Clustering Techniques:

Cluster analysis is a technique used in data mining and machine learning to group similar objects into clusters. Let's try understanding this with a simple example.

A bank wants to give credit card offers to its customers. Currently, they look at the details of each customer and, based on this information, decide which offer should be given to which customer.

Now, the bank can potentially have millions of customers. Does it make sense to look at the details of each customer separately and then make a decision? Certainly not! It is a manual process and will take a huge amount of time.

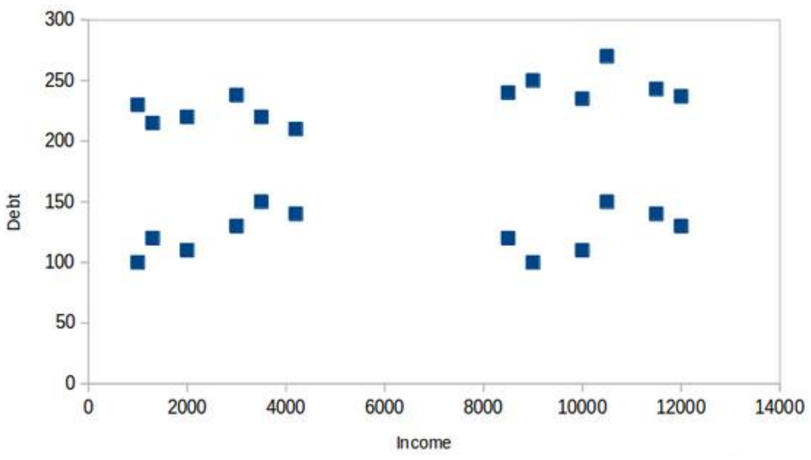
So, what can the bank do? One option is to segment its customers into different groups. For instance, the bank can group the customers based on their income:

High Income
Average Income
Low Income

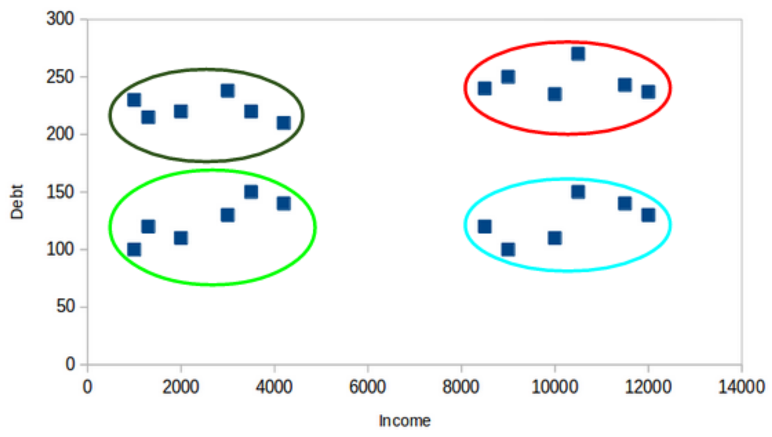
The bank can now make three different strategies or offers, one for each group. Here, instead of creating different strategies for individual customers, they only have to make 3 strategies. This will reduce the effort as well as the time.

The groups I have shown above are known as clusters, and the process of creating these groups is known as clustering. Clustering is the process of dividing the entire data into groups (also known as clusters) based on the patterns in the data.

We'll take the same bank as before, which wants to segment its customers. For simplicity purposes, let's say the bank only wants to use the income and debt to make the segmentation. They collected the customer data and used a scatter plot to visualize it:



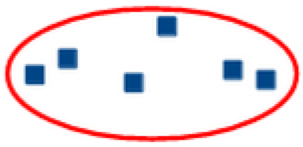
On the X-axis, we have the income of the customer, and the y-axis represents the amount of debt. Here, we can clearly visualize that these customers can be segmented into 4 different clusters, as shown below:



This is how clustering helps to create segments (clusters) from the data. The bank can further use these clusters to make strategies and offer discounts to its customers. So, let's look at the properties of these clusters.

First Property of Clustering:

All the data points in a cluster should be similar to each other. **E.g.:**

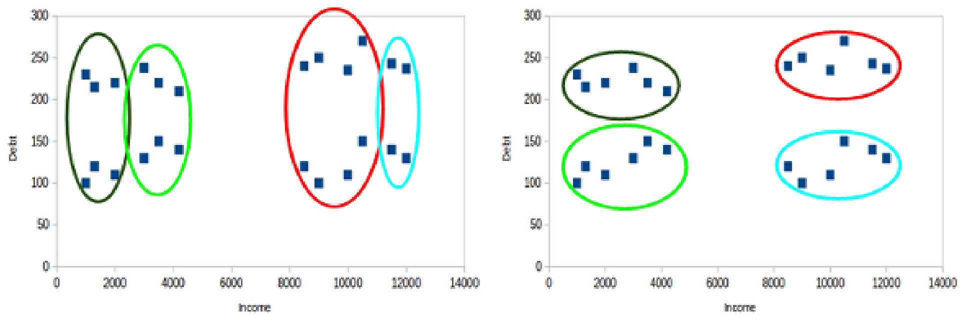


If the customers in a particular cluster are not similar to each other, then their requirements might vary, right? If the bank gives them the same offer, they might not like it, and their interest in the bank might reduce. Not ideal.

Having similar data points within the same cluster helps the bank to use targeted marketing. You can think of similar examples from your everyday life and consider how clustering will (or already does) impact the business strategy.

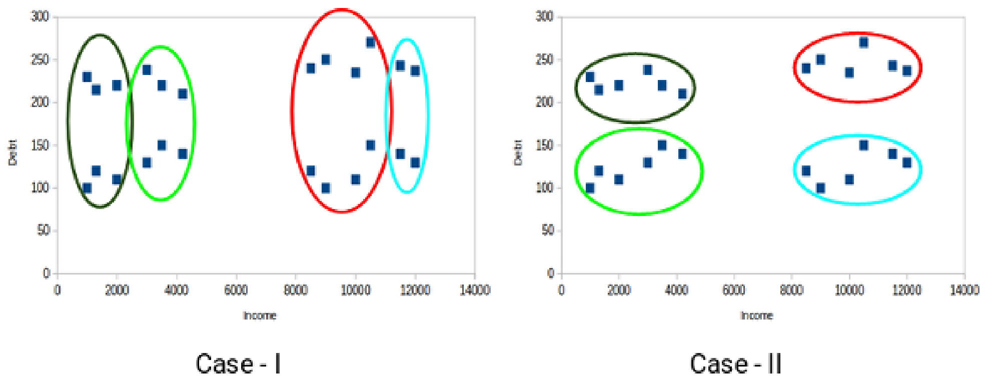
Second Property of Clustering:

The data points from different clusters should be as different as possible. This will intuitively make sense if you've grasped the above property. Let's again take the same example to understand this property:



Case - I Case - II

Understanding Different Evaluation Metrics for Clustering:
The primary aim of clustering is not just to make clusters but to make good and meaningful ones. We saw this in the below example:



Here, we used only two features, and hence it was easy for us to visualize and decide which of these clusters was better.

Unfortunately, that's not how real-world scenarios work. We will have a ton of features to work with. Let's take the customer segmentation example again – we will have features like customers' income, occupation, gender, age, and many more. We would not be able to visualize all these features together and decide on better and more meaningful clusters.

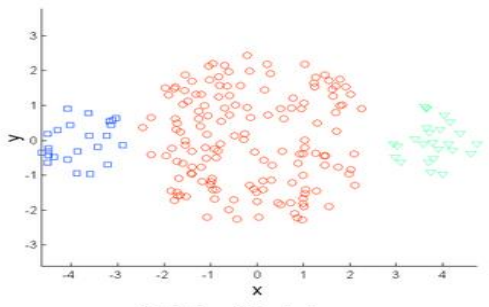
This is where we can make use of evaluation metrics. Let's discuss a few of them and understand how we can use them to evaluate the quality of our clusters.

K-Means Clustering: (note copy).

