 types of data points:

Core Point: A point is a core point if it has more than MinPts points within eps.

Border Point: A point which has fewer than MinPts within eps but it is in the neighbourhood of a core point.

Noise or outlier: A point which is not a core point or border point.



Steps Used in DBSCAN Algorithm

1. Find all the neighbour points within eps and identify the core points or visited with more than Mints neighbours.
2. For each core point if it is not already assigned to a cluster, create a new cluster.
3. Find recursively all its density-connected points and assign them to the same cluster as the core point.
   A point *a* and *b* are said to be density connected if there exists a point *c* which has a sufficient number of points in its neighbours and both points *a* and *b* are within the *eps distance*. This is a chaining process. So, if *b* is a neighbour of *c*, *c* is a neighbour of *d*, and *d* is a neighbour of *e*, which in turn is neighbour of *a* implying that *b* is a neighbour of *a*.
4. Iterate through the remaining unvisited points in the dataset. Those points that do not belong to any cluster are noise.

Pseudocode For DBSCAN Clustering Algorithm

DBSCAN (dataset, eps, MinPts) {

cluster index

C = 1

for each unvisited point p in dataset {

```
        mark p as visited
        # find neighbors
        Neighbors N = find the neighboring points of p
        if |N|>=MinPts:
            N = N U N'
            if p' is not a member of any cluster:
                add p' to cluster C
}
```

Forward and Backward Feature Selection Methods

Both **forward selection** and **backward elimination** are **wrapper methods** for feature selection. They involve iteratively adding or removing features based on their contribution to model performance. Here's an in-depth look at each:

1. Forward Selection

This method starts with an empty set of features and progressively adds features that improve the model's performance the most.

Steps:

1. **Start** with an empty feature set.
2. For each iteration:
    - Add one feature at a time to the current set.
    - Train a model using the current feature set.
    - Evaluate model performance (e.g., using a metric like accuracy, RMSE, or AIC).
3. **Select** the feature that improves performance the most.
4. Repeat until adding another feature does not significantly improve the model.

Advantages:

- Simple and intuitive.
- Computationally efficient for datasets with a smaller number of features.
- Useful when only a few features are relevant.

Disadvantages:

- Ignores interactions between features initially, as it starts with individual features.
- May get stuck in a local optimum by selecting features one at a time.

Applications:

- Suitable for cases where the number of features is not excessively large.
- Used when computational resources are limited.

2. Backward Elimination

This method starts with all features and progressively removes features that contribute the least to model performance.

Steps:

1. **Start** with the full set of features.
2. For each iteration:
    - Train a model using the current feature set.
    - Evaluate the importance of each feature (e.g., p-values for regression coefficients, feature importance scores).
    - Remove the feature that has the least impact on model performance.
3. Repeat until removing additional features does not improve the model.

Advantages:

- Considers interactions between features from the beginning.
- Can work well when the initial feature set is highly correlated.

Disadvantages:

- Computationally expensive for large datasets with many features.
- Risk of overfitting when the dataset is small.

Applications:

- Effective for datasets with many features but limited observations.
- Commonly used in statistical modelling, such as regression analysis.

Comparison: Forward vs. Backward

Criteria	Forward Selection	Backward Elimination
Starting Point	Begins with an empty set of features.	Begins with all features.
Feature Addition/Removal	Adds features iteratively.	Removes features iteratively.
Computational Cost	Lower, especially with many features.	Higher for datasets with many features.
Feature Interaction	Ignores feature interactions initially.	Considers feature interactions upfront.
Best For	Small datasets with a limited number of relevant features.	Datasets where most features might be irrelevant.