Unsupervised learning - Clustering

The world of machine learning is full of diverse types of problems and accordingly a corresponding appropriate technique to resolve it to an accurate level. But at a major level, these techniques can be grouped into **Supervised** and **Unsupervised Learning techniques**. In a supervised learning technique, there has to be **input** and **output** labeled data for the algorithm to learn through. The algorithm is given an opportunity to **supervise** the data and gain some sense of **patterns** existing in it. The ultimate goal of such an algorithm is to be **accurate,** and **generic** when deployed to an **unseen/out-of-sample** data point. All such problems come under the domain of supervised learning. Examples of supervised learning problems are predicting the **price of a house**, **forecasting the temperature** of the next day, **predicting the stock price,** etc.

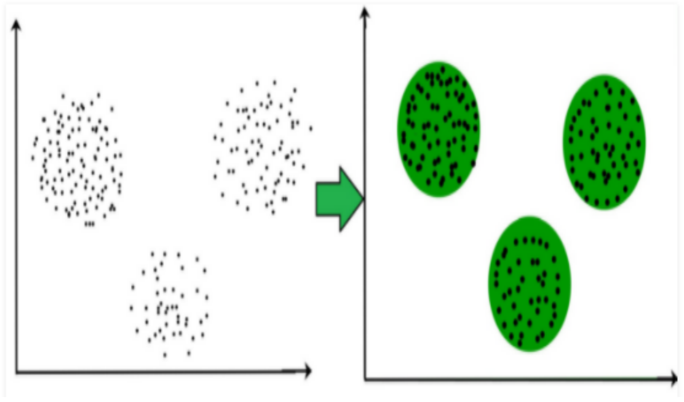
In the existing world, there are a lot of problems where it is not needed to avail the target label in the data. This is because the very nature of the problem is different and it does not require labeling. **Unsupervised learning** techniques are there to solve such problems where due to divergence in purpose, availing the target feature is not required. This technique is used to create groups with some common features among the given data points. The created groups are called **clusters** of the data. Below are a few examples where unsupervised learning is deployed -

1. **Customer segmentation** - It is one of the problems that is solved by unsupervised learning. Given a set of customer data containing features describing them, the goal is to create a number of customer groups called clusters, such that the customers in them show similarities.

2. **Blood group** - Given a set of patients' blood sample data, the goal is to create groups of people with similar profiles based on their corresponding data. Here there is no prior knowledge about the blood group available. After creating these groups or clusters, it is possible that each cluster has a similar blood group, and a few such clusters are created.

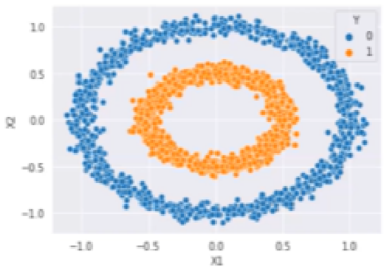
As we have understood the potential of unsupervised learning and its suitability in the present world, let us get into the methodology of how this technique works over the given data. Let us begin with understanding the term **cluster** first -

Cluster - A cluster is a group of **samples/data points** that have some common feature, generally taken from some big source of data. The data points in a cluster are similar to each other while those in different clusters are dissimilar to each other.



In the above plot, the left side plot is the original data which is supposed to have clusters. On the right-hand side in green color, three clusters are shown from the original data. Each and every green cluster contains data with high similarity, and the data in different clusters have a high dissimilarity.

While creating clusters in the data it would be a great help if we already know the possible number of existing clusters in the data. To do so, we take the help of **data visualization**. Then by visual inspection, one can identify the number of clusters present in the data. Looking at the below graph it won't take much time to understand that there are two clusters, one in the **outer circle** and the other in the **inner circle**. This inspection helps a lot in deploying a method to segregate the different clusters from the data.



The above data is **2-dimensional data,** where it is very easy to visually inspect the number of clusters. But in the real world, it is not always possible to get 2-dimensional data in every problem; rather the data is mostly high-dimensional. It becomes visually challenging then to find the number of clusters. For such cases, it becomes customary to use statistical techniques to find the number of clusters and get the clusters separated from the original data.

