

CLUSTER ANALYSIS

UNIT-IV

Cluster Analysis:

What is Cluster Analysis?

Cluster analysis, also known as **clustering** is the process of partitioning a set of data points (or objects) into Groups. Each Group is a **Cluster**, such that data points in a cluster are similar to one another, yet dissimilar to data points in other clusters.

(or)

Clustering is an **unsupervised learning** technique that group's **similar data points** together based on *their features*. **Clustering** is similar to classification. But in this case, the **class label** is **unknown**.

- The main goal of Clustering is to identify important **relationships or patterns** in the data.
- There are many different algorithms used for cluster analysis, such as k-means, hierarchical clustering, and density-based clustering.
- The different clustering methods may generate different clusters on the same data set.

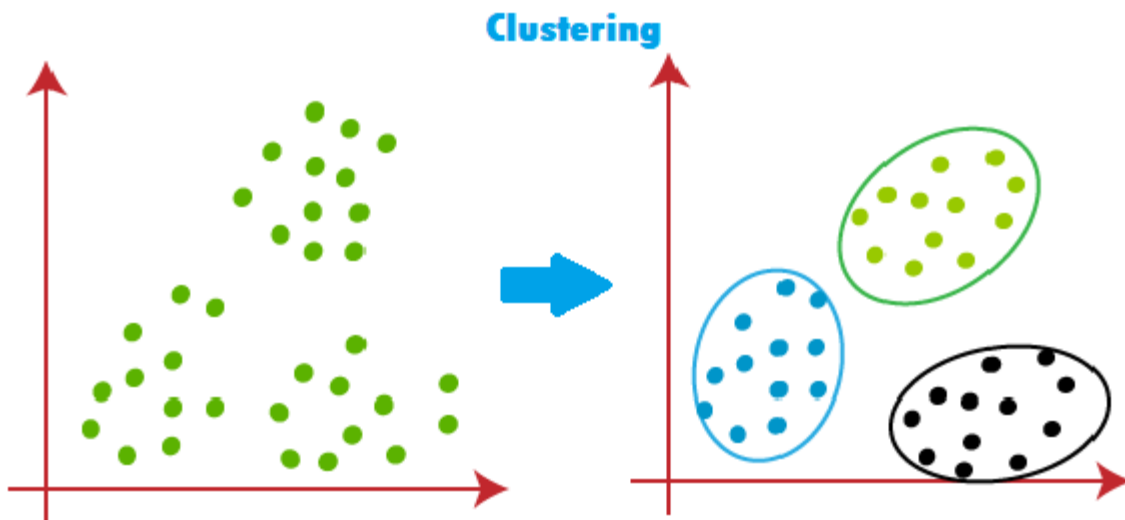


Fig: Clustering

Applications of Clustering in Data Mining

Clustering is a widely used technique in data mining and has numerous applications in various fields. Some of the common applications of clustering in data mining include -

- **Customer Segmentation**

Clustering techniques in data mining can be used to group customers with similar behavior, preferences, and purchasing patterns to create more targeted marketing campaigns.

- **Image Segmentation**

Clustering techniques in data mining can be used to segment images into different regions based on their pixel values, which can be useful for tasks such as object recognition and image compression.

- **Anomaly Detection**

Clustering techniques in data mining can be used to identify outliers or anomalies in datasets that deviate significantly from normal behavior.

- **Text Mining**

Clustering techniques in data mining can be used to group documents or texts with similar content, which can be useful for tasks such as document summarization and topic modelling.

- **Biological Data Analysis**

Clustering techniques in data mining can be used to group genes or proteins with similar characteristics or expression patterns, which can be useful for tasks such as drug discovery and disease diagnosis.

- **Recommender Systems**

Clustering techniques in data mining can be used to group users with similar interests or behavior to create more personalized recommendations for products or services.

Requirements for Cluster Analysis:

Cluster analysis is a process used to group a set of items or data points into clusters, where the items in the same cluster are more similar to each other than to those in other clusters.

The following are requirements for Cluster Analysis in data mining. Those are,

- **Scalability:** The algorithm must be able to efficiently handle large databases containing millions of data objects without a significant decline in quality or performance.
- **Ability to Deal with Different Attribute Types:** The algorithm should be flexible enough to cluster various data types, including numerical (interval-based), binary, categorical (nominal), and ordinal data.
- **Discovery of Clusters with Arbitrary Shape:** Many traditional algorithms (like K-means) are biased toward finding spherical clusters. A robust algorithm should be capable of identifying clusters of complex and arbitrary shapes.
- **High Dimensionality:** In a data warehouse or large dataset, data can have many attributes (high dimensions). The algorithm must be able to find meaningful clusters in a high-dimensional space.
- **Ability to Deal with Noisy Data:** Real-world data often contains noise, outliers, or missing/erroneous values. The clustering algorithm must be robust and insensitive to this noisy data to produce high-quality clusters.
- **Minimal Requirements for Input Parameters:** An effective algorithm should require minimal prior knowledge (domain expertise) from the user to set input parameters (like the number of clusters, k). Results should not be overly sensitive to these choices.
- **Interpretability and Usability:** The clustering results must be easy for users to comprehend, interpret, and use for decision-making in real-world applications.
- **Insensitivity to Input Order and Incremental Clustering:** The results should be stable and not change significantly based on the order in which data records are processed. Ideally, the algorithm should also support incremental updates to incorporate new data without re-computing the entire clustering from scratch.

Categorization of Major Clustering Methods:

In general, the major fundamental clustering methods can be classified into the following categories.

- Partitioning Methods
- Hierarchical Methods
- Density-based Methods
- Grid-Based Methods

Partitioning Methods:

Suppose we have a set of n objects, a partitioning method constructs k partitions of the data, where each partition represents a *cluster* and $k \leq n$. That is, it divides the data into k groups such that each group must contain at least one object.

For given the number of partitions(k) to construct, a partitioning method creates an initial partitioning. Then, it uses an **iterative relocation technique** that attempts to improve the partitioning by moving objects from one group to another.

The general criterion of a good partitioning is that objects in the same cluster are “*close*” or related to each other, whereas objects in different clusters are “*far apart*” or very different.

Hierarchical Methods:

A hierarchical method creates a *hierarchical decomposition* of the given set of data objects.

A hierarchical method can be classified based on how the hierarchical decomposition is formed.

There are two approaches here

- Agglomerative Approach

This approach, also called the **bottom-up** approach, starts with each object forming a separate group. It successively merges the objects or groups close to one another, until all the groups are merged into one.

- Divisive Approach

This approach, also called the **top-down** approach, starts with all the objects in the same cluster. In each successive iteration, a cluster is split into smaller clusters, until eventually each object is in one cluster

Density-based Methods:

Most partitioning methods are distance-based. Such methods can find only spherical-shaped clusters and encounter difficulty in discovering clusters of arbitrary shapes.

These methods have been developed based on the notion of *density*. Their general idea is to continue growing a given cluster as long as the density (*number of objects or data points*) in the “neighborhood” exceeds some threshold.

Grid-Based Methods:

In this *Grid-based methods*, the object space is divided into a finite number of *cells*, forming a grid structure. All the clustering operations are performed on the grid structure (i.e., on the cell).

The main advantage of this approach is its *fast-processing* time, which is typically independent of the number of data objects and dependent only on the number of cells in each dimension in the quantized space.

Partitioning Clustering Methods:

The simplest and most fundamental version of cluster analysis is *partitioning*, which organizes the objects of a set into several exclusive groups or clusters.

Formally, given a data set, D , of n objects, and k , the number of clusters to form, a partitioning algorithm organizes the objects into k partitions ($k \leq n$), where each partition represents a cluster.

The *most commonly* used partitioning methods are *k-means* and *k-medoids*.

• **k-Means algorithm: A Centroid-Based Technique**

K-Means Clustering is an *Unsupervised Learning* algorithm, which groups the *unlabelled* dataset into different *clusters*. Here K defines the number of pre-defined clusters that need to be created in the process, as if $K=2$, there will be two clusters, and for $K=3$, there will be three clusters, and so on.