Challenges with the K-Means Clustering Algorithm:

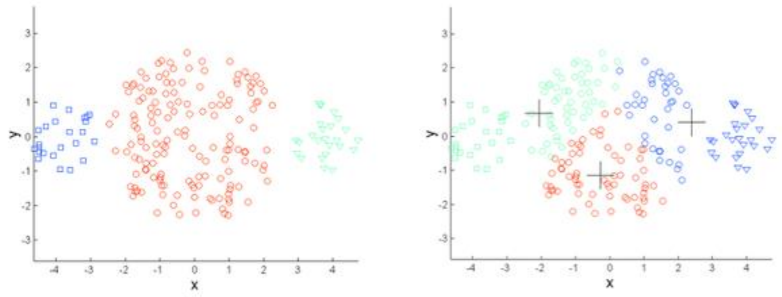
- The k value in k-means clustering is a crucial parameter that determines the number of clusters to be formed in the dataset. Finding the optimal k value in the k-means clustering can be very challenging, especially for noisy data. (The appropriate value of k depends on the data structure and the problem being solved. It is important to choose the right value of k, as a small value can result in under-clustered data, and a large value can cause over-clustering.)

- Also, one of the common challenges we face while working with K-Means is that the size of clusters is different**.** Let's say we have the following points:

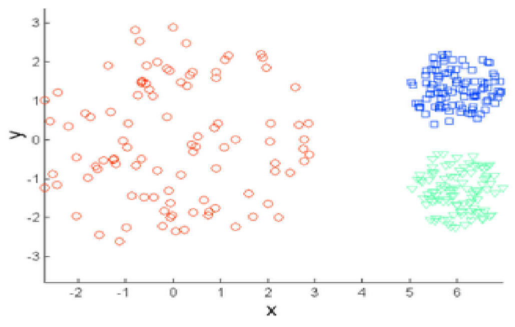


Original Points

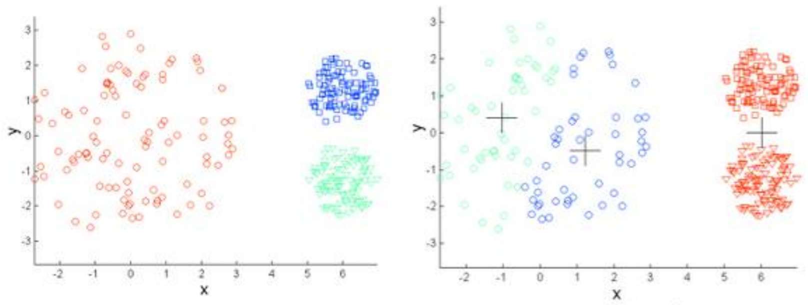
The leftmost and the rightmost clusters are of smaller size compared to the central cluster. Now, if we apply k-means clustering on these points, the results will be something like this:


Original Points K-means (k = 3)

- Another challenge with k-means is when the densities of the original points are different. Let's say these are the original points:


Original Points

Here, the points in the red cluster are spread out, whereas the points in the remaining clusters are closely packed together. Now, if we apply k-means on these points, we will get clusters like this:


Original Points K-means (k = 3)

- Determining the optimal number of clusters for k-means clustering can be another challenge.
- Outliers can have a significant impact on the results of k-means clustering, as the algorithm is sensitive to extreme values.
- However, as the size of the data set increases, the computational cost of k-means clustering can also increase**.**

Advantages:

- Easy to understand and implement.
- Efficient in terms of computational cost and works well on large datasets.
- Can be used for a wide range of data types, including continuous and categorical data.
- Generally, produces clusters that are well-separated, spherical and evenly-sized.

Disadvantages:

- Requires the number of clusters to be specified in advance, which can be difficult to determine in some cases.
- May converge to local minima, resulting in suboptimal clustering results.
- Sensitive to initial conditions, which can result in different clustering results depending on the initial placement of centroids.
- Assumes that clusters are spherical and evenly-sized, which may not be true in all datasets.

It is important to note that many of the disadvantages of K-Means clustering can be addressed by using variations or extensions of the algorithm, such as hierarchical clustering or fuzzy clustering.

K-Means ++ Clustering:

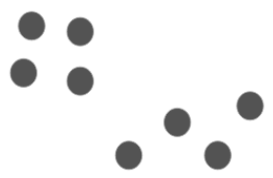
Remember how we randomly initialize the centroids in k-means clustering? Well, this is also potentially problematic because we might get different clusters every time. So, to solve this problem of random initialization, there is an algorithm called **K-Means++** that can be used to choose the initial values, or the initial cluster centroids, for K-Means.

It specifies a procedure to initialize the cluster centres before moving forward with the standard k-means clustering algorithm.

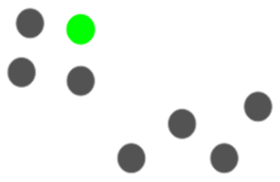
The steps to initialize the centroids using K-Means++ are:

1. The first cluster is chosen uniformly at random from the data points we want to cluster. This is similar to what we do in K-Means, but instead of randomly picking all the centroids, we just pick one centroid here
2. Next, we compute the distance (D(x)) of each data point (x) from the cluster centre that has already been chosen
3. Then, choose the new cluster centre from the data points with the probability of x being proportional to (D(x))2
4. We then repeat steps 2 and 3 until *k* clusters have been chosen.

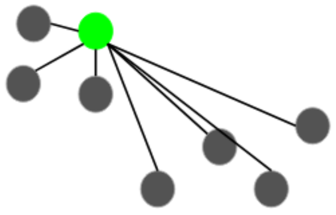
- Let's take an example to understand this more clearly. Let's say we have the following points, and we want to make 3 clusters here:



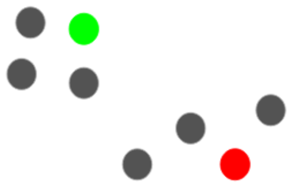
- Now, the **first step** is to randomly pick a data point as a cluster centroid:



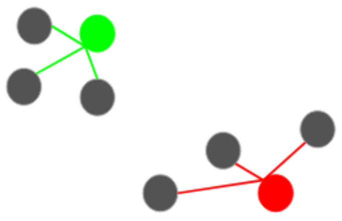
- Let’s say we pick the green point as the initial centroid. Now, we will calculate the distance (D(x)) of each data point with this centroid:



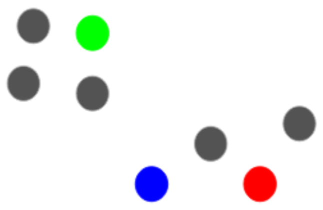
- The next centroid will be the one whose squared distance (D(x)2) is the farthest from the current centroid:



- In this case, the red point will be selected as the next centroid. Now, to select the last centroid, we will take the distance of each point from its closest centroid, and the point having the largest squared distance will be selected as the next centroid:



- We will select the last centroid as:



- We can continue with the K-Means algorithm after initializing the centroids. Using K-Means++ to initialize the centroids tends to improve the clusters. Although it is computationally costly relative to random initialization, subsequent K-Means often converge more rapidly.

Advantages:

K-Means++ is an improvement over the traditional K-Means algorithm that aims to address some of its limitations. Some advantages of K-Means++ algorithm are:

- It produces better initial centroids: K-Means++ algorithm uses a smarter initialization process to select the initial centroids, which are more spread out across the dataset and less likely to result in suboptimal clustering.

- It is less sensitive to initial conditions: The improved initialization process makes K-Means++ less sensitive to the initial placement of centroids, which means that it is more likely to converge to a globally optimal clustering solution.

- It is computationally efficient: K-Means++ algorithm has similar computational complexity as the original K-Means algorithm, which makes it efficient for large datasets.

- It can improve clustering quality: The improved initialization process can lead to better clustering results, particularly in cases where the number of clusters is not known in advance.

Mini Batch K-Means Algorithms:

Optimal value of K in K-Means Clustering K-means is one of the most popular clustering algorithms, mainly because of its good time performance. With the increasing size of the datasets being analysed, the computation time of K-means increases because of its constraint of needing the whole dataset in main memory. For this reason, several methods have been proposed to reduce the temporal and spatial cost of the algorithm.

The Mini-batch K-means clustering algorithm is a variant of the traditional K-means algorithm. It stores data in memory in short, random, fixed-size batches, and then a random sample of the data is gathered and utilized to update the clusters with each iteration.