Before learning about the various clustering methods, it is important to understand:

1. How to identify the number of clusters in the existing data
2. How to find the similarity between two data points

Identification of the number of clusters in the existing data -

As we have seen, the human eye is very efficient in identifying the number of existing clusters in 2-dimensional data. Now to do the same in multi-dimensional data we deploy the following methods -

1. **t-SNE** - t-SNE is one of the most important visualization techniques in approximating the number of clusters in the dataset. The benefit of using t-SNE is that it preserves the nearest points from the original high-dimensional data, to the reduced low-dimensional version. Due to this benefit, it is recommended to do the visualization using t-SNE to get a good idea of the existing number of clusters. This method is just considered sufficient enough to find the number of clusters only. The major task after that is to segregate the clusters from the data. t-SNE is not supposed to be an ideal fit for that task. Below are the probable reasons for that -
    a. It loses the original distance between the data points when dimensions are reduced - As we know, from a higher dimension the data distribution comes to lower dimensions in this method. Since the lower-dimensional space is more **compact** than that in higher dimensions, the projected distance becomes less than that in the original dimension. Due to this, it is not considered a suitable method for clustering.
    b. It loses the original density of data points when dimensions are reduced - Due to the change in volume and space in the lower dimensions the density of the data changes in the lower dimension. As an effect, the clusters made can not be trusted to be ideal ones.
    c. t-SNE tries to optimize scale divergence. When a new data point comes in, the higher dimension's structure changes, and hence the reduced dimension structure also changes as an effect. Since the optimization problem already takes too much time, it is not preferred to proceed directly with t-SNE. Instead, the dimensions are first reduced by removing noise using PCA and then t-SNE is applied

2. PCA - It can also be used to find the number of clusters in the data. But the issue with this is it faces problems in reducing the dimensionality to a very low one. Once we try to reduce the dimensions more, it starts losing the **variability** component of the data. The lower the reduced

dimensions, the lower the retained variability from the original data.

As an effect, it is preferred to use PCA for removing **noise** from the original data and then apply t-SNE to find the number of existing clusters. **Noise** can be considered as the data points that are not adding much to the variability of the data, or a feature with very low variance.

Now that we've understood the methods to estimate the existing number of clusters, let us understand how to measure the similarity between data points -

How to find the similarity between two data points

Understanding similarity in a general context is easy stuff for humans. But in the context of clustering, every data point is just represented by a certain set of numbers or a vector. Finding whether two data points are similar to each other or not then, certainly needs mathematical intervention. The most suitable quantity used to find similarity between two points is the **distance** between them. In a multi-dimensional feature space, distance seems to be a more appropriate way of finding similarities. Points that are closer to each other are considered similar points. This is because a small distance ensures a **very small difference/high similarity** in the existing dimensions of the data points. Now, there are multiple kinds of distances that are used to measure the similarity between two points.

The different types of distances are as follows -

1. **Euclidean distance** - Euclidean distance is the distance between two points in a feature space. It is the **most widely** used distance measuring method for any type of data. Mathematically it can be given as follows -

$$d(x^{(i)} - x^{(j)}) = \sqrt{(x_1^{(i)} - x_1^{(j)})^2 + (x_2^{(i)} - x_2^{(j)})^2 + \ldots. + (x_p^{(i)} - x_p^{(j)})^2}$$

   Here $d(x^{(i)} - x^{(j)})$ is the distance between the two points, $x^{(i)}$ and $x^{(j)}$, each of the points is having p dimensions. This method uses l_2 normalization as the power of each individual difference term in the formula is 2.

2. **Manhattan distance** - This method uses the perpendicular distance along certain axes to find the distance. Due to this, the name of this distance is termed as **Manhattan distance**, as in the borough of Manhattan, the streets have been made very perpendicular to each other. Mathematically it can be calculated as follows -

$$d(x^{(i)} - x^{(j)}) = |x_1^{(i)} - x_1^{(j)}| + |x_2^{(i)} - x_2^{(j)}| + ... + |x_p^{(i)} - x_p^{(j)}|$$

This is called as l_1 norm because it is using the first power of difference between the points.