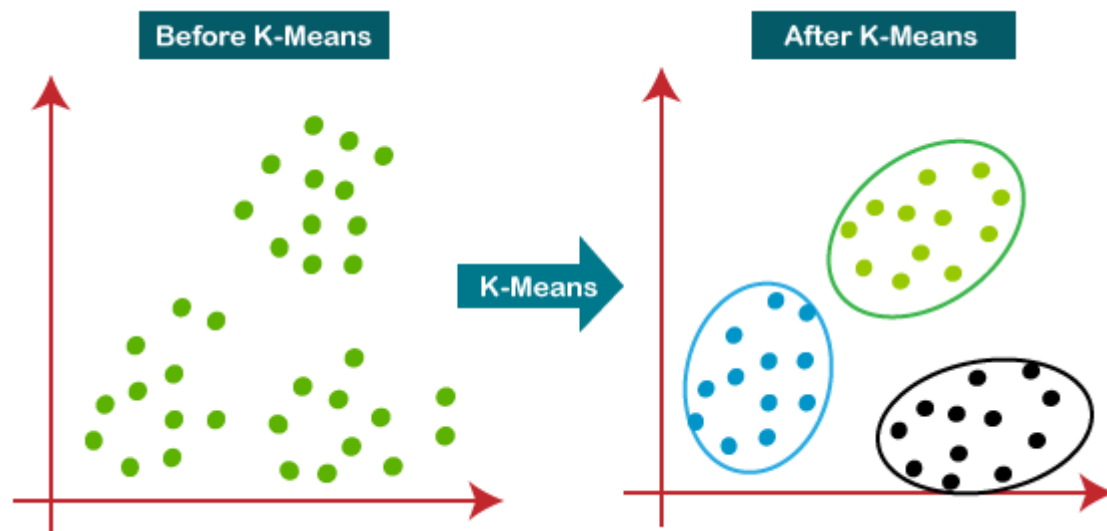
It is an *iterative algorithm* that divides the *unlabelled dataset into k different clusters* in such a way that each data point belongs only one group that has similar properties.

It allows us to cluster the data into different groups in the unlabelled dataset on its own without the need for any training.

It is a *centroid-based algorithm*, where each cluster is associated with a **centroid**. The main aim of this algorithm is to minimize the sum of distances between the data point and their corresponding clusters.

How does the K-Means Algorithm Work?

The algorithm takes the unlabelled dataset as input, divides the dataset into k-number of clusters, and repeats the process until it does not find the best clusters. The value of k should be predetermined in this algorithm.



K-Means Algorithm:

The working of the *K-Means algorithm* is explained in the below steps:

Step-1: Select the number **K** to decide the number of clusters.

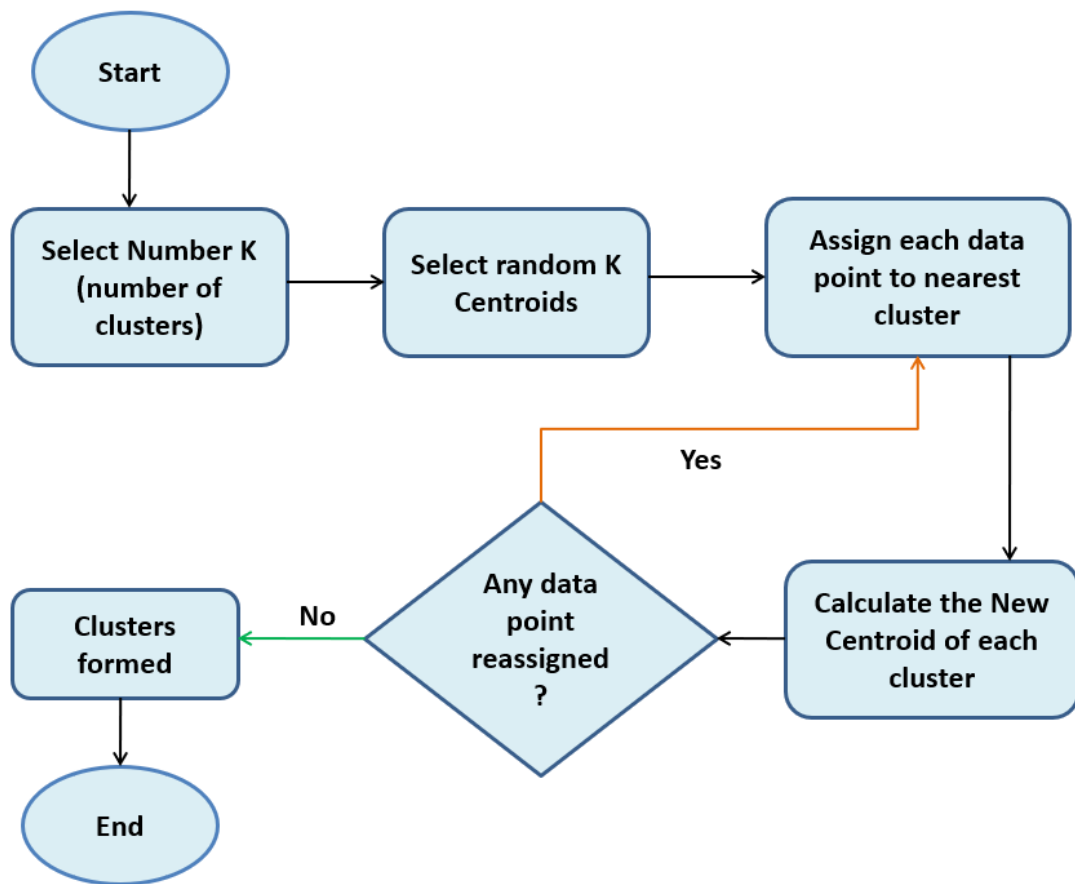
Step-2: Select random **K points or centroids**. (It can be other from the input dataset).

Step-3: Assign data point to their closest cluster centre according to the similarity or distance (Euclidean distance).

Step-4: Calculate the **centroid or mean** of all objects in each cluster.

Step-5: Repeat 3 and 4 steps, which means reassign each datapoint to the new closest centroid of each cluster.

Step-6: If any reassignment occurs, then go to step-4 else go to FINISH.



Example: Suppose we want to group the visitors to a website using just their age as follows:

16, 17, 20, 21, 22, 23, 29, 36, 41, 43, 44, 45, 61, 62

Solution:

Step 1:

First, Initialize the number of clusters i.e. K value.

Let Us $K = 2$

Step 2:

Next, Select random $K=2$ points or centroids.

i.e.

$centroid(c1) = 17$

$centroid(c2) = 23$

Step 3:

Now, calculate the distance from each data point to each centroid.

Let us assume **Distance 1** is the distances from each data point (X_i) to **centroid 1** and **Distance 2** is the distances from each data point X_i to **centroid 2**

$$\text{Distance 1} = |X_i - C_1|$$

$$\text{Distance 2} = |X_i - C_2|$$

Iteration 1:

The Initial centroids are

$$\text{centroid } (c1) = 17$$

$$\text{centroid } (c2) = 23$$

X_i	C_1	C_2	Distance 1	Distance 2	Nearest Cluster	New Centroid
16	17	23	1	7	1	16.5
17	17	23	0	6	1	
20	17	23	3	3	2	37.25
21	17	23	4	2	2	
22	17	23	5	1	2	
23	17	23	6	0	2	
29	17	23	12	6	2	
36	17	23	19	13	2	
41	17	23	24	18	2	
43	17	23	26	20	2	
44	17	23	27	21	2	
45	17	23	28	22	2	
61	17	23	44	38	2	
62	17	23	45	39	2	

Step:4

The Updated centroids are

$$\text{centroid } (c1) = \text{mean } (16, 17) = 16.5$$

$$\text{centroid } (c2) = \text{mean } (20, 21, 22, 23, 29, 36, 41, 43, 44, 45, 61, 62) = 37.25$$

Iteration 2:

x_i	c_1	c_2	Distance 1	Distance 2	Nearest Cluster	New Centroid
16	16.5	37.25	0.5	21.25	1	19.83
17	16.5	37.25	0.5	20.25	1	
20	16.5	37.25	3.5	17.25	1	
21	16.5	37.25	4.5	16.25	1	
22	16.5	37.25	5.5	15.25	1	
23	16.5	37.25	6.5	14.25	1	
29	16.5	37.25	12.5	8.25	2	45.12
36	16.5	37.25	19.5	1.25	2	
41	16.5	37.25	24.5	3.75	2	
43	16.5	37.25	26.5	5.75	2	
44	16.5	37.25	27.5	6.75	2	
45	16.5	37.25	28.5	7.75	2	
61	16.5	37.25	44.5	23.75	2	
62	16.5	37.25	45.5	24.75	2	

The Updated centroids are

$$\text{centroid } (c1) = 19.83$$

$$\text{centroid } (c2) = 45.12$$

Iteration 3:

x_i	c_1	c_2	Distance 1	Distance 2	Nearest Cluster	New Centroid
16	19.83	45.12	3.83	29.12	1	21.14
17	19.83	45.12	2.83	28.12	1	
20	19.83	45.12	0.17	25.12	1	
21	19.83	45.12	1.17	24.12	1	
22	19.83	45.12	2.17	23.12	1	
23	19.83	45.12	3.17	22.12	1	
29	19.83	45.12	9.17	16.12	1	47.42
36	19.83	45.12	16.17	9.12	2	
41	19.83	45.12	21.17	4.12	2	
43	19.83	45.12	23.17	2.12	2	
44	19.83	45.12	24.17	1.12	2	
45	19.83	45.12	25.17	0.12	2	
61	19.83	45.12	41.17	15.88	2	
62	19.83	45.12	42.17	16.88	2	