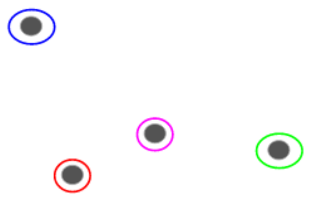
The algorithm iterates between two major steps, similar to vanilla k-means. In the **first step**, samples are drawn randomly from the dataset, to form a mini-batch. These are then assigned to the nearest centroid. In the **second step**, the centroids are updated. In contrast to k-means, this is done on a per-sample basis. For each sample in the mini-batch, the assigned centroid is updated by taking the streaming average of the sample and all previous samples assigned to that centroid. This has the effect of decreasing the rate of change for a centroid over time. These steps are performed until convergence or a predetermined number of iterations is reached.

Mini Batch K-Means converges faster than **K-Means**, but the quality of the results is reduced.

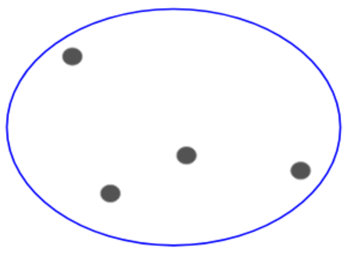
Hierarchical Clustering:

Let's say we have the below points and we want to cluster them into groups:

We can assign each of these points to a separate cluster:



Now, based on the similarity of these clusters, we can combine the most similar clusters together and repeat this process until only a single cluster is left:



We are essentially building a hierarchy of clusters. That's why this algorithm is called hierarchical clustering.

Types of Hierarchical Clustering:

- **Agglomerative Hierarchical Clustering:** Consider each data point is a cluster and then based on similarity create a Big Cluster.
- **Divisive Hierarchical Clustering:** Consider there is a big cluster and then divide the big cluster into small individual cluster.

We merge the most similar points or clusters in hierarchical clustering – we know this. Now the question is – how do we decide which points are similar and which are not? It's one of the most important questions in clustering!

Here's one way to calculate similarity – Take the distance between the centroids of these clusters. The points having the least distance are referred to as similar points and we can merge them. We can refer to this as a **distance-based algorithm** as well (since we are calculating the distances between the clusters).

In hierarchical clustering, we have a concept called a **proximity matrix**. This stores the distances between each point.

Understand through an Example:

Let’s take a sample of 5 students:

Student_ID	Marks
1	10
2	7
3	28
4	20
5	35

- **Creating a Proximity Matrix**

  First, we will create a proximity matrix which will tell us the distance between each of these points. Since we are calculating the distance of each point from each of the other points, we will get a square matrix of shape n X n (where n is the number of observations).

  Let’s make the 5 x 5 proximity matrix for our example:

ID	**1**	**2**	**3**	**4**	**5**
1	0	3	18	10	25
2	3	0	21	13	28
3	18	21	0	8	7
4	10	13	8	0	15
5	25	28	7	15	0

  The diagonal elements of this matrix will always be 0 as the distance of a point with itself is always 0. We will use the Euclidean distance formula to calculate the rest of the distances. So, let’s say we want to calculate the distance between point 1 and 2:

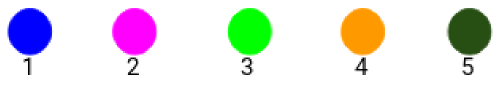
  $\sqrt{(10-7)^2} = \sqrt{9} = 3$

  Similarly, we can calculate all the distances and fill the proximity matrix.
  (basically, this points are belong in x and y co-ordinate, then we can calculate Euclidean distance.

- **Steps to Perform Hierarchical Clustering:**

Step 1:

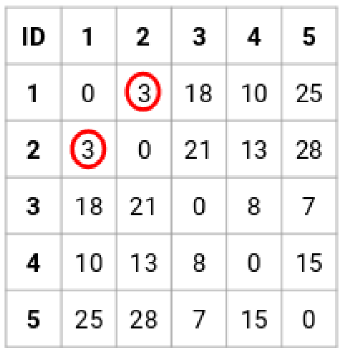
First, we assign all the points to an individual cluster:



Different colors here represent different clusters. You can see that we have 5 different clusters for the 5 points in our data.

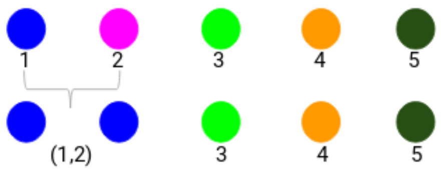
Step 2:

Next, we will look at the smallest distance in the proximity matrix and merge the points with the smallest distance. We then update the proximity matrix:

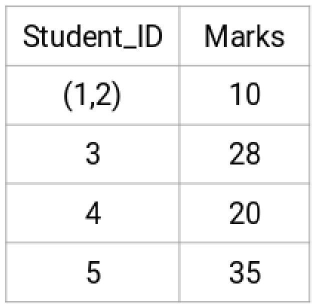


ID	1	2	3	4	5
1	0	3	18	10	25
2	3	0	21	13	28
3	18	21	0	8	7
4	10	13	8	0	15
5	25	28	7	15	0

Here, the smallest distance is 3 and hence we will merge point 1 and 2:



Let’s look at the updated clusters and accordingly update the proximity matrix:



Student_ID	Marks
(1,2)	10
3	28
4	20
5	35

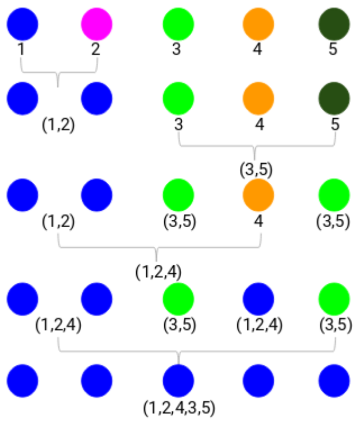
Here, we have taken the maximum of the two marks (7, 10) to replace the marks for this cluster. Instead of the maximum, we can also take the minimum value or the average values as well. Now, we will again calculate the proximity matrix for these clusters:

ID	**(1,2)**	**3**	**4**	**5**
(1,2)	0	18	10	25
3	18	0	8	7
4	10	8	0	15
5	25	7	15	0

Step-3:

We will repeat step 2 until only a single cluster is left.

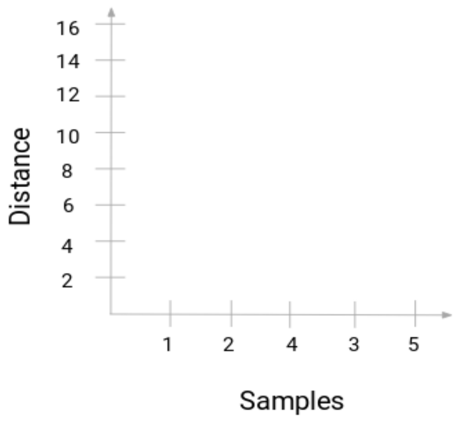
So, we will first look at the minimum distance in the proximity matrix and then merge the closest pair of clusters. We will get the merged clusters as shown below after repeating these steps:



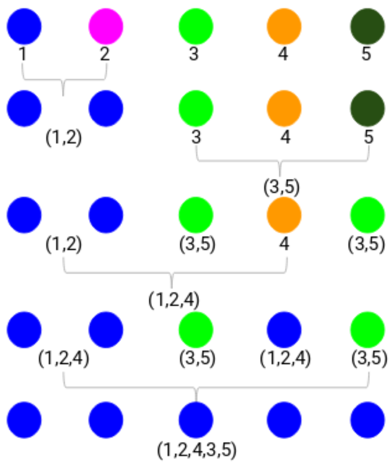
We started with 5 clusters and finally have a single cluster. **This is how agglomerative hierarchical clustering works**.

- **How to choose number of cluster in Hierarchical Clustering:**
  Ready to finally answer this question that's been hanging around since we started learning? To get the number of clusters for hierarchical clustering, we make use of an awesome concept called a **Dendrogram** (A dendrogram is a tree-like diagram that records the sequences of merges or splits).