3. **Maximum distance**:- It calculates the distance along all the dimensions between two points and then picks the maximum one.

$$d(x^{(i)} - x^{(j)}) = max_{k=1,2,...p} |x_k^{(i)} - x_k^{(j)}|$$

To understand this let us take an example, the age difference between two people is 10 and the difference of income is 10000. Then the max distance will be 10000.

So these are the methods to compute the distance between any two points in a specific feature space. There is no **order of superiority** between these distances. One can be used in place of another without any bias and priority. While computing the distance between two points, there is a possibility that the two points belong to different units. The coordinate of a single point consists of multiple features. The distance computing method does not bother about units but only cares for the magnitude. If there is a scale difference between the features, normalization can be done to omit that.

Now let us get into the different methods that are used to create clusters. Majorly there are four methods to create clusters based on the suitability of the application where they will be deployed for clustering the data. They are as follows -

1. K-means clustering
2. Gaussian mixture model
3. Hierarchical clustering
4. DBSCAN

Let us go one by one into their details -

1. **K-means clustering** - As the name suggests it creates K clusters from the given data. It starts with fixing the number of clusters K. K can be estimated by either visual inspection of the data (preferably if it is in 2 dimensions) or by using the t-SNE method to estimate the number of existing clusters in the data. As we know in machine learning, to fit an algorithm best to the data there has to be an optimization function that will best fit the algorithm. In the case of K-means, let us proceed with the loss function associated with this algorithm.

$$WGSS = \sum_{k=1}^{K} \sum_{C(x^{(i)})=k} \sum_{C(x^{(j)})=k} d(x^{(i)}, x^{(j)})^2$$

Here $WGSS$ is **within groups sum of squares**. It represents the sum of squares of existing points in a cluster from the corresponding centroid or means of the cluster. This is why the name “K-means” of the algorithm. Lower the value of $WGSS$ better the quality of the cluster. The intuition of this loss function can also be interpreted as maximizing the sum of squares from between the groups or existing clusters. As we know from a certain point **the sum of squares of all the points is a constant**, so minimizing the within-group will automatically maximize the between-group sum of squares. The above loss function characterizes the extent to which observations assigned to the same cluster tend to be close to one another or observations between different clusters are further apart from each other.

This is also the distance of each point from the mean. It can be interpreted as follows -

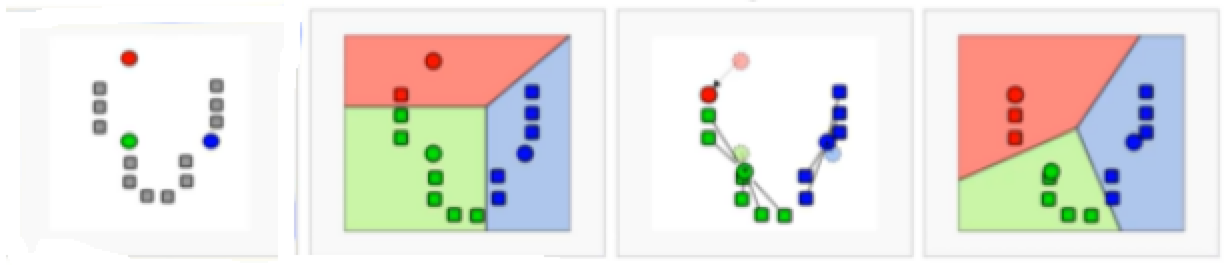
$$WGSS = \sum_{k=1}^{K} \sum_{C(x^{(i)})=k} \sum_{C(x^{(j)})=k} d((x^{(i)} - mean) - (x^{(j)} - mean))^2$$

The mean term gets canceled and the loss function can be listed as the following intuitions -

1. The within-group sum of squares
2. The between-group sum of squared
3. Distance from mean

Let us understand the working of K-means with K=3 with the help of the below diagram.

We begin with the assumption that three clusters are existing in the data and choose three random centroids or means shown in the first diagram as red, green, and blue dots. They are not the actual data points but they might be that also.



