

Unit 4 Choosing the number of clusters



OUTLINE:

- ❖ Choosing the number of clusters

Choosing the number of clusters

Clustering in Machine Learning:

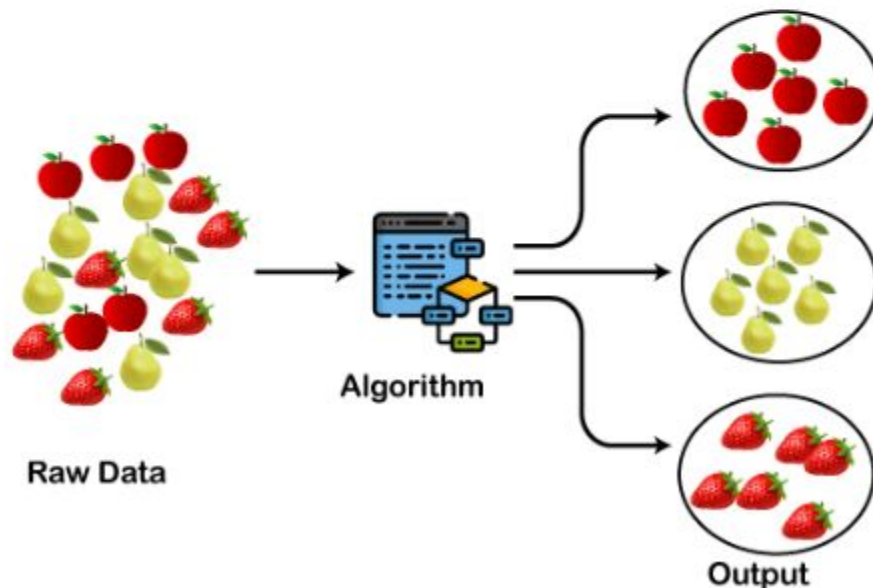
- Clustering or cluster analysis is a machine learning technique, which groups the unlabelled dataset. It can be defined as *"A way of grouping the data points into different clusters, consisting of similar data points. The objects with the possible similarities remain in a group that has less or no similarities with another group."*
- It does it by finding some similar patterns in the unlabelled dataset such as shape, size, color, behavior, etc., and divides them as per the presence and absence of those similar patterns.
- It is an [unsupervised learning](#) method, hence no supervision is provided to the algorithm, and it deals with the unlabeled dataset.
- After applying this clustering technique, each cluster or group is provided with a cluster-ID. ML system can use this id to simplify the processing of large and complex datasets.

Choosing the number of clusters

- Market Segmentation
- Statistical data analysis
- Social network analysis
- Image segmentation
- Anomaly detection, etc.

Apart from these general usages, it is used by the **Amazon** in its recommendation system to provide the recommendations as per the past search of products. **Netflix** also uses this technique to recommend the movies and web-series to its users as per the watch history.

The below diagram explains the working of the clustering algorithm. We can see the different fruits are divided into several groups with similar properties.



Choosing the number of clusters

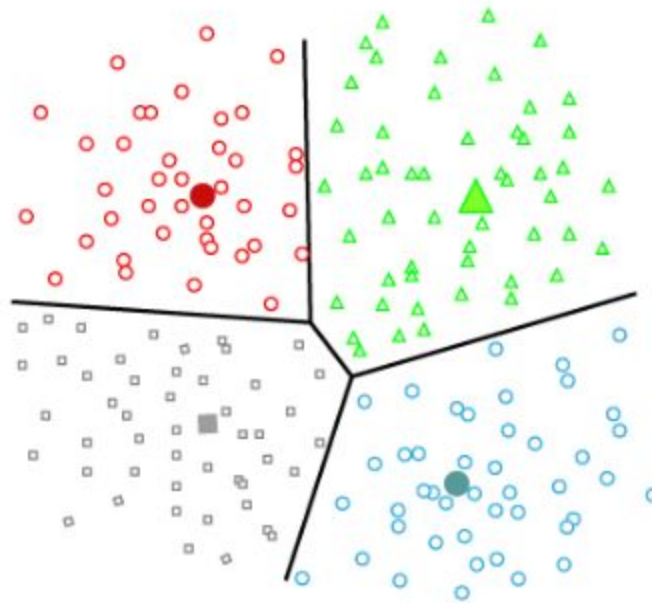
Types of Clustering Methods

- The clustering methods are broadly divided into **Hard clustering** (datapoint belongs to only one group) and **Soft Clustering** (data points can belong to another group also).
- But there are also other various approaches of Clustering exist. Below are the main clustering methods used in Machine learning:
 - Partitioning Clustering**
 - Density-Based Clustering**
 - Distribution Model-Based Clustering**
 - Hierarchical Clustering**
 - Fuzzy Clustering**

Choosing the number of clusters

Partitioning Clustering:

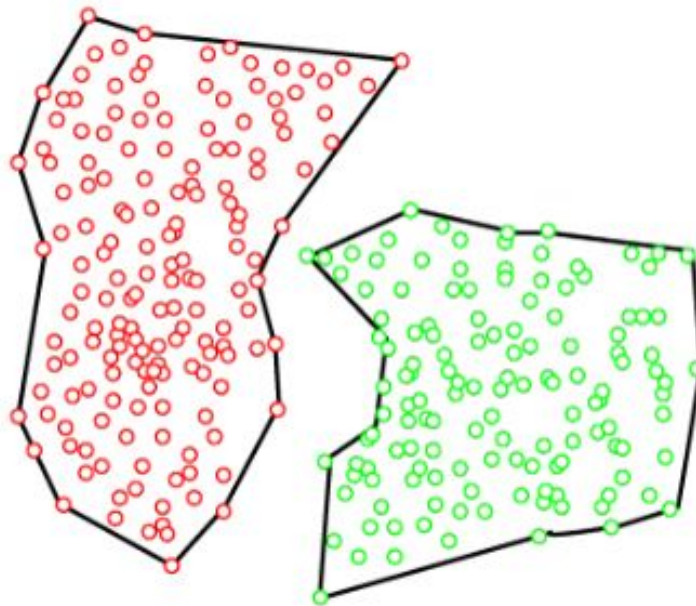
- It is a type of clustering that divides the data into non-hierarchical groups. It is also known as the **centroid-based method**. The most common example of partitioning clustering is the [K-Means Clustering algorithm](#).
- In this type, the dataset is divided into a set of k groups, where K is used to define the number of pre-defined groups. The cluster center is created in such a way that the distance between the data points of one cluster is minimum as compared to another cluster centroid.



Choosing the number of clusters

Density-Based Clustering:

- The density-based clustering method connects the highly-dense areas into clusters, and the arbitrarily shaped distributions are formed as long as the dense region can be connected. This algorithm does it by identifying different clusters in the dataset and connects the areas of high densities into clusters. The dense areas in data space are divided from each other by sparser areas.
- These algorithms can face difficulty in clustering the data points if the dataset has varying densities and high dimensions.