The Updated centroids are

$$\text{centroid } (c1) = 21.14$$

$$\text{centroid } (c2) = 47.42$$

Iteration 4:

x_i	c_1	c_2	Distance 1	Distance 2	Nearest Cluster	New Centroid
16	21.14	47.42	5.14	31.42	1	21.14
17	21.14	47.42	4.14	30.42	1	
20	21.14	47.42	1.14	27.42	1	
21	21.14	47.42	0.14	26.42	1	
22	21.14	47.42	0.86	25.42	1	
23	21.14	47.42	1.86	24.42	1	
29	21.14	47.42	7.86	18.42	1	
36	21.14	47.42	14.86	11.42	2	47.42
41	21.14	47.42	19.86	6.42	2	
43	21.14	47.42	21.86	4.42	2	
44	21.14	47.42	22.86	3.42	2	
45	21.14	47.42	23.86	2.42	2	
61	21.14	47.42	39.86	13.58	2	
62	21.14	47.42	40.86	14.58	2	

No change between **iterations 3 and 4** has been noted. By using clustering, 2 groups have been identified 16-29 and 36-62.

The resultant clusters are

Cluster 1: 16, 17, 20, 21, 22, 23, 29

Cluster 2: 36, 41, 43, 44, 45, 61, 62

K-Medoids clustering Algorithm (or) (PAM-Partitioning Around Method):

K-Medoids is an unsupervised clustering algorithm using the partition method of clustering. It is an improvised version of the K-Means clustering algorithm especially used to handle outlier data. It requires unlabeled data to work with.

The partitioning will be carried on such that:

- Each cluster must have at least one object
- An object must belong to only one cluster

In the K-Medoids algorithm, *each data point* is called a medoid. The medoids serve as cluster centers. The medoid is a point such that its *sum of the distance from all other points* in the same cluster is *minimum*. For distance, any suitable metric like *Euclidian distance* or *Manhattan distance* can be used.

K-Medoids Algorithm:

The working of the K-Medoids algorithm is explained in the below steps:

Step-1: Choose k number of random points from the data and consider these k points are the *initial medoids*.

Step-2: For all the remaining data points, *calculate the distance* from *each medoid* and assign it to the cluster with the nearest medoid.

Step-3: Calculate the total cost (Sum of all the distances from all the data points to the medoids)

Step-4: Select k random points as the new medoids and swap the previous medoids with them. Repeat 2 and 3 steps.

Step-5: If the total cost of the new medoid is *less than* that of the previous medoid, make the new medoid permanent and repeat step 4.

Step-6: If the total cost of the new medoid is *greater than* the cost of the previous medoid, undo the swap and repeat step 4.

Step-7: The Repetitions have to continue until **no change** is encountered with new medoids to classify data points.

Example: Suppose we want to group the visitors to a website using just their age as follows:

16, 17, 21, 23, 29, 36, 41, 43

Solution:

Select random **K=2** points or *medoids*. i.e.

$$\text{medoid}(m_1) = 17$$

$$\text{medoid}(m_2) = 23$$

Now, calculate the distance from each data point to each *medoid*.

Let us assume **Distance 1** is the distances from each data point (X_i) to *medoid 1* and **Distance 2** is the distances from each data point (X_i) to *medoid 2*

$$\text{Distance 1} = |X_i - m_1|$$

$$\text{Distance 2} = |X_i - m_2|$$

Iteration 1:

The Initial *medoids* are

$$\text{medoid1}(m_1) = 17$$

$$\text{medoid2}(m_2) = 23$$

x_i	m_1	m_2	Distance 1	Distance 2	Nearest Cluster	Cost of Cluster	Total Cost
16	17	23	1	7	1	1	60
17	17	23	0	6	1		
21	17	23	4	2	2	59	
23	17	23	6	0	2		
29	17	23	12	6	2		
36	17	23	19	13	2		
41	17	23	24	18	2		
43	17	23	26	20	2		

$$\text{Cost of cluster1} = (1+0) = 1$$

$$\text{Cost of cluster2} = (2+0+6+13+18+20) = 59$$

$$\text{Total Cost} = 1+59 = \mathbf{60}$$

Iteration-2:

Select random *medoids*

$$\text{medoid1}(m_1) = 21$$

$$\text{medoid2}(m_2) = 36$$

x_i	m_1	m_2	Distance 1	Distance 2	Nearest Cluster	Cost of Cluster	Total Cost
16	21	36	5	20	1	11	30
17	21	36	4	19	1		
21	21	36	0	15	1		
23	21	36	2	13	1		
29	21	36	8	7	2	19	
36	21	36	15	0	2		
41	21	36	20	5	2		
43	21	36	22	7	2		

Cost of cluster1 = $(5+4+0+2) = 11$

Cost of cluster2 = $(7+0+5+7) = 19$

Total Cost = $11+19 = 30$

The **total cost** of the *new medoids* is **less than** that of the *previous medoids*, make the new medoids **permanent**.

Iteration-3:

Select random *medoids*

medoid1 (m_1) = 21

medoid2 (m_2) = 41

x_i	m_1	m_2	Distance 1	Distance 2	Nearest Cluster	Cost of Cluster	Total Cost
16	21	41	5	25	1	19	26
17	21	41	4	24	1		
21	21	41	0	20	1		
23	21	41	2	18	1		
29	21	41	8	12	1		
36	21	41	15	5	2	7	
41	21	41	20	0	2		
43	21	41	22	2	2		

Cost of cluster1 = $(5+4+0+2+8) = 19$

Cost of cluster2 = $(5+0+2) = 7$

Total Cost = $19 + 7 = 26$

The **total cost** of the *new medoids* is **less than** that of the *previous medoids*, make the new medoids **permanent**.

Iteration-4:

Select random *medoids*

$$\text{medoid1 (m1)} = 23$$

$$\text{medoid2 (m2)} = 41$$

x_i	m_1	m_2	Distance 1	Distance 2	Nearest Cluster	Cost of Cluster	Total Cost
16	23	41	7	25	1	21	28
17	23	41	6	24	1		
21	23	41	2	20	1		
23	23	41	0	18	1		
29	23	41	6	12	1		
36	23	41	13	5	2	7	
41	23	41	18	0	2		
43	23	41	20	2	2		

$$\text{Cost of cluster1} = (7+6+2+0+6) = 21$$

$$\text{Cost of cluster2} = (5+0+2) = 7$$

$$\text{Total Cost} = 21 + 7 = \mathbf{28}$$

The **total cost** of the new medoid is **greater than** the cost of the previous medoid, undo the swap and Repeat the same process until **no change**.

After completion of the above process, the finalized/best medoids are:

$$\text{medoid1 (m1)} = 21$$

$$\text{medoid2 (m2)} = 41$$

Hence, the **resultant clusters** are:

$$\text{Cluster 1} = (16, 17, 21, 23, 29)$$

$$\text{Cluster 2} = (36, 41, 43)$$

Exercise: Implement cluster analysis in the following dataset using **k-medoid** algorithm to find the best clusters.

X	Y
5	4
7	7
1	3
8	6
4	9

K-Means vs. K-Medoids:

K-Means	K-Medoid
Metric of similarity: Euclidian Distance	Metric of similarity: Manhattan Distance
Clustering is done based on distance from centroids .	Clustering is done based on distance from medoids .
A centroid can be a data point or some other point in the cluster	A medoid is always a data point in the cluster.
Can't cope with outlier data	Can manage outlier data too
Sometimes, outlier sensitivity can turn out to be useful	Tendency to ignore meaningful clusters in outlier data

Hierarchical Clustering Methods:

Hierarchical clustering is another unsupervised learning algorithm, which is used to group the unlabelled datasets into a cluster.

Hierarchical clustering is a method of cluster analysis in data mining that creates a hierarchical representation of the clusters in a dataset. The method starts by treating each data point as a separate cluster and then iteratively combines the closest clusters until a stopping criterion is reached.

The hierarchical clustering method works by combining data objects into a tree of clusters. The result of hierarchical clustering is a *tree-like structure*, and this tree-like structure is known as the *dendrogram*, which illustrates the hierarchical relationships among the clusters.

Why hierarchical clustering?

We have seen in the K-means clustering that there are some challenges with this algorithm, which are a *predetermined number* of clusters, and it always tries to create the clusters of the *same size*. To solve these two challenges, we can opt for the hierarchical clustering algorithm because, in this algorithm, we do not need to have knowledge about the predefined number of clusters.