Now each and every data point is grouped with either of the three centroids depending on which point is closest. Then in the next iteration, the **means or centroids** are revised to get a new set of three centroids. They are computed by taking the mean of data points that are there in existing clusters in the first iteration. This time multiple points will change their clusters from the last iteration. This process is repeated till there is no change in total points in the cluster. In such cases, the clusters can be considered to be stabilized. A good cluster is at the end as compact as possible and preferred to be in a round shape. It depicts the quality of clustering.

Now that we understand the working principle, let us look into the drawbacks of the algorithm -

Drawbacks of the K-means algorithm

1. Depending on the clusters selected in the beginning, the final clusters may change. For example, if we begin with three different points in the above figure than the current points, it is possible that in the end we may have got slightly different clusters.

2. The second drawback of K-means is that any individual point is associated with only one cluster.

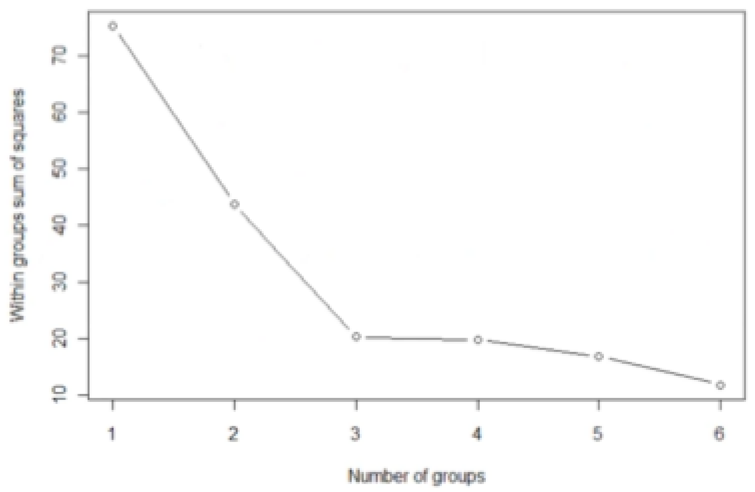
K-means clustering is considered a greedy algorithm because for every point it is trying to put in the closest cluster. So it is greedy about getting into the closest cluster.

While dealing with supervised learning techniques we are aware of the impact of outliers present in the data on any kind of analysis and insights drawn through the entire process. Outliers also impact a lot the working and outcome of unsupervised learning.

Impact of outliers in clusters

As we know, outliers are the points in the dataset that are more than $1.5 \times IQR$ greater than the third quartile or lesser than the first quartile. So if we have outliers in the dataset then it impacts the final clusters. In the presence of the outliers, the change in mean in every iteration will be much more than if the outlier is not there. To deal with this in the case of K-means clustering one of the solutions is, to begin with, centroid that is the actual data points of the dataset. In such a case, it resists the change in mean on every iteration more than the random selection of centroids. This way without removing the outliers from the data one can handle them and diminish their impact.

Now let us understand a good way of selecting the number of clusters present in a certain dataset using the K-means algorithm. It begins with finding the WGSS value for different values of K (the number of clusters). The plot of the within-group sum of squares with the number of clusters is called the **elbow plot** because it is in the shape of a human elbow. The below plot shows a sample elbow plot. To find the most appropriate number of clusters present in the data we pick the value of K after the last big drop. According to the figure below WGSS has two big drops till K=3.



It is not recommended to select the minimum WGSS because as K increases the value of WGSS keeps on decreasing. For the above plot, the K value is 3. It means the number of appropriate clusters is three in the given dataset.

Gaussian mixture model

It is another method of doing clustering in machine learning. It allows a single data point to be in different clusters with some corresponding **probabilities**. For example, a certain point A can belong to two clusters c and d with probabilities say 86% and 14%. The closer the points to the cluster, the higher the probability of that point belonging to that cluster. So while computing the mean value in any iteration in this method it is calculated as the weighted average of the distance from the mean with the probability of that point belonging to a certain cluster.

Some of the key features of this method are as follows -

1. To understand the advantage of this method let us take an example of a patient going to the hospital for treatment. Assume that the doctor is sure about some of the clusters of diseases but he found this patient to be in between those clusters. So it will help the doctor in selecting the probability of which treatment to provide to the patient.