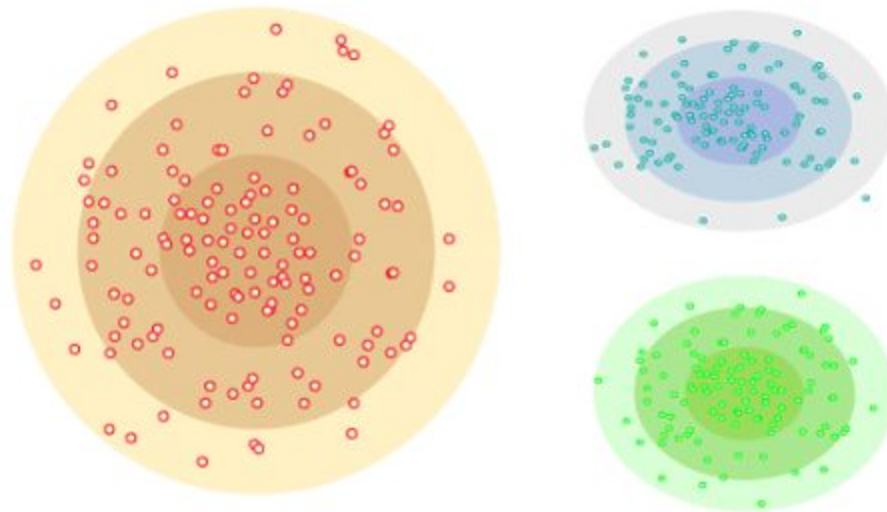


Choosing the number of clusters

Distribution Model-Based Clustering:

In the distribution model-based clustering method, the data is divided based on the probability of how a dataset belongs to a particular distribution. The grouping is done by assuming some distributions commonly **Gaussian Distribution**.

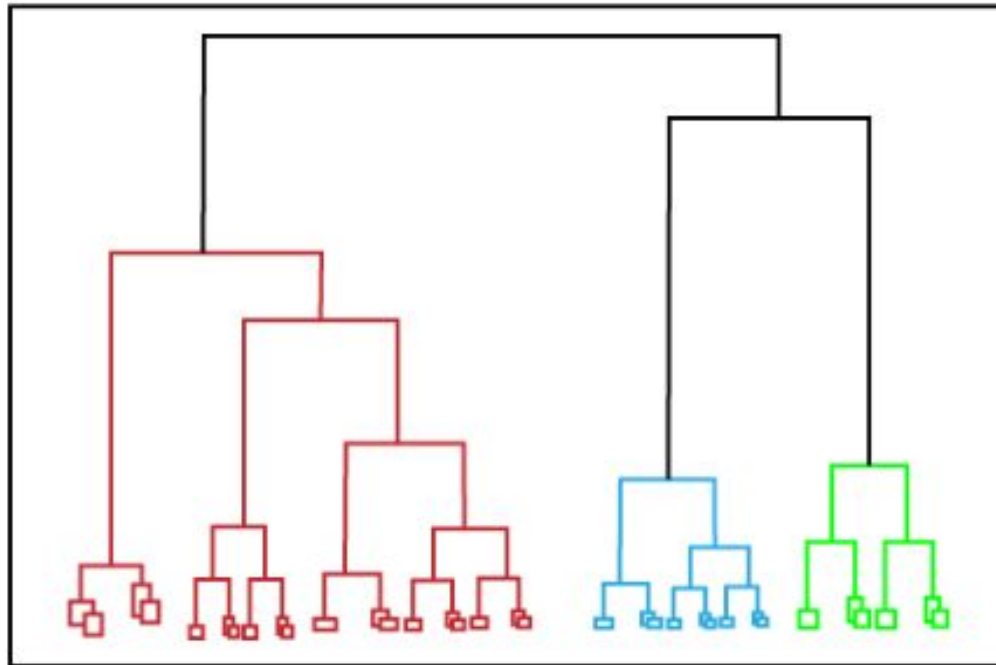
The example of this type is the **Expectation-Maximization Clustering algorithm** that uses Gaussian Mixture Models (GMM).



Choosing the number of clusters

Hierarchical Clustering:

Hierarchical clustering can be used as an alternative for the partitioned clustering as there is no requirement of pre-specifying the number of clusters to be created. In this technique, the dataset is divided into clusters to create a tree-like structure, which is also called a **dendrogram**. The observations or any number of clusters can be selected by cutting the tree at the correct level. The most common example of this method is the **Agglomerative Hierarchical algorithm**.



Choosing the number of clusters

Fuzzy Clustering:

Fuzzy clustering is a type of soft method in which a data object may belong to more than one group or cluster. Each dataset has a set of membership coefficients, which depend on the degree of membership to be in a cluster. **Fuzzy C-means algorithm** is the example of this type of clustering; it is sometimes also known as the Fuzzy k-means algorithm.

Choosing the number of clusters

Applications of Clustering:

In Identification of Cancer Cells: The clustering algorithms are widely used for the identification of cancerous cells. It divides the cancerous and non-cancerous data sets into different groups.

In Search Engines: Search engines also work on the clustering technique. The search result appears based on the closest object to the search query. It does it by grouping similar data objects in one group that is far from the other dissimilar objects. The accurate result of a query depends on the quality of the clustering algorithm used.

Customer Segmentation: It is used in market research to segment the customers based on their choice and preferences.

In Biology: It is used in the biology stream to classify different species of plants and animals using the image recognition technique.

In Land Use: The clustering technique is used in identifying the area of similar lands use in the GIS database. This can be very useful to find that for what purpose the particular land should be used, that means for which purpose it is more suitable.

Unit 4 Hierarchical Clustering in Machine Learning



OUTLINE:

- ❖ Hierarchical Clustering in Machine Learning

Hierarchical Clustering in Machine Learning

Clustering Techniques:

- Hierarchical clustering is another unsupervised machine learning algorithm, which is used to group the unlabeled datasets into a cluster and also known as **hierarchical cluster analysis** or HCA.
- In this algorithm, we develop the hierarchy of clusters in the form of a tree, and this tree-shaped structure is known as the **dendrogram**.
- Sometimes the results of K-means clustering and hierarchical clustering may look similar, but they both differ depending on how they work.
- As there is no requirement to predetermine the number of clusters as we did in the K-Means algorithm.

Hierarchical Clustering in Machine Learning

The hierarchical clustering technique has two approaches:

- **Agglomerative:** Agglomerative is a **bottom-up** approach, in which the algorithm starts with taking all data points as single clusters and merging them until one cluster is left.
- **Divisive:** Divisive algorithm is the reverse of the agglomerative algorithm as it is a **top-down approach**.

Agglomerative Hierarchical clustering

Why hierarchical clustering?