Types of Hierarchical Clustering

Basically, there are two types of hierarchical Clustering:

- Agglomerative Hierarchical Clustering (AHC)
- Divisive Hierarchical clustering (DHC)

Agglomerative Hierarchical Clustering (AHC):

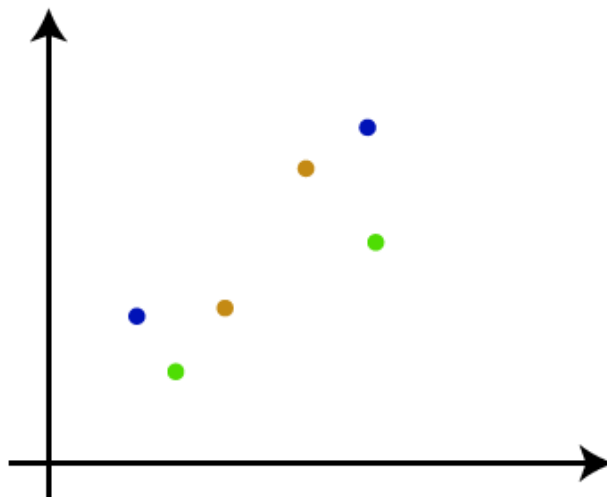
In this technique, initially *each data point* is considered as an *individual cluster*. At each iteration, the similar datapoints / clusters merge with other clusters until one cluster.

In other words, the Agglomerative clustering follows a *bottom-up approach*, it means, this algorithm considers each datapoint as a single cluster at the beginning, and then start combining the closest pair of clusters together. It does this until all the clusters are merged into a single cluster that contains all the datapoints.

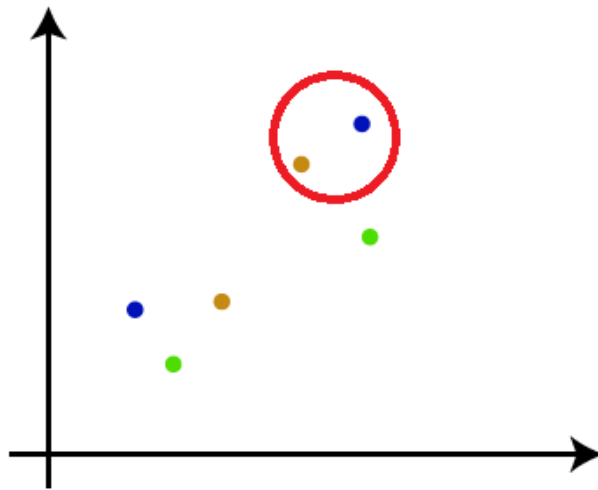
How the Agglomerative Hierarchical clustering Work?

The working of the AHC algorithm can be explained using the below steps:

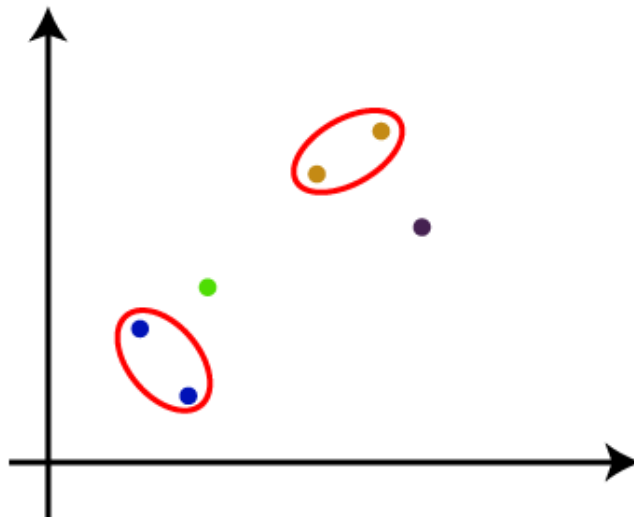
Step-1: Create each data point as a *single cluster*. Let us say there are N data points, so the number of clusters will also be N .



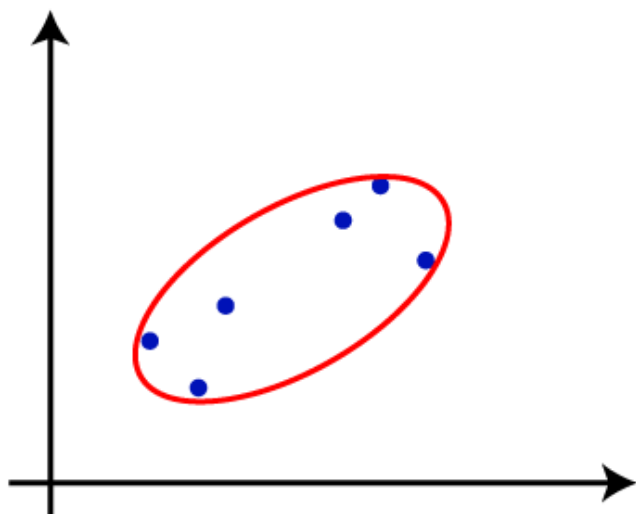
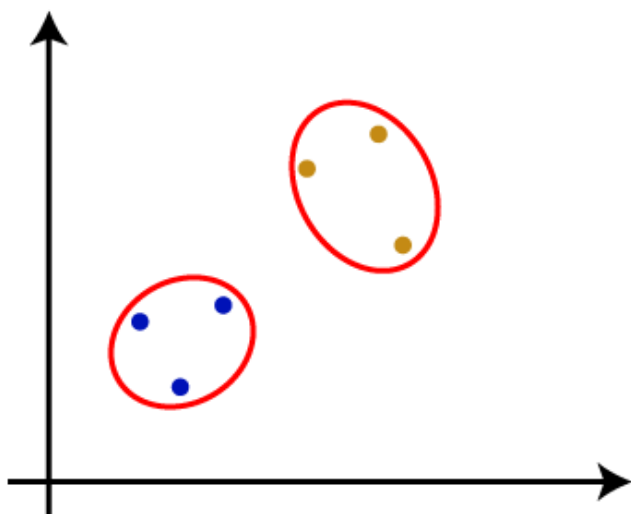
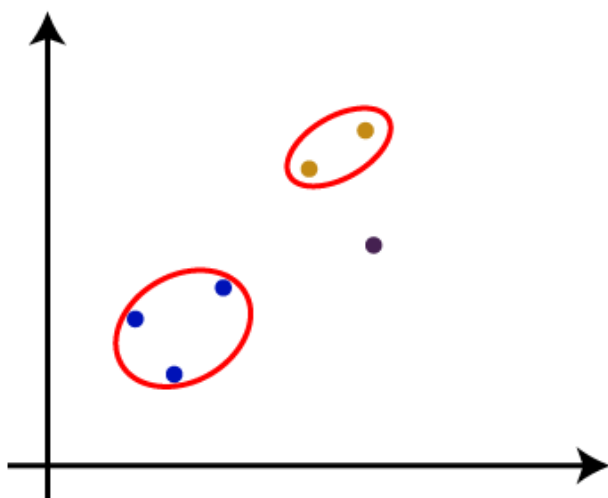
Step-2: Take two closest data points or clusters and *merge them* to form one cluster. So, there will now be $N-1$ clusters.



Step-3: Again, take the two closest clusters and merge them together to form one cluster. There will be $N-2$ clusters.