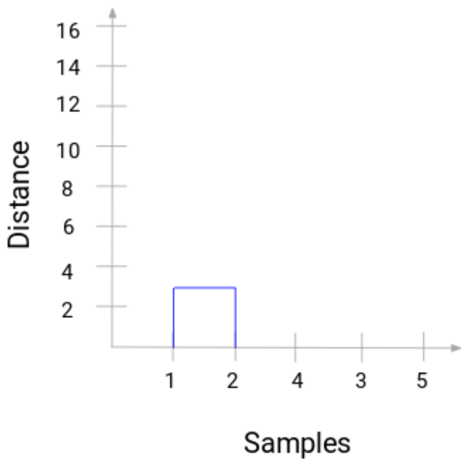
  Let's get back to our teacher-student example. Whenever we merge two clusters, a dendrogram will record the distance between these clusters and represent it in graph form. Let's see how a dendrogram looks like:



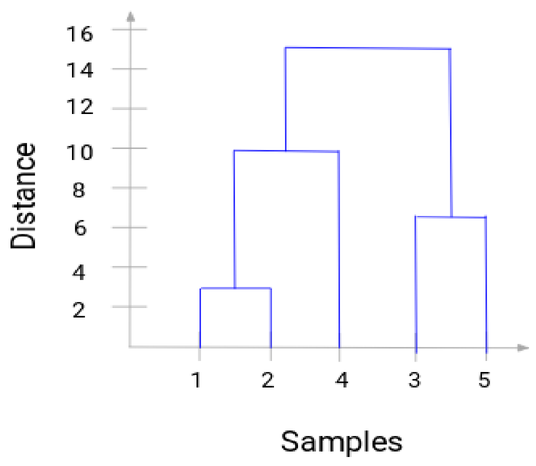
We have the samples of the dataset on the x-axis and the distance on the y-axis. **Whenever two clusters are merged, we will join them in this dendrogram and the height of the join will be the distance between these points.** Let's build the dendrogram for our example:



Take a moment to process the above image. We started by merging sample 1 and 2 and the distance between these two samples was 3 (refer to the first proximity matrix in the previous section). Let's plot this in the dendrogram:

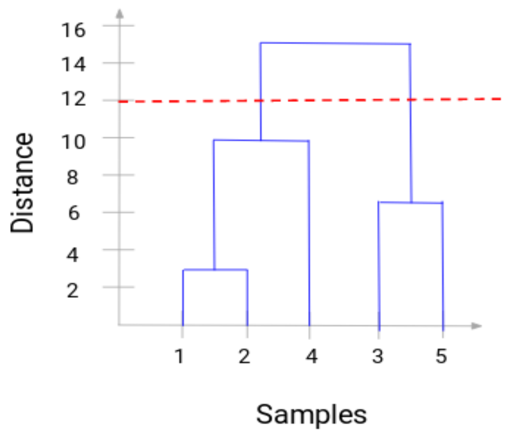


Here, we can see that we have merged sample 1 and 2. The vertical line represents the distance between these samples. Similarly, we plot all the steps where we merged the clusters and finally, we get a dendrogram like this:



We can clearly visualize the steps of hierarchical clustering. **More the distance of the vertical lines in the dendrogram, more the distance between those clusters.**

Now, we can set a threshold distance and draw a horizontal line (<u>Generally, we try to set the threshold in such a way that it cuts the tallest vertical line</u>). Let's set this threshold as 12 and draw a horizontal line:



The number of clusters will be the number of vertical lines which are being intersected by the line drawn using the threshold. In the above example, since the red line intersects 2 vertical lines, we will have 2 clusters. One cluster will have a sample (1,2,4) and the other will have a sample (3,5). Pretty straightforward, right?

This is how we can decide the number of clusters using a dendrogram in Hierarchical Clustering.

Since clusters are sets of points, there are many different kinds of linkage methods:

- **Single Linkage:** cluster distance = smallest pairwise distance
- **Complete Linkage:** cluster distance = largest pairwise distance

- **Average Linkage:** cluster distance = average pairwise distance
- **Centroid Linkage:** cluster distance= distance between the centroids of the clusters
- **Ward's Linkage:** cluster criteria= Minimize the variance in the cluster

Advantages:

- No prior knowledge of the number of clusters is required, as the number of clusters can be determined based on the dendrogram.
- The resulting dendrogram provides a visual representation of the hierarchical relationship between data points and clusters.
- Can work well on small and medium-sized datasets with a relatively small number of clusters.
- Can handle a variety of distance metrics and linkage methods to define clusters.

Disadvantages:

- Can be computationally intensive for large datasets, particularly with the agglomerative hierarchical clustering method.
- Results can be sensitive to the choice of distance metric and linkage method used.
- Lack of flexibility in the number of clusters that can be produced, as once the dendrogram is cut, the number of clusters is fixed.
- May not produce well-separated clusters when the dataset is noisy or when the data points do not have clear hierarchical relationships.

Overall, hierarchical clustering can be a useful technique for identifying relationships between data points and clusters in small and medium-sized datasets. However, it may not be the best choice for large datasets or when the data does not have clear hierarchical relationships.

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

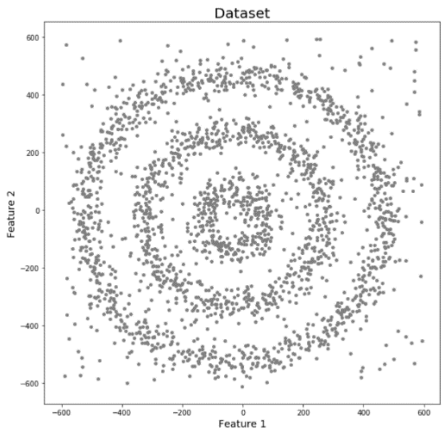
Clustering is an **unsupervised learning** technique where we try to group the data points based on specific characteristics. There are various clustering algorithms with **K-Means** and **Hierarchical** being the most used ones. Some of the use cases of clustering algorithms include:

- Document Clustering
- Recommendation Engine
- Image Segmentation
- Market Segmentation
- Search Result Grouping
- and Anomaly Detection.

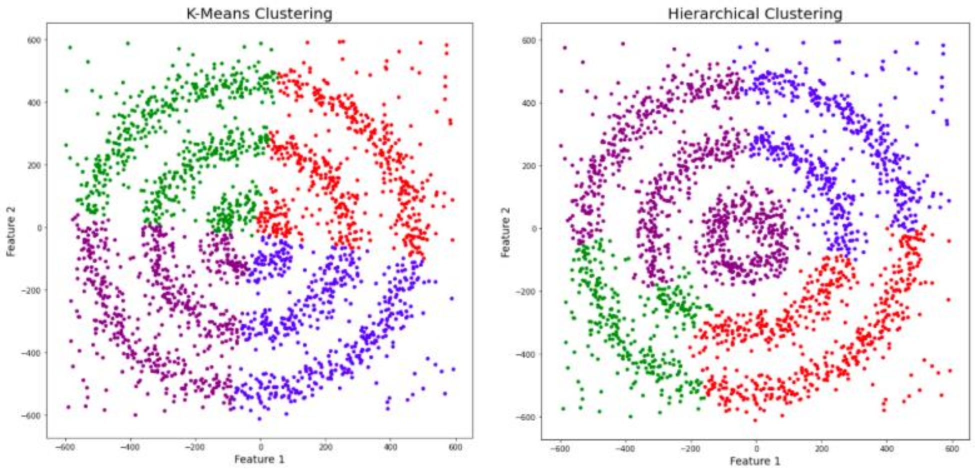
All these problems use the concept of clustering to reach their end goal. Therefore, it is crucial to understand the concept of clustering. But here's the issue with these two clustering algorithms:

K-Means and Hierarchical Clustering both fail in creating clusters of arbitrary shapes. They are not able to form clusters based on varying densities (data is too far from each other). That's why we need DBSCAN clustering.

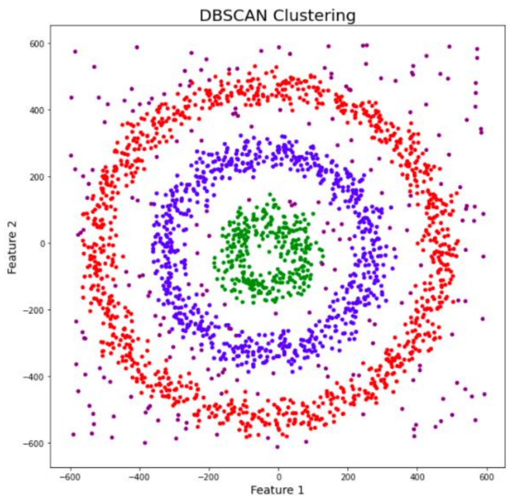
Let's try to understand it with an example. Here we have data points densely present in the form of concentric circles:



We can see three different dense clusters in the form of concentric circles with some noise here. Now, let's run K-Means and Hierarchical clustering algorithms and see how they cluster these data points.