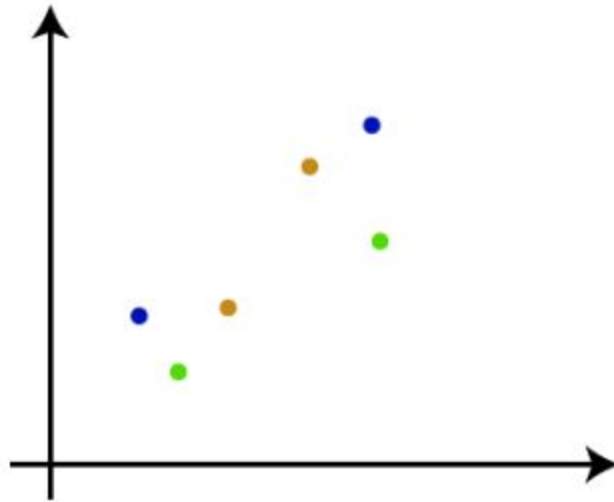
- As we already have other [clustering](#) algorithms such as [K-Means Clustering](#), then why we need hierarchical clustering? So, as we have seen in the K-means clustering that there are some challenges with this algorithm, which are a predetermined number of clusters, and it always tries to create the clusters of the same size.
- To solve these two challenges, we can opt for the hierarchical clustering algorithm because, in this algorithm, we don't need to have knowledge about the predefined number of clusters.

Agglomerative Hierarchical clustering

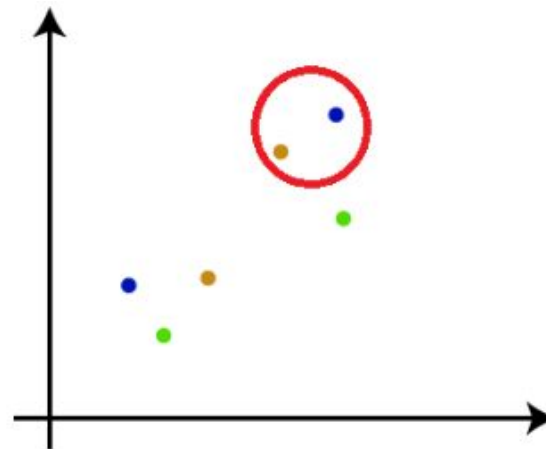
The agglomerative hierarchical clustering algorithm is a popular example of HCA. To group the datasets into clusters, it follows the **bottom-up approach**. It means, this algorithm considers each dataset as a single cluster at the beginning, and then start combining the closest pair of clusters together. It does this until all the clusters are merged into a single cluster that contains all the datasets.

How the Agglomerative Hierarchical clustering Work?

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

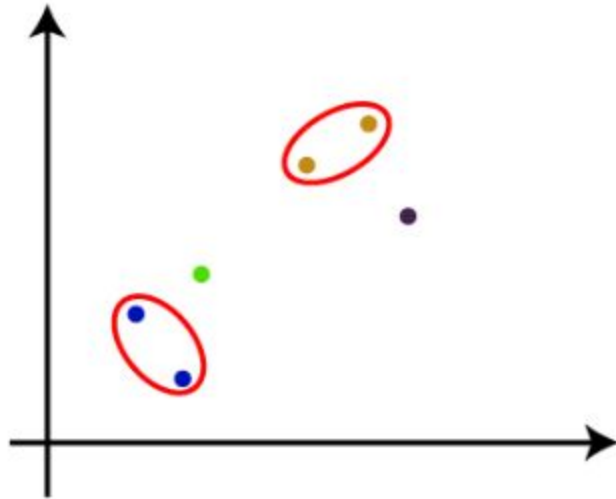


Step-2: Take two closest data points or clusters and merge them to form one cluster. So, there will now be $N-1$ clusters.

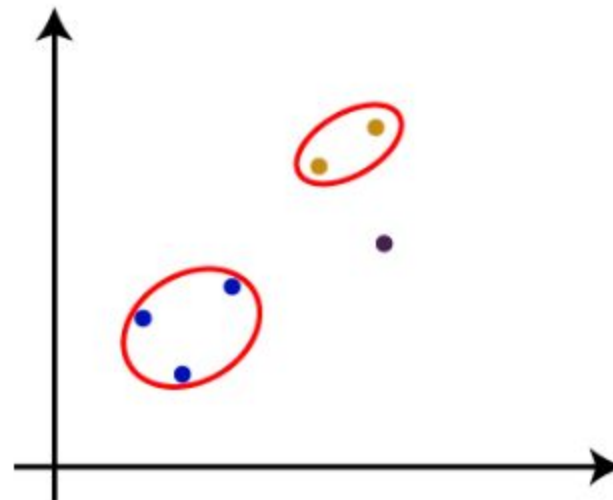


How the Agglomerative Hierarchical clustering Work?

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

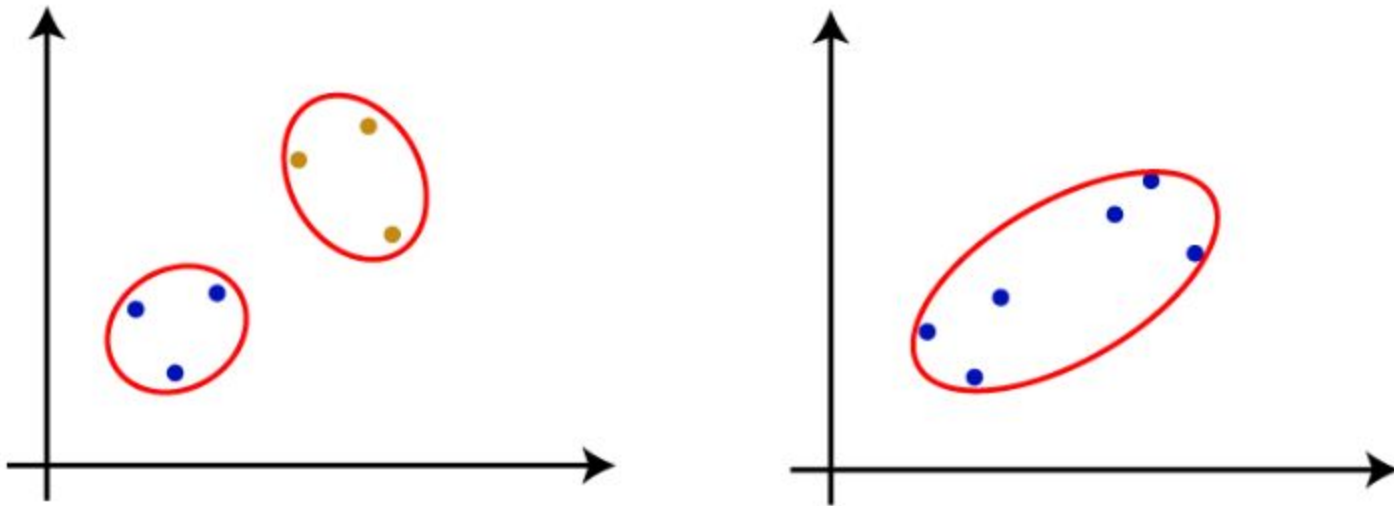


Step-4: Repeat Step 3 until only one cluster left. So, we will get the following clusters. Consider the below images:



How the Agglomerative Hierarchical clustering Work?