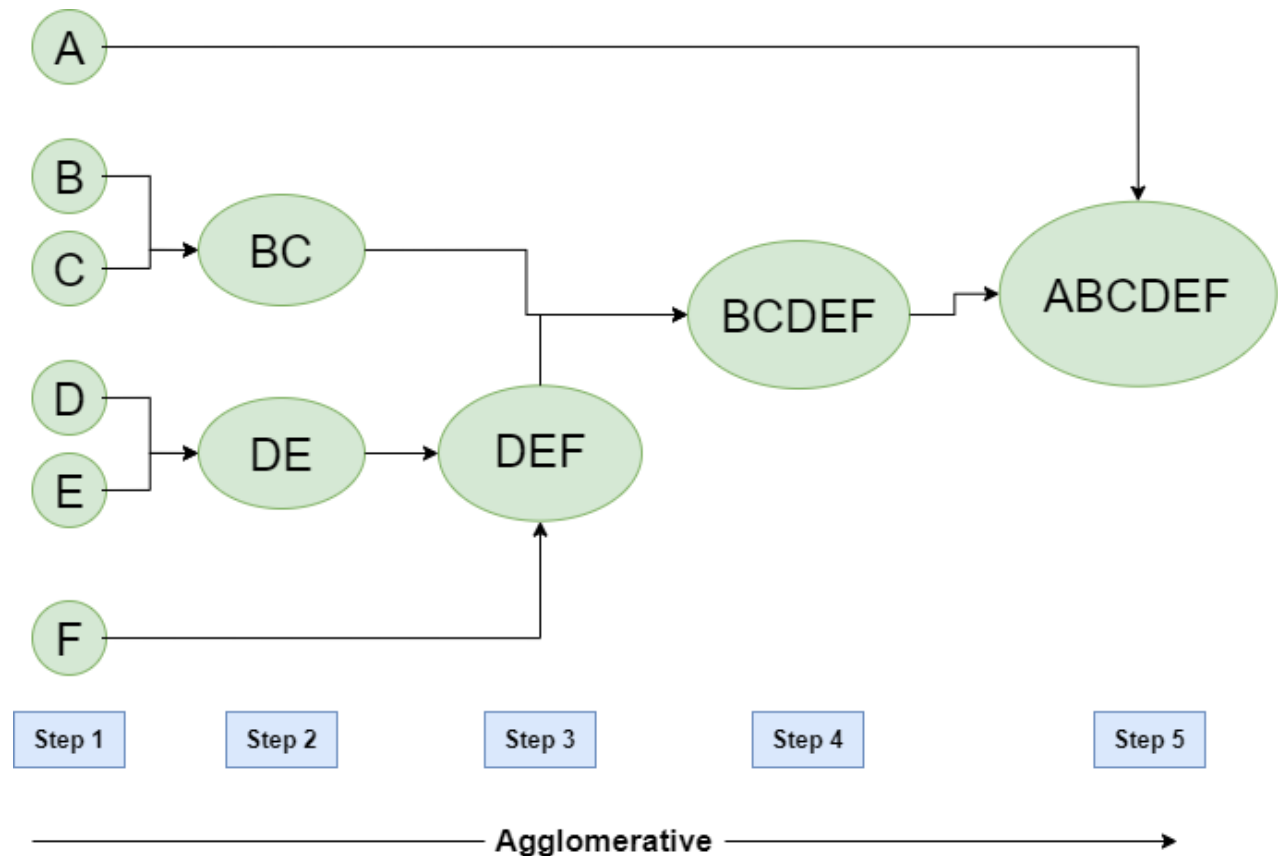


Step-4: Repeat Step 3 until only *one cluster* left. So, we will get the following clusters. Consider the below images:



Example:

Let us assume, we have six data points A, B, C, D, E, and F.



Step-1: Consider each datapoint(alphabet) as a single cluster and *calculate the distance* of one cluster from all the other clusters.

Step-2: In this step, comparable clusters are merged together to form a single cluster. Let's say cluster (B) and cluster (C) are very similar to each other therefore we merge them in the second step similarly to cluster (D) and (E) and at last, we get the clusters [(A), (BC), (DE), (F)]

Step-3: We recalculate the distances according to the algorithm and merge the two nearest clusters([(DE), (F)]) together to form new clusters as [(A), (BC), (DEF)]

Step-4: Repeating the same process; The clusters DEF and BC are comparable and merged together to form a new cluster. We're now left with clusters [(A), (BCDEF)].

Step-5: At last, the two remaining clusters are merged together to form a single cluster [(ABCDEF)].

Divisive Hierarchical clustering (DHC):

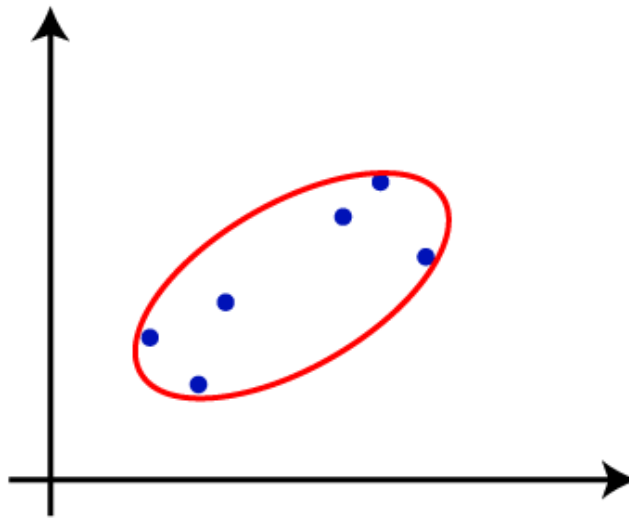
In simple words, we can say that the Divisive Hierarchical clustering is exactly the *opposite* of the Agglomerative Hierarchical clustering. In Divisive Hierarchical clustering, we consider all the data points as a *single cluster* and in each iteration, we *separate* the data points from the cluster which are *not similar*. Each data point which is separated is considered as an individual cluster. In the end, we'll be left with n clusters.

The Divisive Hierarchical clustering follows a *top - down approach*.

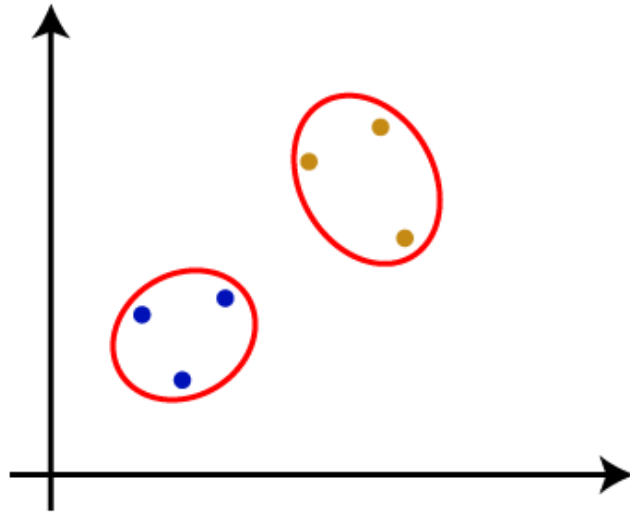
How the Divisive Hierarchical clustering Work?

The working of the DHC algorithm can be explained using the below steps:

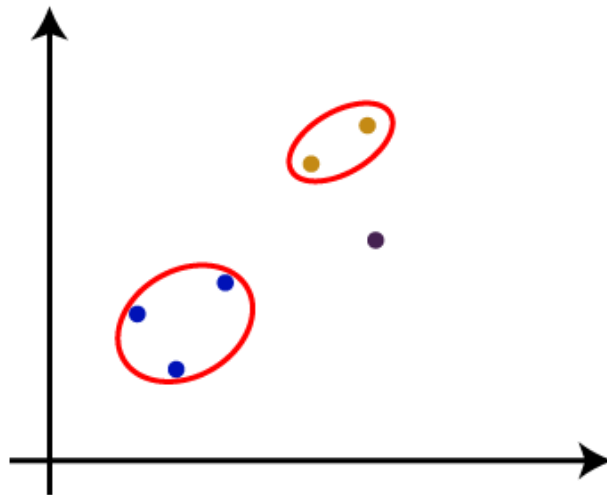
Step-1: Consider all the datapoints as a *single cluster*. Let us say there are N data points, so the number of clusters is 1.



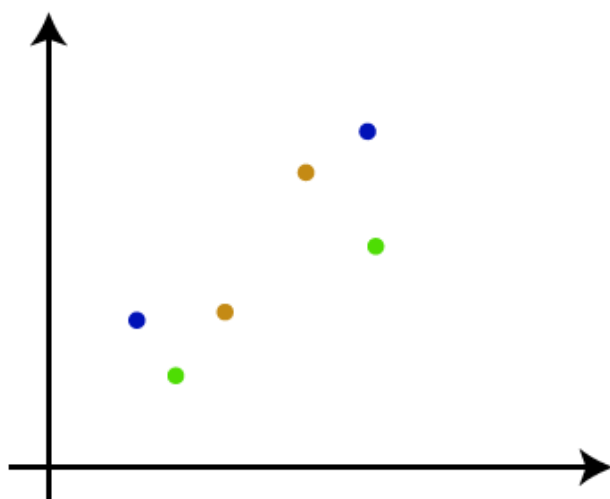
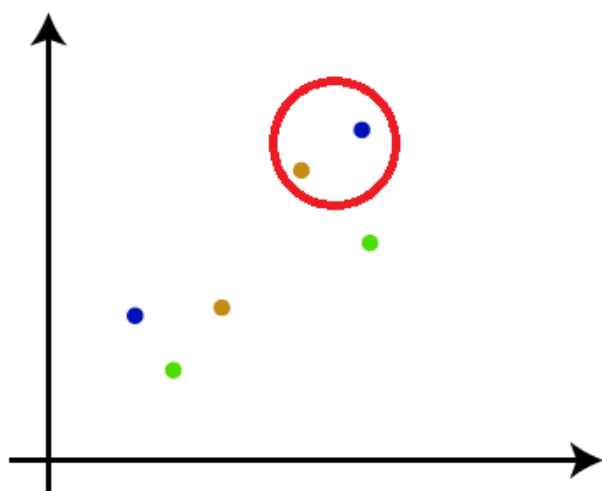
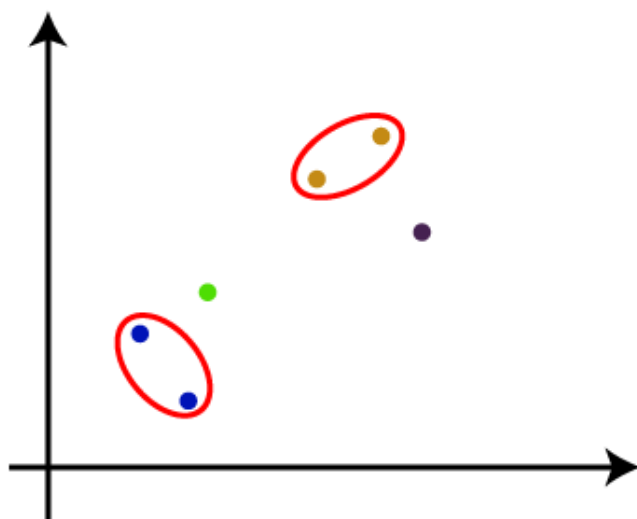
Step-2: *Separate* the datapoints which are not similar into clusters. So, there will now be 2 clusters.