You might be wondering why there are four colors in the graph? As I said earlier, this data contains noise too, therefore, I have taken noise as a different cluster which is represented by the purple color. Sadly, both of them failed to cluster the data points. Also, they were not able to properly detect the noise present in the dataset. Now, let's take a look at the results from DBSCAN clustering.



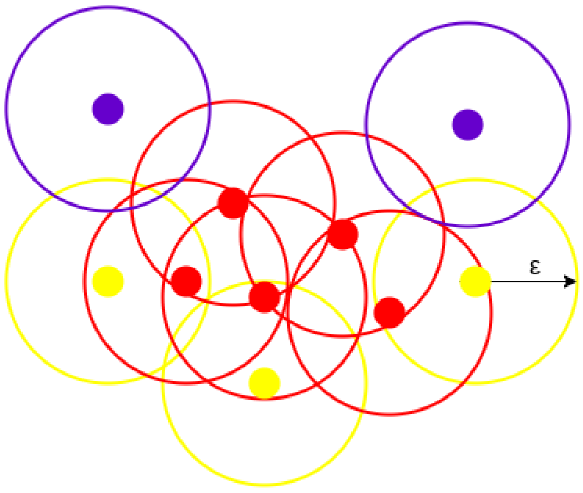
How Algorithm work:

- **Epsilon:** This is also called eps. This is the distance till which we look for the neighbouring points (nothing but radius).
- **Min_points:** The minimum number of points specified by the user which is located inside the eps or radius.
- **Core Points:** If the number of points inside the eps radius of a point is greater than or equal to the min_points then it's called a core point.
- **Border Points:** If the number of points inside the eps radius *of a point is less than the* min_points *and it lies within the* eps radius region of a core point, it's called a border point.
- **Noise:** A point which is neither a core nor a border point is a noise point.

Steps:

1. The algorithm starts with a random point in the dataset which has not been visited yet and its neighbouring points are identified based on the eps value.
2. If the point contains greater than or equal points than the min_pts, then the cluster formation starts and this point becomes a *core point*, else it's considered as noise. The thing to note here is that a point initially classified as noise can later become a border point if it's in the eps radius of a core point.
3. If the point is a core point, then all its neighbours become a part of the cluster. If the points in the neighbourhood turn out to be core points, then their neighbours are also part of the cluster.
4. Repeat the steps above until all points are classified into different clusters or noises.

This algorithm works well if all the clusters are dense enough, and they are well separated by low-density regions.



Advantages:

- DBSCAN does not require prior knowledge of the number of clusters to be specified in advance.
- It can handle datasets with arbitrary shapes and cluster sizes, making it suitable for a wide range of applications.
- It can identify noise points as outliers and exclude them from the clustering process.
- The clustering results are not sensitive to the order in which the data points are processed.

Disadvantages:

- The performance of DBSCAN can be sensitive to the choice of hyperparameters, such as the distance metric and the minimum number of points in a cluster.
- DBSCAN is not well-suited to datasets with varying densities, as it relies on a fixed radius to define neighbourhood sizes.
- The algorithm can produce suboptimal clustering results if the data contains clusters with significantly different densities.
- DBSCAN may not be computationally efficient for high-dimensional datasets, as the distance calculations become more expensive with increasing dimensionality.

Mean Shift Clustering Algorithm:

Mean Shift is a clustering algorithm that is used to group similar data points together in a dataset. It is a non-parametric algorithm, which means that it does not make any assumptions about the underlying distribution of the data.

The algorithm works by iteratively shifting the mean of the data points in the direction of the densest area of the dataset. Specifically, it computes the mean of all data points within a certain radius (called the bandwidth) of a given point, and then moves the point to the location of the mean. This process is repeated for all points in the dataset until convergence is reached.

The result of the algorithm is a set of clusters, where each cluster is represented by a centroid (i.e., the mean of the points in the cluster). The number of clusters is not predetermined and is determined by the algorithm based on the density of the data.

Mean Shift is a powerful algorithm for clustering data that does not fit well into traditional clustering methods such as k-means. It has been successfully used in a variety of applications, including computer vision, image segmentation, and object tracking, etc.

Step by Step process:

1. **Define the kernel or bandwidth size:** The first step is to define the size of the kernel or bandwidth (nothing but radius). This is a parameter that determines the size of the region around each data point that will be used to calculate its mean. The kernel can be defined using different methods, such as a Gaussian function or a uniform function.

2. **Initialize the centroids:** The second step is to initialize the centroids. This can be done by selecting random data points from the dataset or using some other initialization method.

3. **Calculate the mean shift vector:** For each data point, calculate the mean shift vector by taking the weighted average of the distance between the data point and the other data points within the kernel.

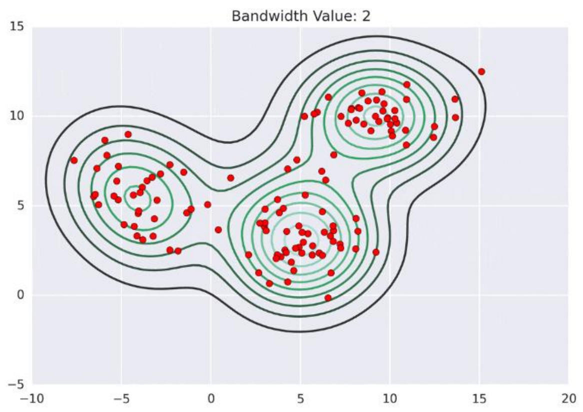
4. **Shift the centroids:** Move the centroids to the location of the mean shift vectors.

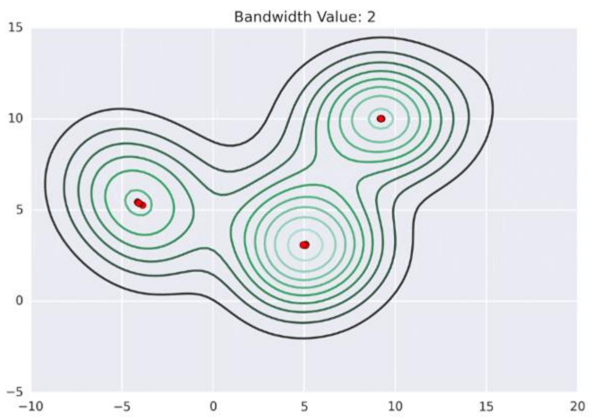
5. **Repeat steps 3 and 4 until convergence:** Repeat steps 3 and 4 until the centroids no longer move or until a predefined number of iterations is reached.

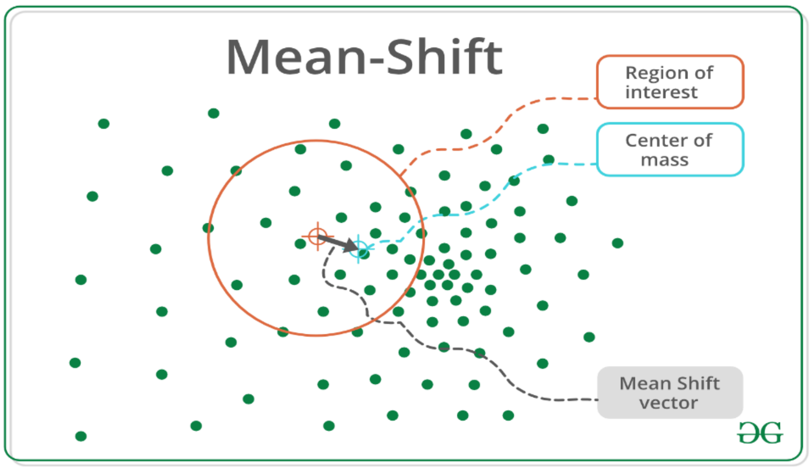
6. **Assign points to clusters:** Assign each data point to the nearest centroid.