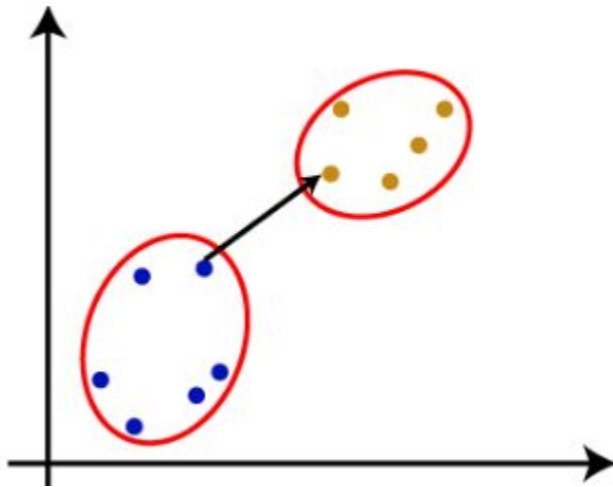
Step-3: Again, take the two closest clusters and merge them together to form one cluster. There will be $N-2$ clusters.



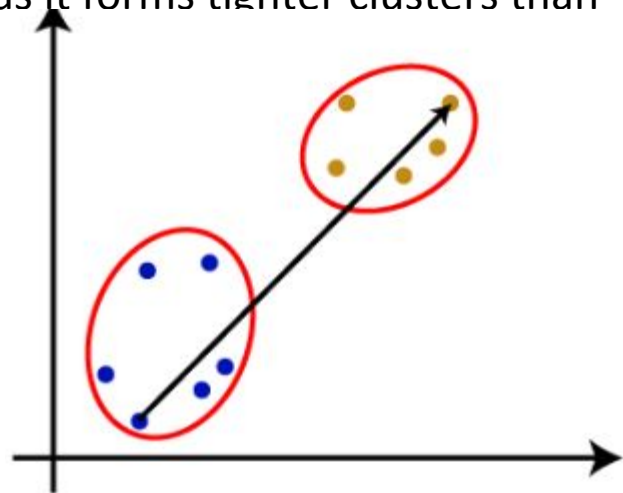
Step-5: Once all the clusters are combined into one big cluster, develop the dendrogram to divide the clusters as per the problem.

Measure for the distance between two clusters

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



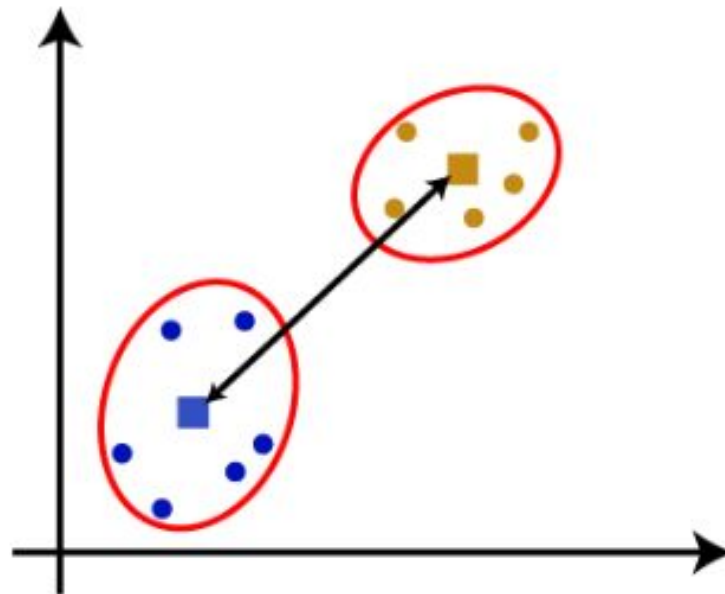
2. Complete Linkage: It is the farthest distance between the two points of two different clusters. It is one of the popular linkage methods as it forms tighter clusters than single-linkage.



Measure for the distance between two clusters

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.

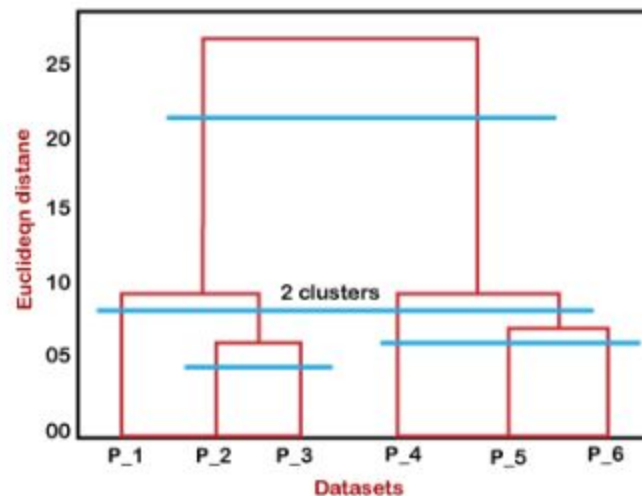
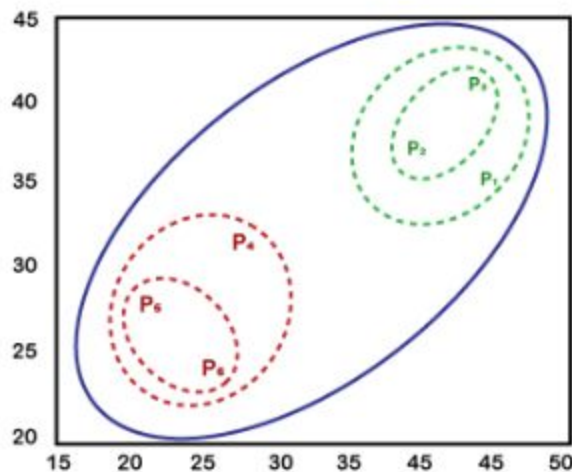
4. Centroid Linkage: It is the linkage method in which the distance between the centroid of the clusters is calculated. Consider the below image:



Working of Dendrogram in Hierarchical clustering

The dendrogram is a tree-like structure that is mainly used to store each step as a memory that the HC algorithm performs. In the dendrogram plot, the Y-axis shows the Euclidean distances between the data points, and the x-axis shows all the data points of the given dataset.

The working of the dendrogram can be explained using the below diagram:



Working of Dendrogram in Hierarchical clustering

In the above diagram, the left part is showing how clusters are created in agglomerative clustering, and the right part is showing the corresponding dendrogram.

As we have discussed above, firstly, the datapoints P2 and P3 combine together and form a cluster, correspondingly a dendrogram is created, which connects P2 and P3 with a rectangular shape. The height is decided according to the Euclidean distance between the data points.

In the next step, P5 and P6 form a cluster, and the corresponding dendrogram is created. It is higher than of previous, as the Euclidean distance between P5 and P6 is a little bit greater than the P2 and P3.

Again, two new dendrograms are created that combine P1, P2, and P3 in one dendrogram, and P4, P5, and P6, in another dendrogram.

At last, the final dendrogram is created that combines all the data points together.

Python Implementation of Agglomerative Hierarchical Clustering

Steps for implementation of AHC using Python:

The steps for implementation will be the same as the k-means clustering, except for some changes such as the method to find the number of clusters.

Below are the steps:

Data Pre-processing

Finding the optimal number of clusters using the Dendrogram

Training the hierarchical clustering model

Visualizing the clusters

Python Implementation of Agglomerative Hierarchical Clustering

Data Pre-processing Steps:

In this step, we will import the libraries and datasets for our model.