2. For K means to deal with outliers, we were taking the actual data points as initial clusters and the problem of high variance in the mean was resolved. But in the **Gaussian method**, this is not going to work. For this, the solution is to remove the outliers before beginning the process of clustering. So first remove outliers from it using **PCA** then apply the **Gaussian method**.

The disadvantage associated with this method is explained below -

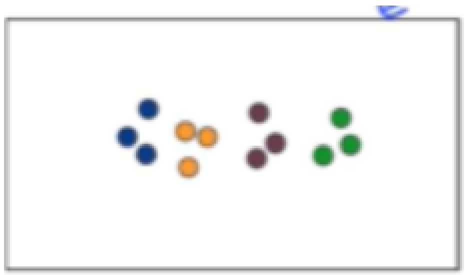
- Due to the involvement of the probability component associated with each point in the dataset, the method becomes more mathematical and statistics-dependent.

Hierarchical clustering

In hierarchical clustering, we begin with considering all the points to be individual clusters. Then we start finding the points closest to each individual point and merge them to create clusters. So in the beginning we have a certain number of clusters that are to be merged to create bigger clusters. To do so we need to understand the distance between two clusters. Because this will help us bind two different clusters efficiently to create bigger clusters.

Below are the methods to do so -

1. **Complete linkage** - Here it creates perfect but small clusters by merging the very small clusters. It uses the maximum distance between the two clusters to measure the distance between them. In the below diagram it can be observed that small but compact and clear clusters are created.



Below is the mathematical expression for such clusters -

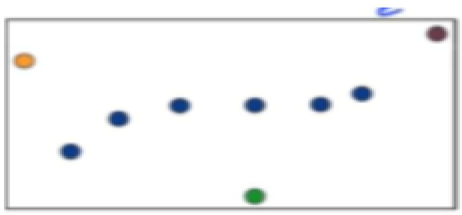
$$d(C_r, C_s) = max_{x^{(i)} \epsilon C_r, x^{(j)} \epsilon C_s,} d(x^{(i)}, x^{(j)})$$

Where $d(C_r, C_s)$ is the distance between two clusters C_r and C_s

2. **Single linkage** - In this method, a probable single line of points is identified using the minimum distance method. Here the two distinct clusters are merged by using the minimum possible distances between the existing points between them. In the below figure it can be seen that a chain of blue points is forming the merged clusters. Mathematically it can be defined as follows

$$d(C_r, C_s) = min_{x^{(i)} \epsilon C_r, x^{(j)} \epsilon C_s,} d(x^{(i)}, x^{(j)})$$

Where symbols have their usual meaning.



3. **Average linkage** - Here it takes the mean/centroid of two clusters and then computes the distance between them. The below figure shows the average linkage of clusters.

Mathematically it can be given as follows -

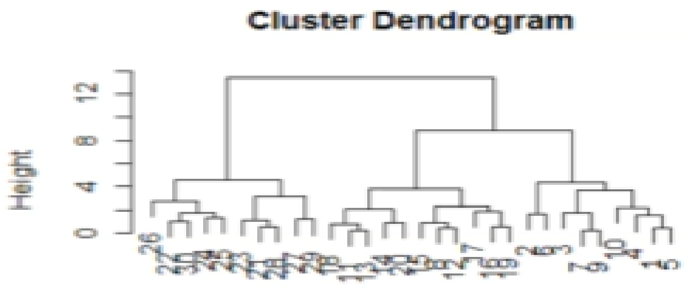
$$d(C_r, C_s) = \frac{1}{n_r} \times \frac{1}{n_s} \sum_{x^{(i)} \epsilon C_r} \sum_{x^{(i)} \epsilon C_s} d(x^{(i)}, x^{(j)})$$

n_r is the number of data points in cluster r while n_s is the same in cluster s.

Other symbols have their usual meaning.

Now let us understand how to choose a number of clusters in this method. The foremost thing to understand is that there is no hard and fast rule followed for this. So the estimated count may vary a little and some other value of cluster count may work better.