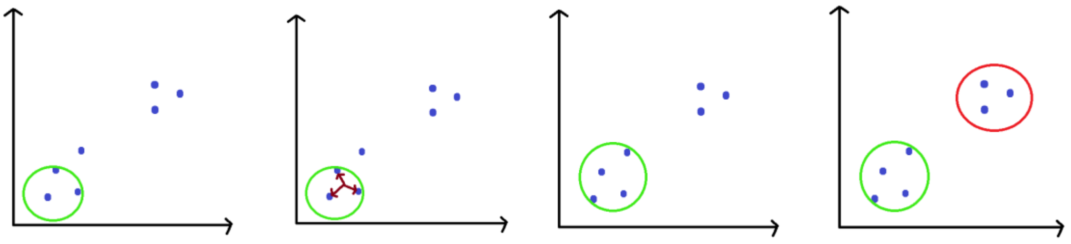
7. **Return the clusters:** Return the resulting clusters, where each cluster consists of the data points assigned to a particular centroid.

Overall, the mean shift algorithm works by iteratively shifting the centroids towards the densest areas of the data until they converge to a set of clusters. The algorithm is sensitive to the choice of kernel size and initialization method, so these parameters may need to be adjusted to achieve the best results.









Advantages:

- Mean Shift is a powerful algorithm for clustering data that does not fit well into traditional clustering methods such as k-means.
- It does not make any assumptions about the underlying distribution of the data, making it a non-parametric method.
- It can automatically determine the number of clusters based on the density of the data, which makes it easier to use than other clustering algorithms.
- It can handle non-linear data and does not require the data to be linearly separable.
- It can work well on high-dimensional datasets.

Disadvantages:

- The Mean Shift algorithm can be slow and computationally intensive, especially for large datasets or when the kernel size is large.
- The choice of kernel size is crucial and can significantly impact the results of the algorithm. Choosing an inappropriate kernel size can result in poor clustering performance.
- The algorithm is sensitive to the initialization of the centroids, and the quality of the results can depend on the initialization method.
- It can be difficult to interpret the resulting clusters since the algorithm does not provide clear boundaries between the clusters.
- Mean Shift is not suitable for datasets with widely varying densities, as it can result in uneven clustering.

OPTICS Clustering (Ordering Points To Identify Cluster Structure):

OPTICS (Ordering Points To Identify the Clustering Structure) is a density-based clustering algorithm, similar to DBSCAN (Density-Based Spatial Clustering of Applications with Noise), but it can extract clusters of varying densities and shapes. It is useful for identifying clusters of different densities in large, high-dimensional datasets.

The main idea behind OPTICS is to extract the clustering structure of a dataset by identifying the density-connected points. The algorithm builds a density-based representation of the data by creating an ordered list of points called the reachability plot. Each point in the list is associated with a reachability distance, which is a measure of how easy it is to reach that point from other points in the dataset. Points with similar reachability distances are likely to be in the same cluster.

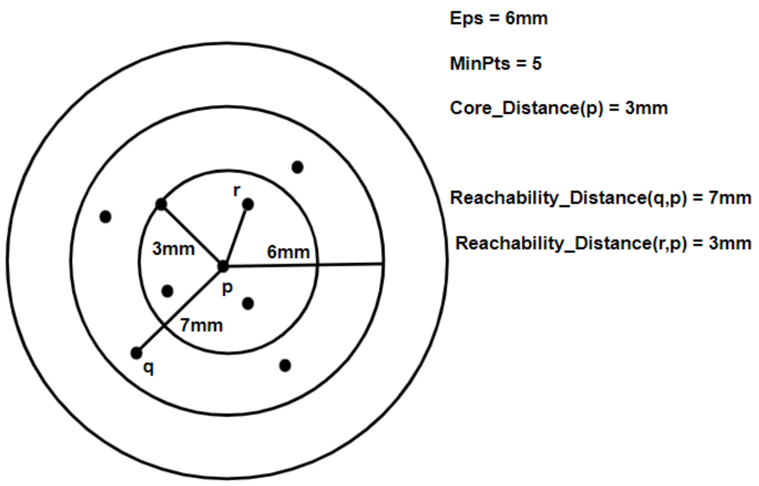
Step by Step process to create cluster:

- Define a density threshold parameter, Eps, which controls the minimum density of clusters.
- For each point in the dataset, calculate the distance to its k-nearest neighbours.
- Starting with an arbitrary point, calculate the reachability distance of each point in the dataset, based on the density of its neighbours.
- Order the points based on their reachability distance and create the reachability plot.
- Extract clusters from the reachability plot by grouping points that are close to each other and have similar reachability distances.

They are: -

1. **Core Distance:** It is the minimum value of radius required to classify a given point as a core point. If the given point is not a Core point, then it’s Core Distance is undefined.

2. **Reachability Distance:** It is defined with respect to another data point q(Let). The Reachability distance between a point p and q is the maximum of the Core Distance of p and the Euclidean Distance (or some other distance metric) between p and q. Note that The Reachability Distance is not defined if q is not a Core point.


Eps = 6mm
MinPts = 5
Core_Distance(p) = 3mm
Reachability_Distance(q,p) = 7mm
Reachability_Distance(r,p) = 3mm
r
3mm
6mm
p
7mm
q

This clustering technique is different from other clustering techniques in the sense that this technique does not explicitly segment the data into clusters. Instead, it produces a visualization of Reachability distances and uses this visualization to cluster the data.

Difference between OPTICS & DBSCAN:

Both OPTICS and DBSCAN (Density-Based Spatial Clustering of Applications with Noise) are density-based clustering algorithms that can discover clusters of arbitrary shapes and sizes. However, there are some differences between the two algorithms:

- **Cluster Extraction:** In DBSCAN, clusters are extracted based on a fixed density threshold (minimum number of points within a given radius) around each core point, while in OPTICS, the cluster extraction is based on the reachability distance and steepness of the points in the reachability plot. OPTICS can extract clusters of varying density, while DBSCAN requires a fixed density threshold.

- **Scalability:** OPTICS can handle larger datasets than DBSCAN because it does not require the construction of a neighbourhood graph, which can be memory-intensive and computationally expensive for large datasets. OPTICS only requires the computation of pairwise distances between data points.

- **Parameters:** DBSCAN requires two user-defined parameters: the radius epsilon and the minimum number of points required to form a dense region. OPTICS also requires two user-defined parameters: the number of nearest neighbours k and the cluster **extraction threshold**, which determines the minimum steepness required for a point to be considered as a cluster.

- **Noise Handling:** Both algorithms can handle noise points, but OPTICS can distinguish between noise points and points that belong to small clusters, while DBSCAN considers all points that do not belong to any cluster as noise points.

Advantages:

1. **Flexibility:** OPTICS can detect clusters of varying densities and shapes, making it suitable for datasets with complex structures.
2. **Hierarchy:** OPTICS produces a hierarchical clustering result, which can be useful for exploratory data analysis and identifying nested clusters.
3. **Noise handling:** OPTICS can distinguish between noise points and points that belong to small clusters, allowing for more accurate identification of clusters.
4. **Parameter tuning:** OPTICS requires fewer parameters to be tuned than other density-based clustering algorithms like DBSCAN, which can be advantageous for datasets with varying density levels.
5. **Scalability:** OPTICS is more scalable than DBSCAN and other density-based clustering algorithms as it does not require a neighbourhood graph to be constructed.

Disadvantages:

1. **Complexity:** The OPTICS algorithm has a higher computational complexity than other clustering algorithms like K-means, making it less suitable for large datasets.
2. **Parameter tuning:** Although OPTICS requires fewer parameters than other density-based clustering algorithms, tuning the parameters can still be challenging for users who are not familiar with the algorithm.
3. **Interpretation:** The hierarchical structure of the clusters produced by OPTICS can be difficult to interpret, and the resulting dendrogram may not be intuitive for some users.
4. **Implementation:** OPTICS is not implemented in some popular data analysis libraries, such as scikit-learn, making it less accessible to users who are not familiar with the algorithm.

Overall, OPTICS is a powerful clustering algorithm that can handle complex datasets and provide a hierarchical clustering result. However, its complexity and parameter tuning requirements can make it less accessible to users who are not familiar with the algorithm.

BIRCH Clustering (Balanced Iterative Reducing and Clustering using Hierarchies):

Clustering algorithms like K-means clustering do not perform clustering very efficiently and it is difficult to process large datasets with a limited amount of resources (like memory or a slower CPU).

So, regular clustering algorithms do not scale well in terms of running time and quality as the size of the dataset increases. This is where BIRCH clustering comes in.

Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH) is a clustering algorithm that can cluster large datasets by first generating a small and compact summary of the large dataset that retains as much information as possible. This smaller summary is then clustered instead of clustering the larger dataset. BIRCH is often used to complement other clustering algorithms by creating a summary of the dataset that the other clustering algorithm can now use. However, BIRCH has one major drawback – it can only process metric attributes.