Importing the libraries

Importing the libraries

```
import numpy as nm
```

```
import matplotlib.pyplot as mtp
```

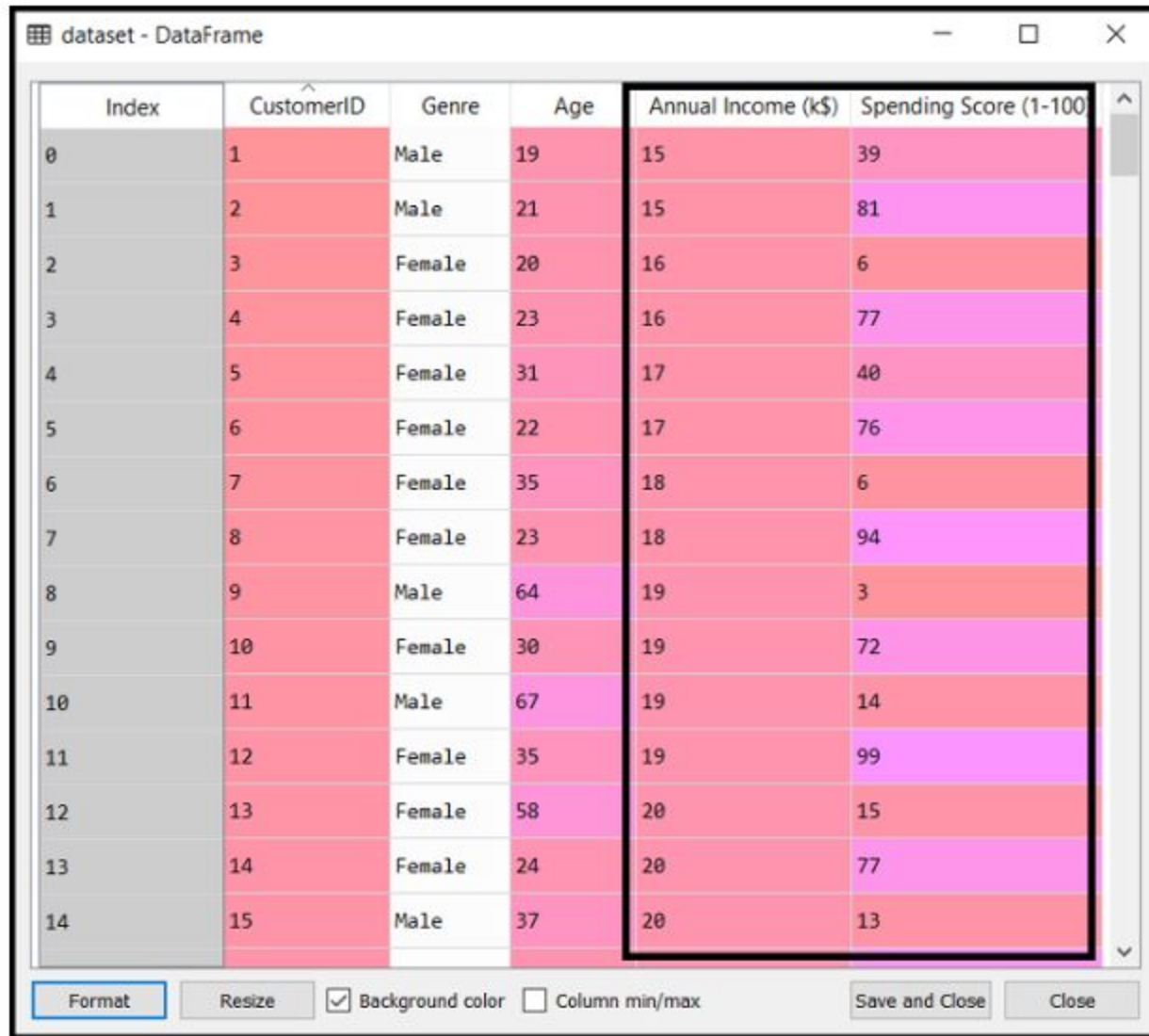
```
import pandas as pd
```

Importing the dataset

Importing the dataset

```
dataset = pd.read_csv('Mall_Customers_data.csv')
```

Python Implementation of Agglomerative Hierarchical Clustering



dataset - DataFrame

Index	CustomerID	Genre	Age	Annual Income (k\$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40
5	6	Female	22	17	76
6	7	Female	35	18	6
7	8	Female	23	18	94
8	9	Male	64	19	3
9	10	Female	30	19	72
10	11	Male	67	19	14
11	12	Female	35	19	99
12	13	Female	58	20	15
13	14	Female	24	20	77
14	15	Male	37	20	13

Format Resize ☒ Background color ☐ Column min/max Save and Close Close

Python Implementation of Agglomerative Hierarchical Clustering

Step-2: Finding the optimal number of clusters using the Dendrogram

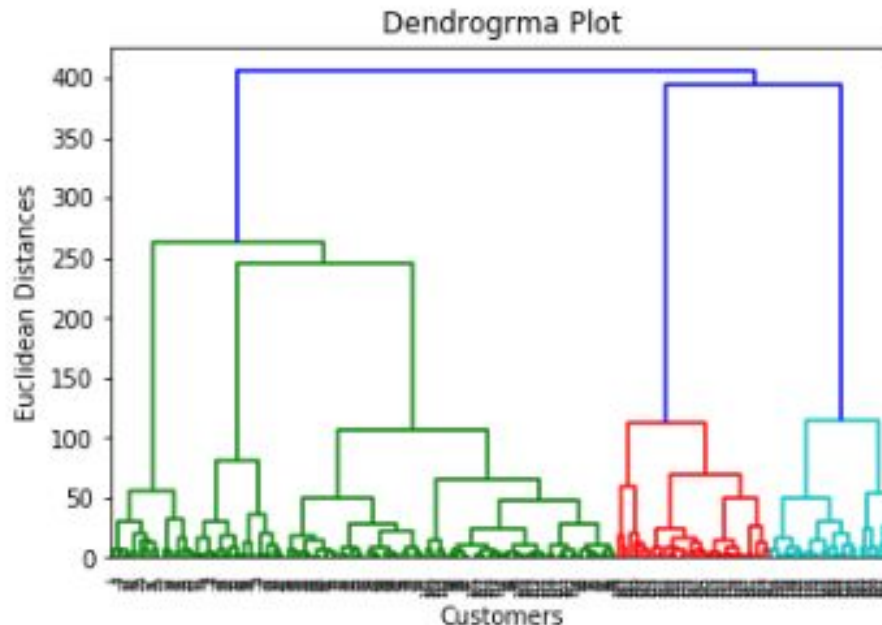
Now we will find the optimal number of clusters using the Dendrogram for our model. For this, we are going to use **scipy** library as it provides a function that will directly return the dendrogram for our code. Consider the below lines of code:

```
#Finding the optimal number of clusters using the dendrogram
import scipy.cluster.hierarchy as shc
dendro = shc.dendrogram(shc.linkage(x, method="ward"))
mtp.title("Dendrogrma Plot")
mtp.ylabel("Euclidean Distances")
mtp.xlabel("Customers")
mtp.show()
```