Step-3: Again, *Separate* the datapoints which are not similar into clusters. So, there will now be 3 clusters.

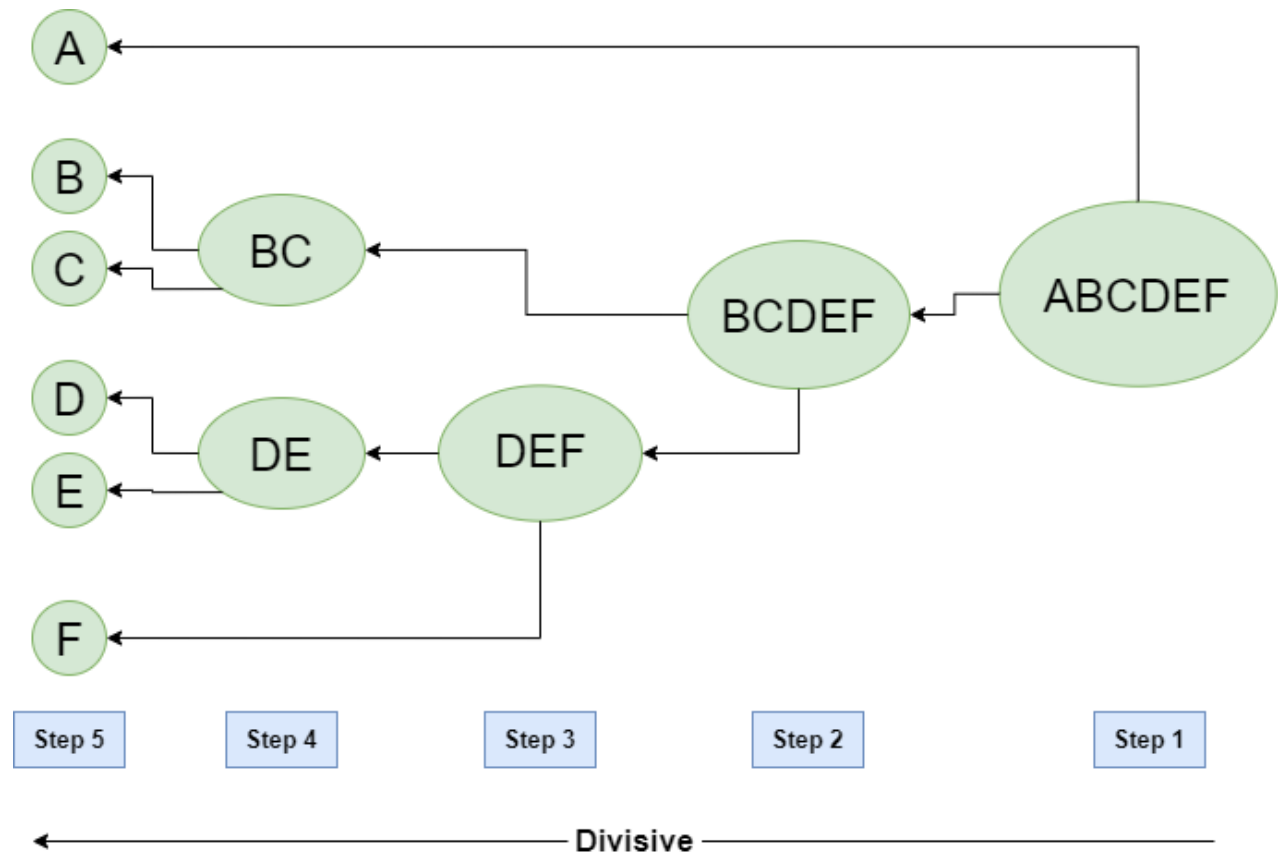


Step-4: Repeat Step 3 until left with **N** clusters. So, we will get the following clusters. Consider the below images:



Example:

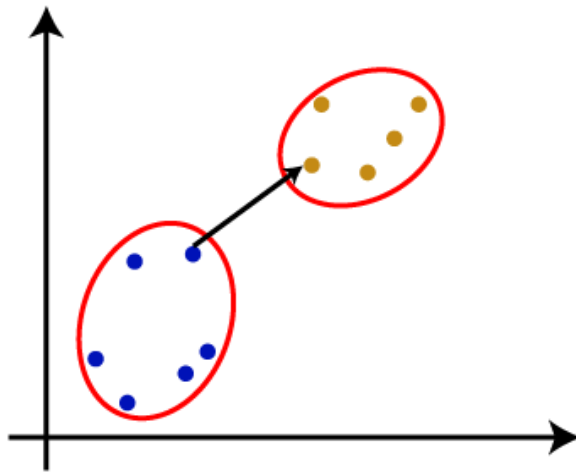
Let us assume, we have six data points A, B, C, D, E, and F.



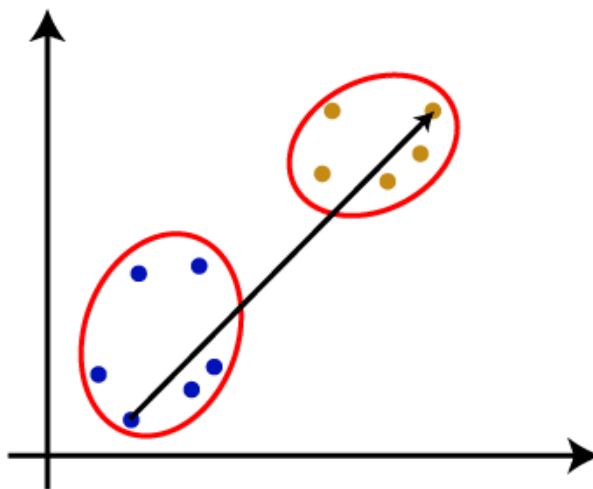
Measure for the distance between two clusters

As we have seen, the *closest distance* between the two clusters is crucial for the hierarchical clustering. There are various ways to calculate the distance between two clusters, and these ways decide the rule for clustering. These measures are called *Linkage methods*. Some of the popular linkage methods are given below:

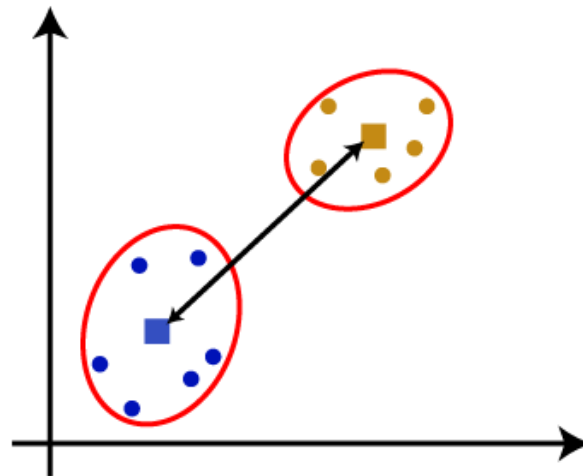
1. **Single Linkage:** It is the Shortest Distance between the closest points of the clusters.
Consider the below image:



2. **Complete Linkage:** It is the farthest distance between the two points of two different clusters. It is one of the popular linkage methods as it forms tighter clusters than single-linkage.



3. **Average Linkage:** It is the linkage method in which the distance between each pair of datasets is added up and then divided by the total number of datasets to calculate the average distance between two clusters. It is also one of the most popular linkage methods.
4. **Centroid Linkage:** It is the linkage method in which the distance between the centroid of the clusters is calculated. Consider the below image:



From the above-given approaches, we can apply any of them according to the type of problem or business requirement.