To estimate the number of clusters we will take the help of a plot called a **Dendrogram**. A dendrogram is a tree shape structure that represents hierarchical clusters existing in a given set of data. It follows the hierarchy from top to bottom or bottom to top. It is used to represent the cluster and also to find the number of clusters also. It becomes visually easy to inspect the existing clusters and corresponding similarities and dissimilarities between the clusters. To find the number of clusters we have the choice to draw horizontal lines in the dendrogram that divides the entire data into multiple similar groups called clusters. Still, there is no hard and fast rule for that also. But a preferred way is to find a line where the maximum vertical drop is there in the dendrogram.



With this let us look into the pros and cons of using the hierarchical clustering method.

Pros of hierarchical clustering are as follows -

1. It shows all the possible linkages between the clusters as it is presented by the well-organized plot named **Dendrogram**.
2. It gives **more understanding** of the data due to visual presentation.
3. There is no need to set the number of clusters in the beginning. Initially, the possible number of clusters is created, then by inspecting it the final number of clusters is selected.

Cons of hierarchical clustering -

1. It becomes very hard to interpret when the number of data points is very high and hence the count of clusters. Higher the count of observations the more complex it gets accordingly.

Hierarchical clustering works in two possible ways -

- **Agglomerative clustering** - Here the clustering starts with individual points and then the aggregation of points is done based on the nearest points. So it builds clusters from individual data points.

- **Divisive clustering** - This is the opposite approach. It starts by taking the entire dataset as a cluster and then splitting it to create different clusters.

In this type of clustering, the distance between two clusters is defined as either the minimum distance between two possible points or the maximum distance between two possible points.

DBSCAN

It is a very new method that is understood to be the generalization of **single linkage hierarchical clustering**. It uses certain parameters, which are explained below -

a. **minPts**: The minimum number of points (a threshold) clustered together for a region to be considered dense or a cluster.
b. **eps (ε)**: A distance measure that will be used to locate the points in the neighborhood of any point.

Below are the steps followed in this algorithm -

a. In the beginning, the algorithm starts by randomly picking up a point in the dataset.
b. If there are at least **"minPts"** points within a radius of **"ε"** to the point then all such points are considered to be part of the same cluster.
c. The clusters are expanded by following the same steps ahead.

One of the major benefits of this method is it handles the **outliers** existing in the cluster by itself.

There are many methods of creating the clusters but one of the most needed things in clustering is the quality of the cluster. To estimate this we use a **Silhouette plot**. It is explained below -

Silhouette plot -

This plot displays a measure of how close each point in one cluster is to points in the neighboring clusters. To understand it let us go through the following steps -

For all the points $x^{(i)}$let us compute -

a. The average dissimilarity between $x^{(i)}$and all other points in its cluster. It is the distance within the cluster and is denoted by an $(x^{(i)})$.

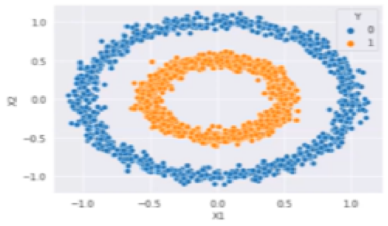
b. $b(x^{(i)})$ = average distance between $x^{(i)}$ and the nearest cluster to which it does not belong. So it is the distance between the clusters

$s(x^{(i)})$ ranges from -1 to +1 and it is calculated as follows -

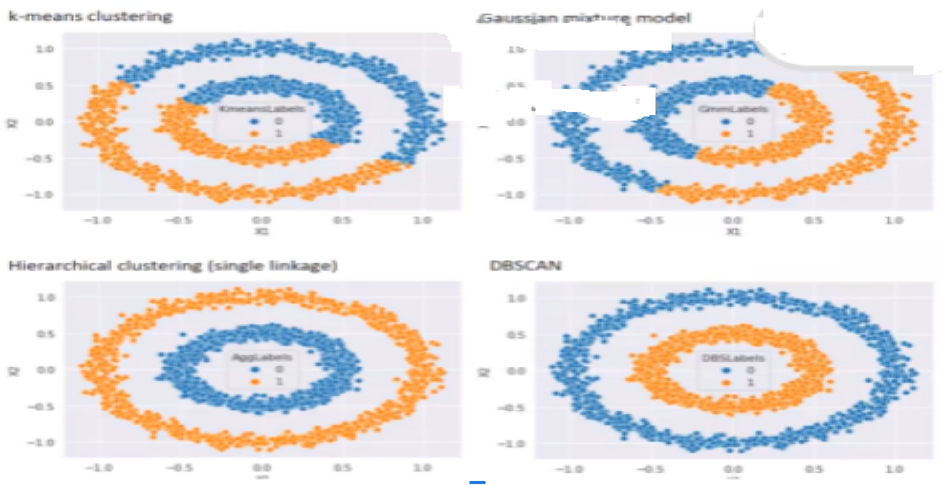
$$s(x^{(i)}) = \frac{(b(x^{(i)}) - a(x^{(i)})}{max(a(x^{(i)}), b(x^{(i)}))}$$