Python Implementation of Agglomerative Hierarchical Clustering

Output:

By executing the above lines of code, we will get the below output:



Python Implementation of Agglomerative Hierarchical Clustering

Step-3: Training the hierarchical clustering model

As we know the required optimal number of clusters, we can now train our model. The code is given below:

```
#training the hierarchical model on dataset
from sklearn.cluster import AgglomerativeClustering
hc= AgglomerativeClustering(n_clusters=5, affinity='euclidean', linkage='single')
y_pred= hc.fit_predict(x)
```

Python Implementation of Agglomerative Hierarchical Clustering

Step-4: Visualizing the clusters

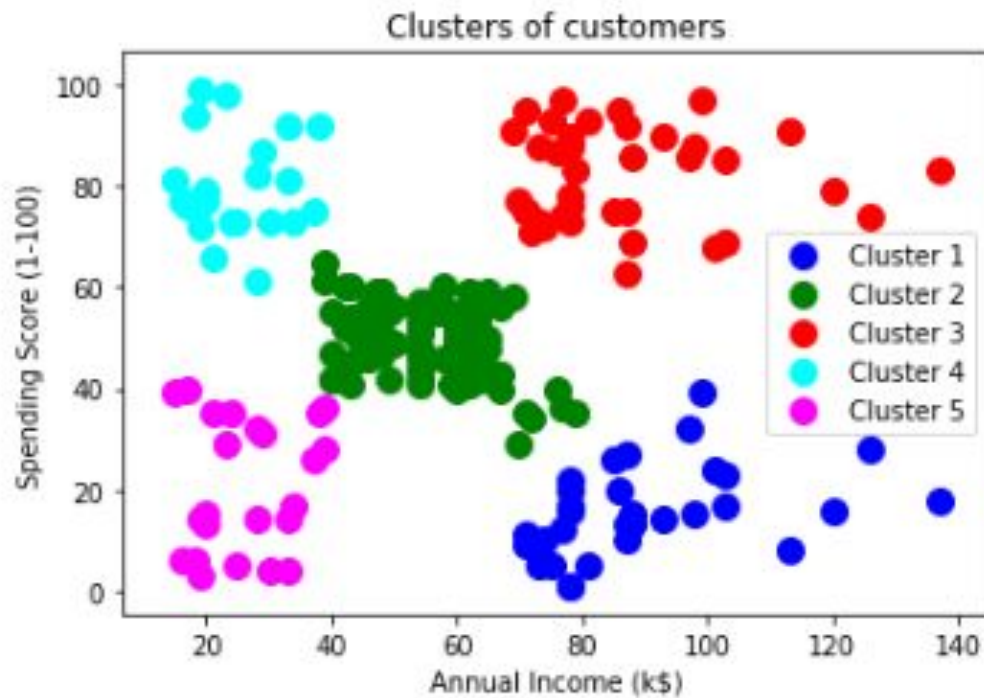
As we have trained our model successfully, now we can visualize the clusters corresponding to the dataset.

#visualizing the clusters

```
mtp.scatter(x[y_pred == 0, 0], x[y_pred == 0, 1], s = 100, c = 'blue', label = 'Cluster 1')
mtp.scatter(x[y_pred == 1, 0], x[y_pred == 1, 1], s = 100, c = 'green', label = 'Cluster 2')
mtp.scatter(x[y_pred == 2, 0], x[y_pred == 2, 1], s = 100, c = 'red', label = 'Cluster 3')
mtp.scatter(x[y_pred == 3, 0], x[y_pred == 3, 1], s = 100, c = 'cyan', label = 'Cluster 4')
mtp.scatter(x[y_pred == 4, 0], x[y_pred == 4, 1], s = 100, c = 'magenta', label = 'Cluster 5'
)
mtp.title('Clusters of customers')
mtp.xlabel('Annual Income (k$)')
mtp.ylabel('Spending Score (1-100)')
mtp.legend()
mtp.show()
```


Python Implementation of Agglomerative Hierarchical Clustering

Output: By executing the above lines of code, we will get the below output:



Unit 4 Divisive Clustering in Machine Learning



OUTLINE:

- ❖ Divisive Clustering in Machine Learning

Hierarchical Clustering in Machine Learning

Clustering Techniques:

- Hierarchical clustering is another unsupervised machine learning algorithm, which is used to group the unlabeled datasets into a cluster and also known as **hierarchical cluster analysis** or HCA.
- In this algorithm, we develop the hierarchy of clusters in the form of a tree, and this tree-shaped structure is known as the **dendrogram**.
- Sometimes the results of K-means clustering and hierarchical clustering may look similar, but they both differ depending on how they work.
- As there is no requirement to predetermine the number of clusters as we did in the K-Means algorithm.

Hierarchical Clustering in Machine Learning

The hierarchical clustering technique has two approaches:

- **Agglomerative:** Agglomerative is a **bottom-up** approach, in which the algorithm starts with taking all data points as single clusters and merging them until one cluster is left.
- **Divisive:** Divisive algorithm is the reverse of the agglomerative algorithm as it is a **top-down approach**.

Divisive Hierarchical clustering

- Divisive clustering, also known as **top-down clustering**, is a hierarchical clustering technique used in machine learning and data analysis.
- Unlike agglomerative clustering, which **starts with individual data points and merges them into clusters**, divisive clustering **begins with a single cluster containing all data points** and recursively divides it into smaller clusters.

Divisive Hierarchical clustering

- Start with a **single cluster** that includes all data points.
- Select a **clustering criterion or dissimilarity measure**, such as **Euclidean distance or cosine similarity**, to determine the dissimilarity between data points.
- Apply a divisive algorithm to divide the current cluster into two or more smaller clusters based on the selected criterion.
- Repeat the clustering process recursively on each of the newly created clusters until stopping criteria are met. The stopping criteria could be a predetermined number of clusters, a certain level of dissimilarity, or other criteria based on the problem at hand.
- Build a hierarchical tree-like structure called a dendrogram, which represents the nested clusters and their relationships. This dendrogram can be cut at different levels to obtain a hierarchy of clusters.

Divisive Hierarchical clustering

Initialization:

Begin with a single cluster that contains all data points.

Dissimilarity Measure:

Choose a dissimilarity measure (or similarity measure) to determine the dissimilarity between data points or clusters. Common measures include Euclidean distance, cosine similarity, or other domain-specific metrics.

Recursive Splitting:

Apply the following steps recursively until a stopping criterion is met:

- Compute the dissimilarity between all pairs of data points or clusters within the current cluster.
- Select the pair of data points or clusters with the highest dissimilarity.
- Split the selected cluster into two or more smaller clusters based on this dissimilarity.

Divisive Hierarchical clustering

Stopping Criterion:

Decide when to stop the recursive splitting. Common stopping criteria include:

- A predetermined number of clusters has been reached.
- A threshold level of dissimilarity is exceeded.
- The algorithm continues until there is only one data point in each cluster, but this may not always be practical.