Density-Based Clustering Methods:

Density-based clustering is an unsupervised learning algorithm that groups similar data points in a dataset based on *their density*. Where the density refers the number of data points within a specified radius (epsilon ϵ) of a given data point.

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

One commonly used density-based clustering algorithm is **DBSCAN** (Density-Based Spatial Clustering of Applications with Noise). DBSCAN groups together data points that are close to each other based on a distance measure and a density threshold.

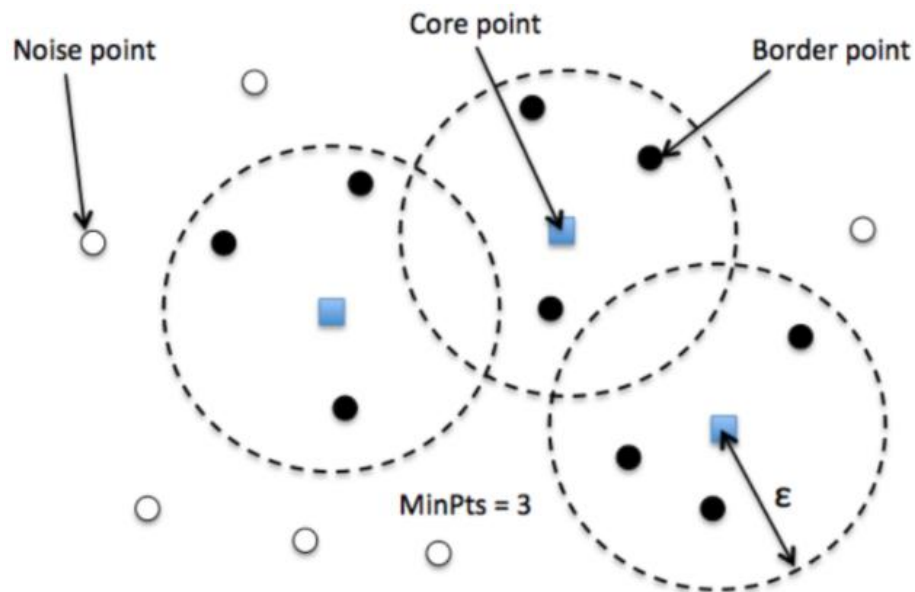
It can discover clusters of different *shapes* and *sizes* from a large amount of data, which is containing noise and outliers.

The DBSCAN algorithm uses two parameters:

- **eps (ϵ):** The *eps* parameter defines the radius of the neighbours around a datapoint x . (or) It is a distance measure that will be used to locate the points in the neighbourhood of any point.
- **MinPts:** The MinPts parameter is the minimum number of neighbours (threshold) within the “eps” radius.

In density-based clustering, the number of clusters does not need to be predetermined, but the values of **MinPts** and **eps** do.

In the DBSCAN (Density-Based Spatial Clustering of Applications with Noise) algorithm, there are three types of points: *core points*, *border points*, and *noise points*.



- **Core point:** A core point is a point that has at least MinPts within its epsilon neighbourhood. In other words, a core point has enough nearby neighbours to be considered as part of a cluster.
- **Border point:** A border point is a point that is not a core point, but it is within the epsilon neighbourhood of a core point. Border points can be considered as part of a cluster, but they are not as strongly connected to the cluster as core points.
- **Noise point:** A noise point is a point that does not have any core points within its epsilon neighbourhood. Noise points are often considered outliers and are not part of any cluster.

DBSCAN algorithm:

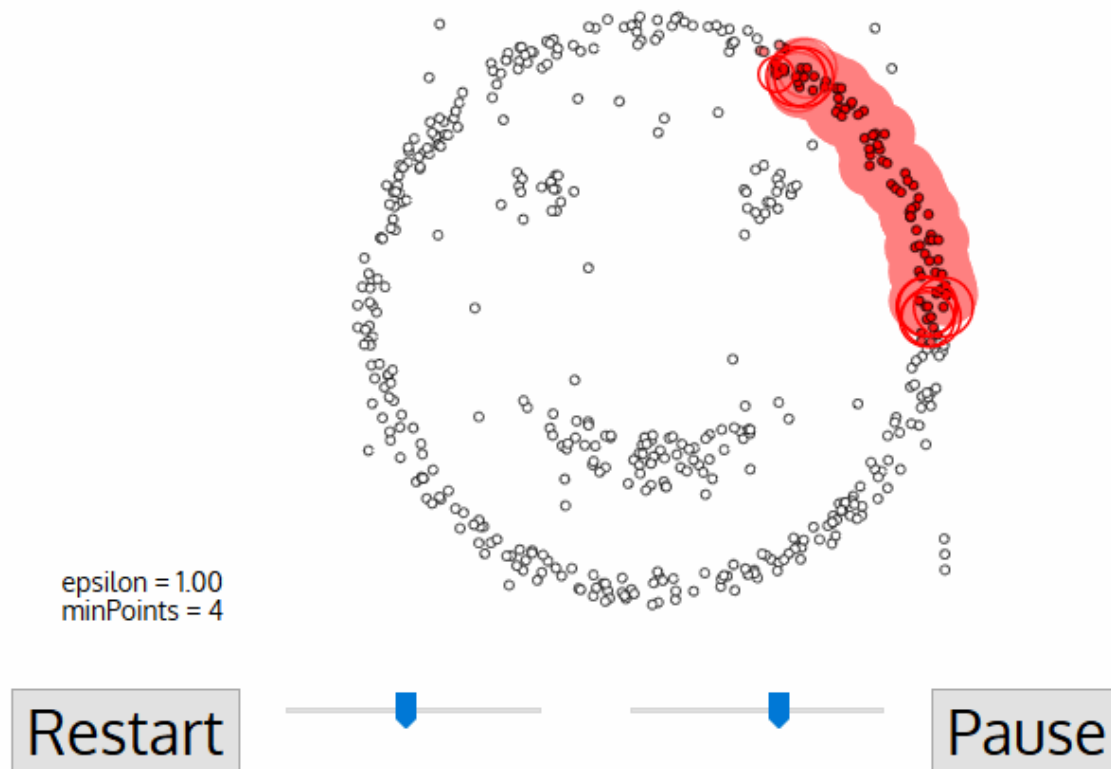
Here are the steps involved in the DBSCAN algorithm:

Step -1: First, define epsilon and identify MinPts.

Step -2: Starting with a core point, the algorithm finds all the points within its epsilon neighbourhood and assigns them to the same cluster. It then repeats this process for each core point and continues to add points to the same cluster as long as they are within epsilon distance of another point in the cluster.

Step -3: Identify border points: A border point is a point that has fewer than MinPts within its epsilon neighbourhood, but is reachable from a core point. These points are often considered part of a cluster but can also be considered noise.

Step-4: Output clusters: The algorithm outputs the clusters that have been identified.



The **advantages** of DBSCAN algorithm:

- Density-based clustering algorithms can handle noisy data since they can identify noise points as outliers that do not belong to any cluster.
- Unlike some other clustering algorithms, density-based clustering can identify clusters of different shapes and sizes, including irregularly shaped clusters.
- They don't require prior knowledge of the number of clusters, making them more flexible and versatile.
- They can efficiently process large datasets and handle high-dimensional data.

The **disadvantages** of DBSCAN algorithm:

- The performance of density-based clustering algorithms is highly dependent on the choice of parameters, such as ϵ and MinPts, which can be challenging to tune.
- They can be computationally expensive and time-consuming, especially for large datasets with complex structures.
- These algorithms may not be suitable for datasets with low-density regions or evenly distributed data points.

Example: Apply the DBSCAN algorithm to given data points and create the clusters with

MinPts = 3 and epsilon = 1.9.