Higher the value of $s(x^{(i)})$better the quality of clusters created. Lower the value, the worse the clusters. The threshold value of $s(x^{(i)})$taken under consideration is **0.5**. Above this is a good cluster while below will be a bad one.

After understanding the qualitative measures let us deploy these methods to a single problem and see their performance and suitability. In the below image the target is to identify the existing clusters in two existing colors.



To do so all the existing algorithms are deployed to solve the task. **Hierarchical,** and **DBSCAN** are able to identify the clusters correctly while **Kmeans** and **Gaussian** methods failed to do so. The performance plot of all these algorithms are given below -



From the figure, it can be understood correctly that **Hierarchical** and **DBSCAN** clustering methods are able to do that while other algorithms fail to do so. The reason for this is that K-means and Gaussian methods could not find the clusters because they work with an ellipsoid and it tries to find clusters inside of the ellipsoid.

Community detection

In today's world, it is now required to deploy clustering into network analysis. Community detection is one of the sample problems where it requires clustering techniques in network analysis. It needs to identify a group of similar people among a population of people. Such problems involve nodes and edges. To compare the similarity it needs to associate a distance between the nodes. Below are the measures possibly useful for measuring the distance between nodes -

1. **Geodesic distance** - It is the shortest distance between two nodes of a network.
2. Number of different neighbors
3. Correlation between adjacency matrix columns.

Among all the above distances, **Geodesic distance** is the most used one. Similarity measures can be done using these distances.

There are a few more intuitions that help during clustering in the network.
1. Using betweenness centrality
2. Modularity maximization

Let us go through each of them -
1. **Using betweenness centrality** - This is associated with both edges and nodes in the network. Let us understand them -
    a. In the sense of **edges**, it is associated with the edges through which most paths go through. In a network, if these edges are removed then the remaining nodes will form the requisite clusters.
    b. While in concern of **nodes**, if there is a node that is in between a large number of pairs of nodes, they will be good to find clusters
2. **Modularity maximization** - Modularity is a quality function that is associated with a certain partition of a network or the cluster existing in a network. Mathematically it can be given as

follows -

$$Q = (\frac{1}{2m}) \sum_{ij}(A_{ij} - P_{ij})\, \delta(C_i, C_j)$$

Where P_{ij} is the expected number of edges between i and j in a null model.

To do community detection the method deployed is the **“Louvain method”**. It is similar to Hierarchical clustering where it uses modularity as a measure of similarity. This method can work on a very huge number of data points. The steps followed in this method are as follows -

- Put each node in its own community
- Put node i into community j that yields the biggest increase in modularity
- Replace communities with supernodes, where edge weights between supernodes are the sum of edge weights between corresponding nodes
- Iterate the process till modularity cannot be improved

Reference -

- For clustering
  - Chapter 14 in T.Hastie, R.TIbshirani, & J. Friedman, The elements of statistical learning: Data mining, inference, and prediction. Springer, 2009.
- For community detection in networks :
  - V.D. Blondel, et al. Fast unfolding of communities in large networks. Journal of Statistical Mechanics: Theory and experiment 10, 2008.
  - S.Fortunato. community detection in the graph. Physics Reports 486, 2010.
  - Lecture notes on Laplacian and spectral clustering (prominent method not discussed in this module) by T . Roughgarden & G. Valiant: http://web.stanford.edu/class/cs168/1/111/pdf
- For Clustering images - https://www.geeksforgeeks.org/clustering-in-machine-learning/