Hierarchical Structure:

As the algorithm proceeds, it builds a hierarchical structure or dendrogram that represents how clusters were divided into sub clusters at each step.

Visualization:

Visualize the hierarchical structure to explore the clustering hierarchy and decide at which level of granularity to obtain the final clusters.

Unit 4 Measuring similarities and Dissimilarities



OUTLINE:

- ❖ Measuring similarities and Dissimilarities

Measuring similarities and Dissimilarities

Measuring dissimilarity (or similarity) between data points is a crucial aspect of many machine learning tasks, such as clustering, classification, and recommendation systems. Several distance or similarity metrics are commonly used, depending on the nature of the data and the specific problem at hand. Here are some commonly used distance and similarity metrics in machine learning:

1. **Euclidean Distance**
2. **Manhattan Distance**
3. **Cosine Similarity**

Measuring similarities and Dissimilarities

1. Euclidean Distance: This is the most common distance metric in Euclidean space. It measures the straight-line distance between two points. In 2D space, the Euclidean distance between points (x_1, y_1) and (x_2, y_2) is given by:

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

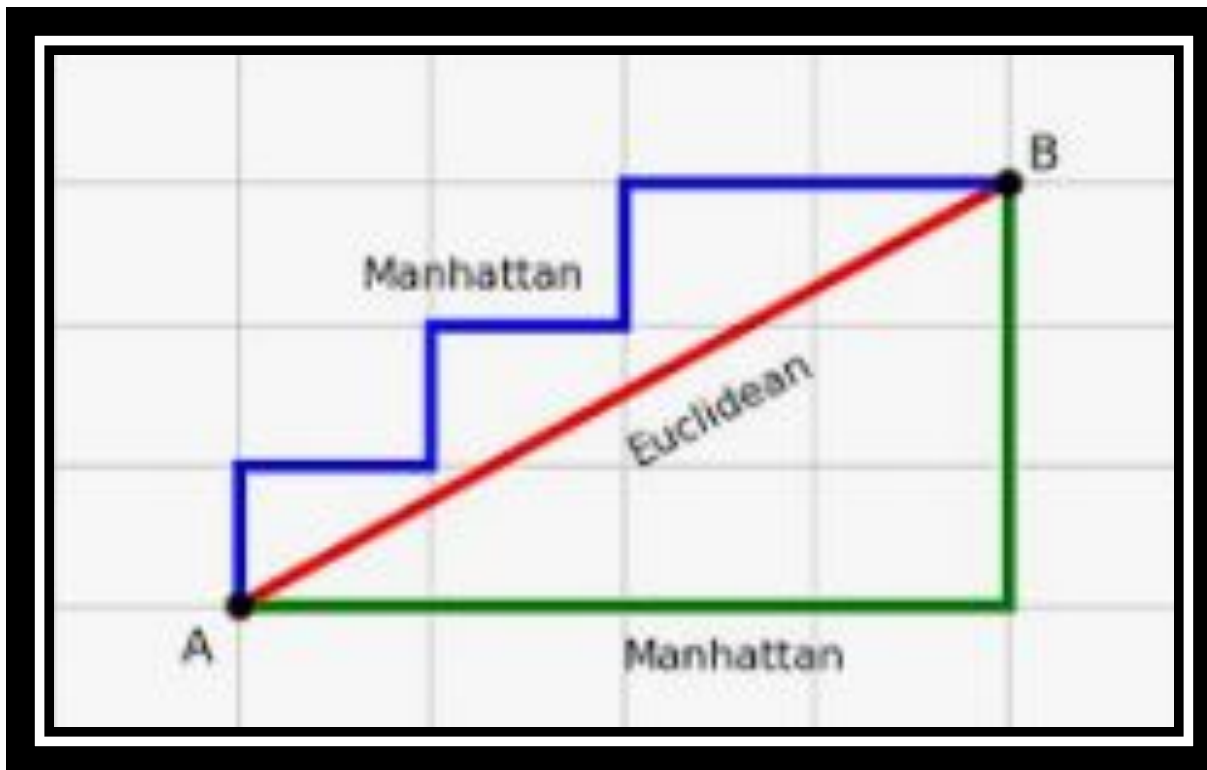
2. Manhattan Distance: Also known as L1 distance or city block distance, it measures the sum of absolute differences between the coordinates of two points. In 2D space, it is calculated as:

$$d = |x_1 - x_2| + |y_1 - y_2|$$

3. Cosine Similarity: This similarity metric is used for text data and other high-dimensional data. It measures the cosine of the angle between two vectors. It's often used in natural language processing for text document comparisons.

$$\text{similarity} = (A \cdot B) / (||A|| * ||B||)$$

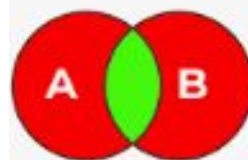
Measuring similarities and Dissimilarities



Measuring similarities and Dissimilarities

Jaccard Similarity: This similarity metric is used for sets. It measures the size of the intersection of two sets divided by the size of their union. It is often used in text analysis for measuring document similarity.

$$\text{similarity} = (|A \cap B|) / (|A \cup B|)$$



$$\text{Jaccard} = \frac{\text{Intersection (A, B)}}{\text{Union (A, B)}}$$

Hamming Distance: It measures the number of positions at which two strings of equal length differ. It's often used for comparing binary data or categorical data.

Correlation Distance: It measures the correlation between two variables. It is often used in feature selection and clustering.