The BIRCH algorithm works by recursively partitioning the data into a tree-like hierarchical structure. It does this by maintaining a set of cluster feature summaries (CFs) that are used to represent the data. The CFs are updated iteratively as new data points are added to the dataset.

Step by Step process to create BIRCH:

- Initialize the algorithm by specifying the number of clusters desired and the maximum amount of memory that can be used to store the data.

- Read in a small number of data points from the input dataset and initialize the cluster feature summaries (CFs). This is done by calculating the initial CFs for each data point and adding them to the set of CFs.

- Read in the next block of data points from the input dataset and update the CFs iteratively. This is done by calculating the CFs for each data point and adding them to the set of CFs. The CFs are accumulated by updating the statistics of each CF, including the sum of the feature values and the number of data points represented by the CF.

- Condense the CFs by merging them iteratively to form larger CFs. This is done by comparing the distance between the CFs to a threshold value. If the distance is less than the threshold, the CFs are merged into a larger CF.

- Repeat steps 3 and 4 until all of the data points have been processed.

- Finalize the CFs by merging the remaining CFs into larger CFs. This is done by comparing the distance between the CFs to the threshold value and merging the CFs that are close together.

- Construct a tree-like hierarchical structure that represents the clusters in the data. This is done by clustering the CFs using a clustering algorithm such as agglomerative hierarchical clustering.

- Traverse the tree-like structure to obtain the final set of clusters. This is done by selecting the clusters that meet the desired criteria, such as having a minimum number of data points or a minimum distance between clusters.

- Output the final set of clusters.

The BIRCH algorithm is designed to handle large datasets efficiently by using CFs to represent the statistical properties of the data points in the clusters. The algorithm can be used for a variety of clustering tasks and is particularly useful for datasets with a large number of features.

Step by Step process to create CFs (Cluster Features Summaries):

- **Initialize the CFs:** At the beginning of the clustering process, the algorithm initializes the CFs. For each data point, a CF is created that includes the feature values for that data point, as well as the count of data points represented by the CF.

- **Accumulate the CFs:** As the algorithm processes new data points, it accumulates the CFs. For each new data point, the algorithm identifies the nearest CF and updates the statistics of that CF. This includes incrementing the count of data points represented by the CF and updating the sum of feature values for each feature in the CF.

- **Condense the CFs:** Once a certain number of data points have been accumulated in the CFs, the algorithm begins to condense them. This is done by merging the CFs that are close together. The distance between the CFs is determined by a threshold value, which is set by the user.

- **Finalize the CFs:** Once all of the data points have been processed, the algorithm finalizes the CFs by merging the remaining CFs. This is done by comparing the distance between the CFs to the threshold value and merging the CFs that are close together.

- **Use the CFs for clustering:** The CFs are used to represent the statistical properties of the data points in the clusters. They can be used to identify the number of clusters, the size of each cluster, and the centroid of each cluster. The CFs can also be used to perform hierarchical clustering, in which the CFs are grouped into larger clusters based on their proximity to one another.

The process of creating CFs is iterative, and the CFs are updated as new data points are added to the dataset. The CFs are designed to be memory-efficient and can be used to cluster very large datasets with a low memory footprint.

Advantages:

- **Scalability:** The BIRCH algorithm can handle large datasets efficiently because it uses a set of CFs to represent the data, which reduces the amount of memory required to store the data.

- **Speed:** The BIRCH algorithm is generally faster than other hierarchical clustering algorithms because it reduces the number of distance calculations required.

- **Robustness:** The BIRCH algorithm is robust to noise and outliers because it uses a set of CFs to represent the data, which reduces the effect of noisy data points.

Disadvantages:

- Sensitivity to parameters: The BIRCH algorithm requires the user to specify several parameters, including the number of CFs, the threshold distance, and the number of clusters. These parameters can have a significant impact on the resulting clusters.
- Limited to spherical clusters: The BIRCH algorithm is designed to work with spherical clusters, which can limit its usefulness in datasets with complex structures.

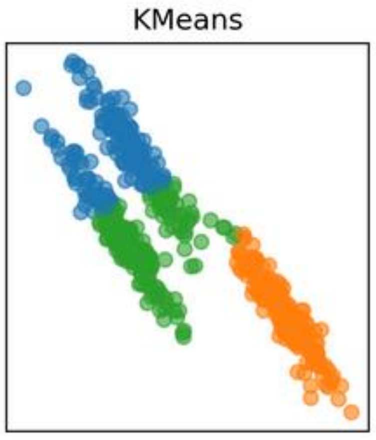
Overall, the BIRCH algorithm is a useful clustering algorithm for large datasets that can handle noise and outliers efficiently. However, it is sensitive to parameters and is limited to spherical clusters.

Gaussian Mixed Algorithm:

k-means clustering is a distance-based algorithm. This means that it tries to group the closest points to form a cluster.

Let's take the same income-expenditure example we saw above. The k-means algorithm seems to be working pretty well, right? Hold on – if you look closely, you will notice that all the clusters created have a circular shape. This is because the centroids of the clusters are updated iteratively using the mean value.

Now, consider the following example where the distribution of points is *not* in a circular form. What do you think will happen if we use k-means clustering on this data? It would still attempt to group the data points in a circular fashion. That's not great! **k-means fails to identify the right clusters**:



Hence, we need a different way to assign clusters to the data points. **So instead of using a distance-based model, we will now use a distribution-based model.** And that is where Gaussian Mixture Models come into picture!

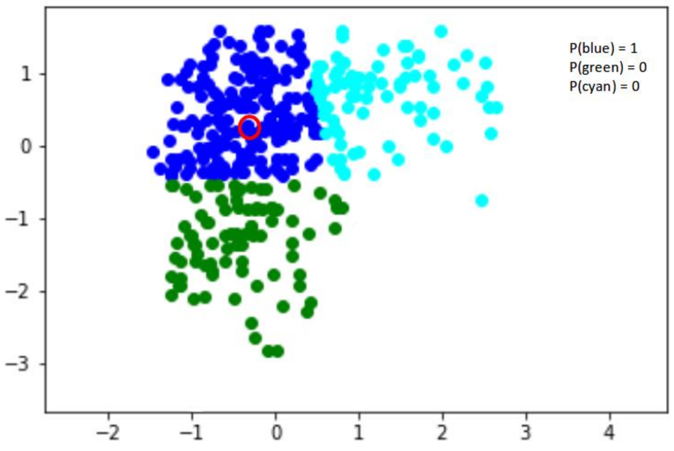
"Gaussian Mixture Models (GMMs) assume that there are a certain number of Gaussian distributions, and each of these distributions represent a cluster. Hence, a Gaussian Mixture Model tends to group the data points belonging to a single distribution together."

Let's say we have three Gaussian distributions (more on that in the next section) – GD1, GD2, and GD3. These have a certain mean (μ1, μ2, μ3) and variance (σ1, σ2, σ3) value respectively.

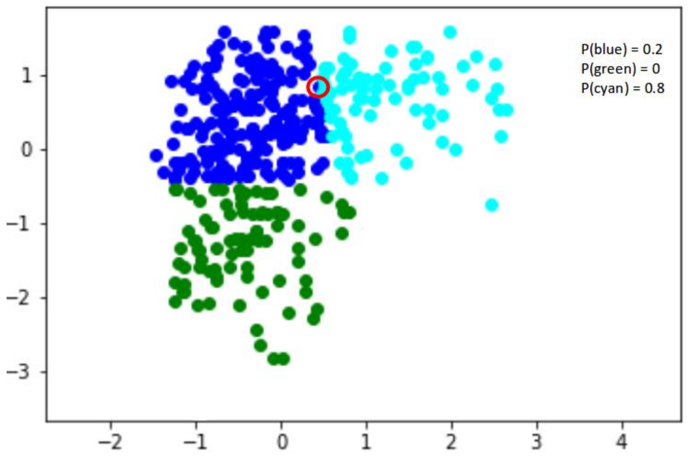
For a given set of data points, our GMM would identify the probability of each data point belonging to each of these distributions.

Gaussian Mixture Models are probabilistic models and use the soft clustering approach for distributing the points in different clusters. I'll take another example that will make it easier to understand.