Data points:

P1: (3,7)
P2: (4,6)
P3: (5,5)
P4: (6,3)
P5: (7,3)
P6: (6,2)
P7: (3,3)

Solution:

The given data

MinPts = 3

epsilon radius = 1.9

In this we use *Euclidian distance* to calculate the distance between the points.

$$\text{Euclidian distance } (A (x1, y1), B (x2, y2)) = \sqrt{((x1 - x2)^2 + (y1 - y2)^2)}$$

minPts = 3 and epsilon = 1.9

	P1	P2	P3	P4	P5	P6	P7
P1: (3,7)	0						
P2: (4,6)	1.41	0					
P3: (5,5)	2.83	1.41	0				
P4: (6,3)	5.00	2.83	3.60	0			
P5: (7,3)	5.66	4.24	2.83	1.41	0		
P6: (6,2)	5.83	4.47	3.16	2.00	1.00	0	
P7: (3,3)	4.00	3.16	2.83	3.16	4.00	3.16	0

Next, find the *nearest data points* for each data point are:

P1: P2
P2: P1, P3
P3: P2
P4:P5
P5:P4, P6
P6:P5
P7

Once the calculate nearest data points, we need to find the *Core*, *Border* and *Noise* data points.

minPts = 3 and epsilon = 1.9

Point	Status	
P1: P2	Noise	Border
P2: P1, P3	Core	
P3: P2	Noise	Border
P4:P5	Noise	Border
P5:P4, P6	Core	
P6:P5	Noise	Border
P7	Noise	

From the above,

- **P2** and **P5** are *Core data points*
- **P1, P3, P4** and **P6** are *Border data points*
- **P7** is *Noise* (or) *Outlier* data point

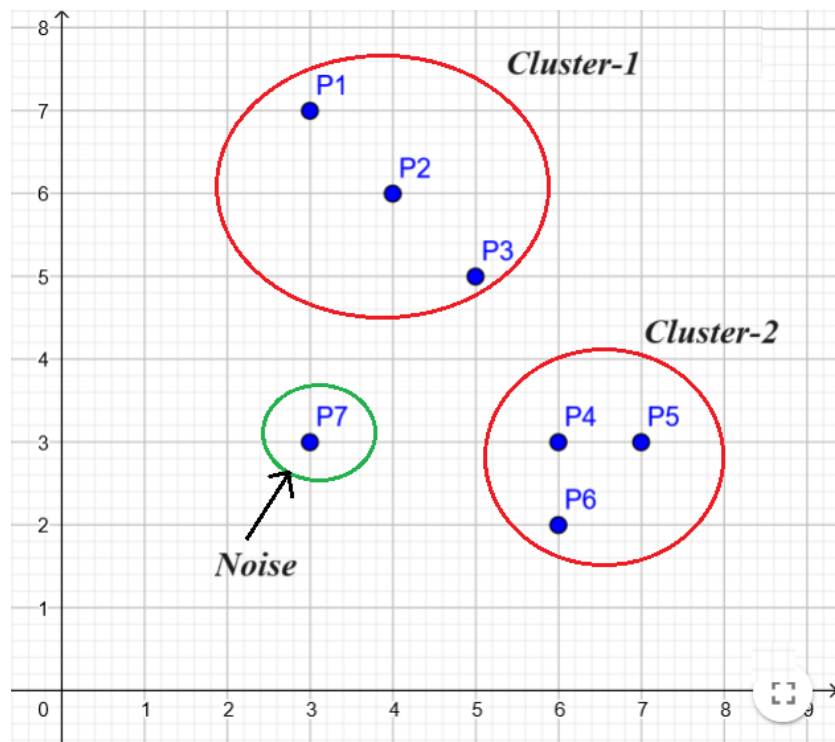
In the above there are *two core* data points that means there are *two clusters*.

Those are

Cluster-1: (P1, P2, P3)

Cluster-2: (P4, P5, P6)

Now, try to draw clusters on 2D space.



Outlier Analysis:

Outlier analysis is the process of identifying data points that are different from the majority of the data. Such data points are called outliers or anomalies because they do not follow the general pattern or trend found in the dataset.

Example: If most students in a class score between 60 and 90 marks, but one student scores 10, that score is an outlier.

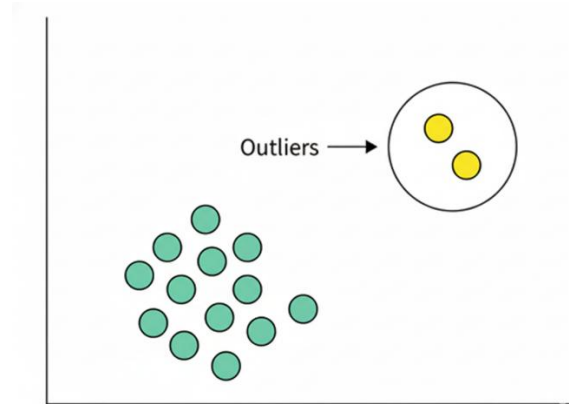


Fig: Outlier Analysis

Types of Outliers:

Outliers can be categorized into different types according to their attributes:

Global Outliers:

The individual data points that are significantly deviates from the rest of the dataset.

Example: For instance, if we have a dataset of ages for a group of people, and one person is 37 years old the rest of the ages range between 20 and 22 years, the age of 37 years would be a global outlier.

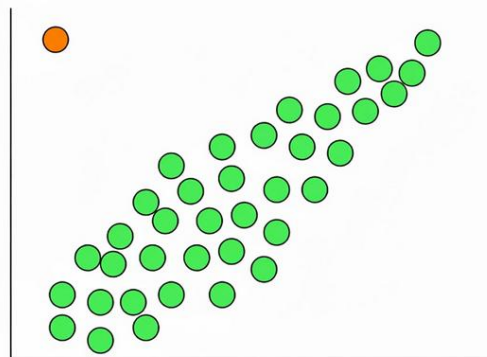


Fig: Global Outliers

Contextual Outliers:

These are data points that are considered outliers in a specific context.

Example: For instance, a high temperature may be normal in summer but an outlier in winter.

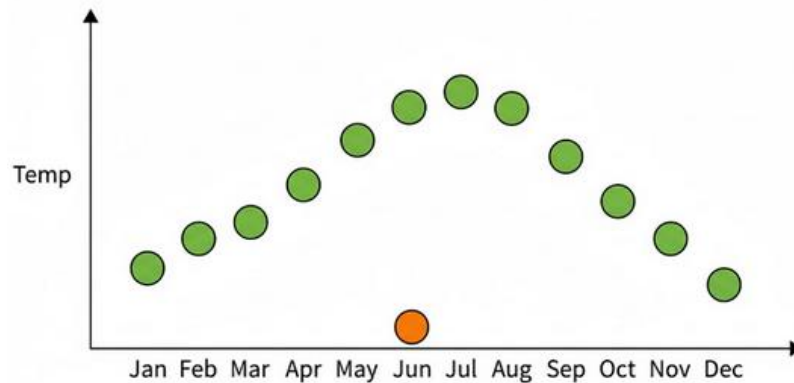


Fig: Contextual Outliers

Collective Outliers:

A group of data points that together behave abnormally, though individually they may seem normal.

Example: For instance, suppose we have a dataset of customer transactions, and a group of customers consistently makes purchases that are significantly larger than the rest of the customers. In that case, this group of customers could be considered a collective outlier.

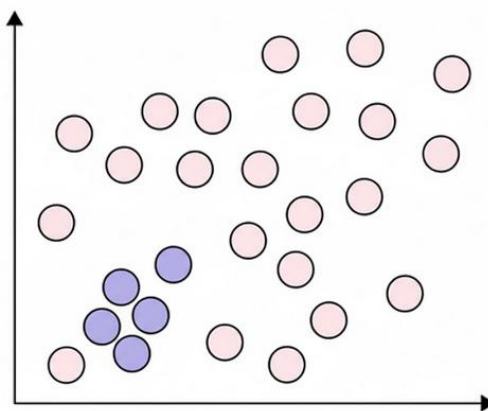


Fig: Collective Outliers

Common Methods for Outlier Detection:

There are several methods are used for outlier detection, falling into different categories:

1. Statistical Methods

These methods use the standard deviation or probability distribution to find extreme values.

Example Methods are,

- Standard Deviation Method
- IQR Method
- Z-Score Method

2. Distance-Based Methods:

These methods use distance measures to find how far a point is from its neighbours.

Example Method is,

- K-Nearest Neighbour

3. Clustering-Based Methods:

These methods use clustering algorithms to treat small or isolated clusters as outliers.

Example Method is,

- K-Means

4. Density-Based Methods:

These methods compare the density of data around a point to that of its neighbours.

Example Method is,

- DBSCAN

Applications of Outlier Analysis

Outlier analysis has many applications in various fields, as mentioned below -

- **Finance** - In finance, outlier analysis is used to identify abnormal fluctuations in stock prices or financial transactions, which can indicate fraud or insider trading.
- **Healthcare** - Outlier analysis is used in healthcare to identify patients with rare or unusual medical conditions or to detect abnormal patterns in medical data that can help diagnose diseases.
- **Manufacturing** - In manufacturing, outlier analysis is used to identify defective products or equipment producing out-of-specification results, which can affect the quality of the final product.
- **Marketing** - Outlier analysis is used to identify customers with high or low purchasing habits, which can help businesses create targeted marketing campaigns and promotions.
- **Environmental science** - Outlier analysis is used in environmental science to identify extreme weather events or natural disasters, which can help predict and mitigate the impact of these events on human populations and ecosystems.
- **Cybersecurity** - Outlier analysis is used in cybersecurity to detect abnormal network behavior or suspicious activity, indicating cyberattacks or data breaches.