Here, we have three clusters that are denoted by three colors – Blue, Green, and Cyan. Let's take the data point highlighted in red. The probability of this point being a part of the blue cluster is 1, while the probability of it being a part of the green or cyan clusters is 0.



Now, consider another point – somewhere in between the blue and cyan (highlighted in the below figure). The probability that this point is a part of cluster green is 0, right? And the probability that this belongs to blue and cyan is 0.2 and 0.8 respectively.



Gaussian Mixture Models use the soft clustering technique for assigning data points to Gaussian distributions.

Link:

Step by Step process to create Gaussian Mixed Cluster:

1. **Initialize the algorithm:** Specify the number of clusters, k, and initialize the means and covariances of the k Gaussian distributions. This can be done using a random initialization or a more sophisticated initialization method, such as k-means clustering.

2. **Calculate the posterior probability:** Calculate the probability density function (PDF) of each Gaussian distribution for each data point and normalize the probabilities so that they sum to 1. This can be done using Bayes' theorem and the multivariate Gaussian distribution.

3. **Assign each data point to a Gaussian distribution:** Assign each data point to the Gaussian distribution that has the highest posterior probability for that data point.

4. **Update the means and covariances:** Update the means and covariances of each Gaussian distribution based on the data points assigned to it. This is done by calculating the weighted mean and covariance of the data points assigned to each Gaussian distribution.

5. **Repeat steps 2-4 until convergence:** Repeat steps 2-4 until the algorithm converges. The convergence criteria can be based on the change in the likelihood of the data or the change in the parameters of the Gaussian distributions.

6. **Output the final set of clusters:** Output the final set of clusters, which correspond to the Gaussian distributions. Each data point is assigned to the cluster that corresponds to the Gaussian distribution with the highest posterior probability for that data point.

The Gaussian Mixture Cluster algorithm is an iterative algorithm that converges to a locally optimal solution. The quality of the solution depends on the initialization of the means and covariances of the Gaussian distributions. The algorithm can be used to cluster data points in various fields such as computer vision, finance, and biology.

Advantages:

- **Flexibility:** The algorithm can model complex data distributions by using a combination of Gaussian distributions.

- **Probabilistic:** The algorithm assigns each data point to a cluster based on the posterior probability of the data point belonging to each Gaussian distribution, which provides a measure of uncertainty.

- **Scalability:** The algorithm can handle large datasets efficiently by using a batch or online learning approach.

- **Robustness:** The algorithm is less sensitive to outliers compared to other clustering algorithms, such as k-means.

Disadvantages:

- **Initialization:** The quality of the solution depends on the initialization of the means and covariances of the Gaussian distributions, which can be challenging for high-dimensional data.

- **Convergence:** The algorithm can converge to a local optimum, which may not be the global optimum.

- **Computationally expensive:** The algorithm can be computationally expensive, especially for high-dimensional data or a large number of clusters.

- **Interpretation:** The interpretation of the clusters can be difficult, especially if the number of clusters is not well-defined or if the clusters overlap.

Clustering Evaluation Matrices:

Why do we need cluster validity indicates:

- To compare clustering algorithms.
- To compare two sets of clusters.
- To compare two clusters i.e., which one is better in terms of compactness and connectedness.
- To determine whether random structure exists in the data due to noise.

Generally, cluster validity measures are categorized into 3 classes, they are –

- **Internal cluster validation**: The clustering result is evaluated based on the data clustered itself (internal information) without reference to external information.
- **External cluster validation**: Clustering results are evaluated based on some externally known result, such as externally provided class labels.
- **Relative cluster validation**: The clustering results are evaluated by varying different parameters for the same algorithm (e.g., changing the number of clusters).

Besides the term cluster validity index, we need to know about **inter-cluster** distance d(a, b) between two cluster a, b and **intra-cluster** index D(a) of cluster a.

Inter-cluster distance d(a, b) between two clusters a and b can be –

- **Single linkage distance:** Closest distance between two objects belonging to *a* and *b* respectively.

- **Complete linkage distance:** Distance between two most remote objects belonging to *a* and *b* respectively.

- **Average linkage distance:** Average distance between all the objects belonging to *a* and *b* respectively.

- **Centroid linkage distance**: Distance between the centroid of the two clusters *a* and *b* respectively.

Intra-cluster distance D(a) of a cluster a can be –

- **Complete diameter linkage distance:** Distance between two farthest objects belonging to cluster a.
- **Average diameter linkage distance:** Average distance between all the objects belonging to cluster a.
- **Centroid diameter linkage distance**: Twice the average distance between all the objects and the centroid of the cluster a.

Clustering Performance Evaluation Metrics are used to evaluate the quality of clustering results. Here are some commonly used metrics:

- **Silhouette Coefficient:** The Silhouette Coefficient is a measure of how well a data point fits into its assigned cluster compared to other clusters. It ranges from -1 to 1, with values closer to 1 indicating that the data point is well-clustered. **

  Step by Step process to calculate: Wikipedia Link

  - For each data point i in a given clustering, calculate the average distance a(i) between i and all other data points in the same cluster.

$$a(i) = \frac{1}{|C_I| - 1} \sum_{j \in C_I, i \neq j} d(i, j)$$

  - For each data point i in a given clustering, calculate the average distance b(i) between i and all data points in the nearest neighbouring cluster (i.e., the cluster with the smallest average distance to i).

$$b(i) = \min_{J \neq I} \frac{1}{|C_J|} \sum_{j \in C_J} d(i, j)$$

  - Calculate the Silhouette Coefficient s(i) for each data point i using the formula: s(i) = (b(i) - a(i)) / max(a(i), b(i))
  - Calculate the overall Silhouette Coefficient for the clustering as the average of all s(i) values.

We now define a *silhouette* (value) of one data point i

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}, \text{ if } |C_I| > 1$$

and

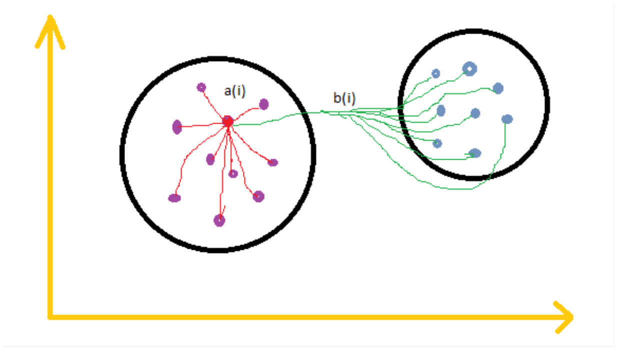
$$s(i) = 0, \text{ if } |C_I| = 1$$

Which can be also written as:

$$s(i) = \begin{cases} 1 - a(i)/b(i), & \text{if } a(i) < b(i) \\ 0, & \text{if } a(i) = b(i) \\ b(i)/a(i) - 1, & \text{if } a(i) > b(i) \end{cases}$$

From the above definition it is clear that

$$-1 \leq s(i) \leq 1$$



- **Davies-Bouldin Index:** The Davies-Bouldin Index measures the average similarity between each cluster and its most similar cluster, normalized by the sum of their dissimilarities. Lower values indicate better clustering.

- **Dunn Index:** The Dunn Index measures the minimum distance between clusters relative to the maximum diameter of each cluster. **(Or)** The Dunn Index is a measure of clustering quality that considers both the inter-cluster distances and the intra-cluster distances. Higher values indicate better clustering. ******

  Step by Step process:

  - Calculate the distance between each pair of clusters in the clustering, using a metric such as Euclidean distance.
  - For each cluster, calculate the diameter, which is the maximum distance between any two points in the cluster.
  - Calculate the minimum inter-cluster distance, which is the smallest distance between any two points in different clusters.
  - Calculate the Dunn Index as the ratio of the minimum inter-cluster distance to the maximum diameter of all clusters.

- **Rand Index:** The Rand Index measures the similarity between the true clustering and the predicted clustering. It ranges from 0 to 1, with values closer to 1 indicating better clustering. **

- **Jaccard Index:** The Jaccard Index measures the similarity between the true clustering and the predicted clustering, normalized by the sum of their dissimilarities. Higher values indicate better clustering. **

It is important to note that no single metric is sufficient for evaluating clustering performance, and multiple metrics should be used together to get a more comprehensive understanding of the quality of the clustering results.