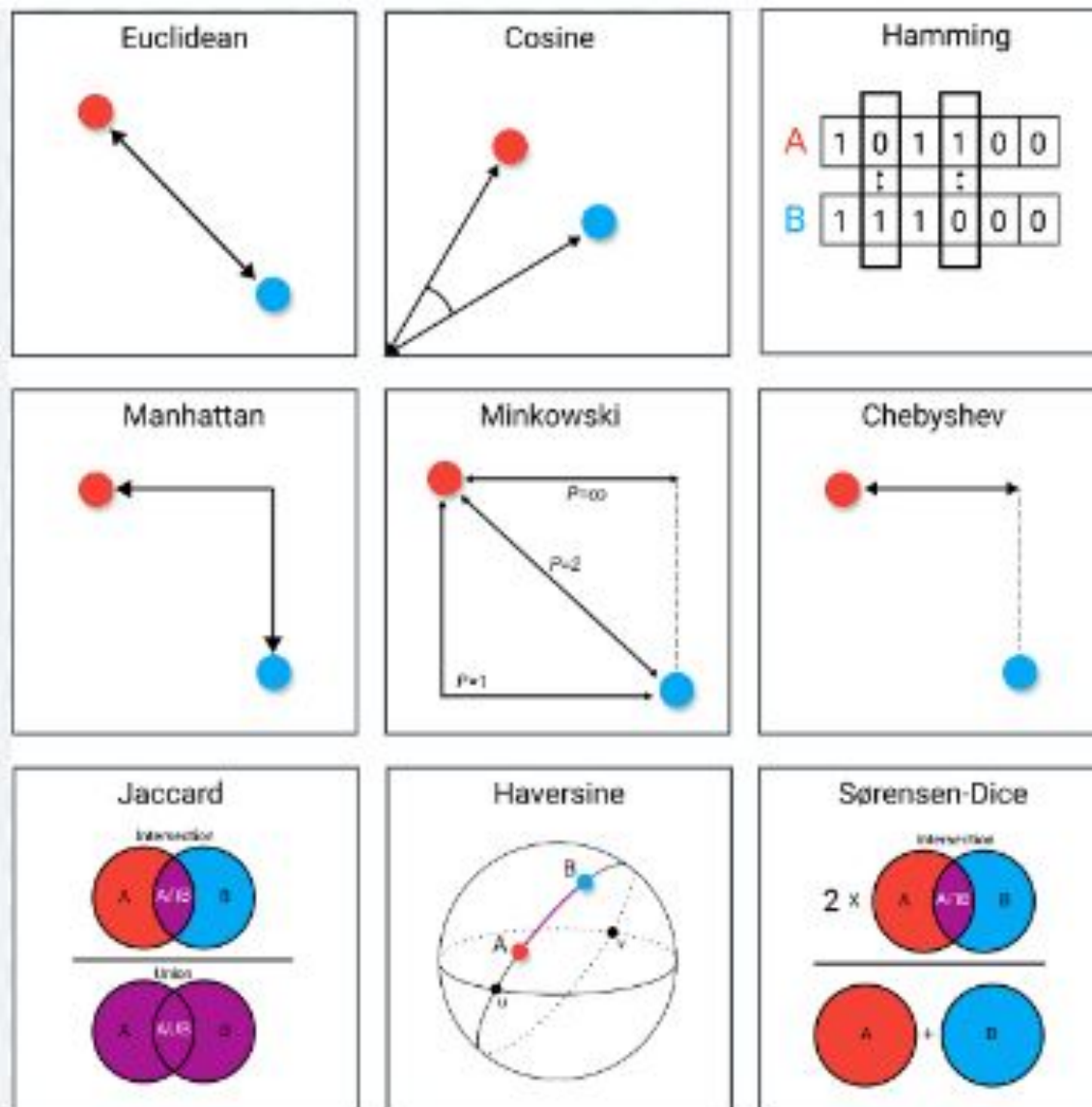
Measuring similarities and Dissimilarities

Minkowski Distance: It is a generalized distance metric that includes both Euclidean and Manhattan distances. The distance formula is:

$$d = (\sum (|x_1 - x_2|^p))^{1/p}$$

where 'p' is a parameter that determines the type of distance (p=1 for Manhattan, p=2 for Euclidean).

Measuring similarities and Dissimilarities





OUTLINE:

- ❖ Evaluating the Output Clustering Methods

- ## OUTLINE:
- ❖ Evaluating the Output Clustering Methods

Understanding the Basics of Clustering

Clustering Techniques:

- The fundamentals of clustering before getting into the evaluation of clustering models.
- The two types of clustering algorithms are hierarchical and non-hierarchical, respectively.
- Whereas **non-hierarchical clustering** methods begin with **random cluster assignments and improve them over iterations**, **hierarchical clustering** begins with **individual data points and organizes them into clusters repeatedly**.
- K-means, DBSCAN, and Gaussian mixture models are some popular non-hierarchical clustering algorithms, whereas agglomerative and divisive clustering are popular hierarchical techniques.

Evaluation Metrics for Clustering

Silhouette Score:

The Silhouette score is the measure of how similar a data point is to its own cluster as compared to other clusters. A higher Silhouette score value indicates that the data point is better matched to its own cluster and badly matched to other clusters. The best score value is 1 and -1 is the worst.

Silhouette coefficient for a sample is defined as:

Where **a** is the average intra-cluster distance, and **b** is the average nearest-cluster distance

```
from sklearn.metrics import silhouette_score  
print(silhouette_score(x, labels))
```

Evaluation Metrics for Clustering

Calinski-Harabasz Index:

A higher index value implies greater clustering performance. The Calinski-Harabasz index evaluates the ratio of between-cluster variation to within-cluster variance. **The CH Index (also known as Variance ratio criterion) is a measure of how similar an object is to its own cluster (cohesion) compared to other clusters (separation).**

```
from sklearn.metrics import calinski_harabasz_score  
print(calinski_harabasz_score(x, labels))
```

Davies-Bouldin index:

A lower Davies-Bouldin index suggests greater clustering performance since it gauges the average similarity between each cluster and its most comparable cluster

```
from sklearn.metrics import davies_bouldin_score  
print(davies_bouldin_score(x, labels))
```

Evaluation Metrics for Clustering

Rand Score:

The Rand score is the measure of similarity between two clusters, considering all pairs of samples and using the count of the sample pairs that are assigned in the same or different clusters in the original and predicted cluster.

The sklearn function for doing so is `adjusted_rand_score()` where the Rand score (RS) is adjusted to create Adjusted Rand Score (ARS), as:

It returns a value of 1.0 for identical clusterings and 0.0 for random labeling. To calculate the Rand score for the above kMeans clustering model, the python code is:

```
from sklearn.metrics import adjusted_rand_score  
print(adjusted_rand_score(y, labels))
```

Evaluation Metrics for Clustering

Jaccard Score:

Jaccard Score is defined as the size of the intersection of two labeled sets divided by the size of the union of those two sets. It is a similarity coefficient score that is used to compare a set of predicted labels to the corresponding set of labels in the original value of y .

To calculate the Jaccard score for the above kMeans clustering model, the python code is:

```
from sklearn.metrics import jaccard_score  
print(jaccard_score(y, y2, average=None))
```

Choosing the Right Evaluation Metric

- The nature and objectives of a clustering problem will dictate the most appropriate assessment measure to employ.
- If the goal of clustering is to group similar data points together, the Calinski-Harabasz index or the silhouette score can be beneficial.
- If the clustering results need to be compared to ground truth clustering, however, the Rand index or AMI would be more appropriate.
- So, it is important to consider the objectives and constraints of the clustering issue while selecting the evaluation metric.

Evaluating the Stability of Clustering Results

- Clustering has certain challenges since the parameters of the algorithm and the initial conditions may affect the results.
- It is essential to execute the clustering technique repeatedly using multiple random initializations or settings in order to judge the sustainability of the clustering findings.
- One can evaluate the stability of the clustering results using metrics such as the Jaccard index or the variance of information.

Visualizing the Clustering Results

- An understanding of the data's structure and patterns can be gained by visualizing the clustering findings.
- **Using scatter plots or heat maps**, where each data point is depicted as a point or a cell with a color-coded depending on its cluster assignment, is one approach to see the clustering findings.
- In order to project the high-dimensional data into a lower-dimensional space and show the clusters, dimensionality reduction techniques like principal component analysis (PCA) or t-SNE can be used.
- In addition, **visualization tools like dendrograms or silhouette plots** are frequently included in cluster analysis software packages allowing users to explore the clustering outcomes.

Unit 4 K Means Clustering Algorithm



OUTLINE:

- ❖ K Means Clustering Algorithm

K Means Clustering Algorithm

What is K-Means Algorithm?

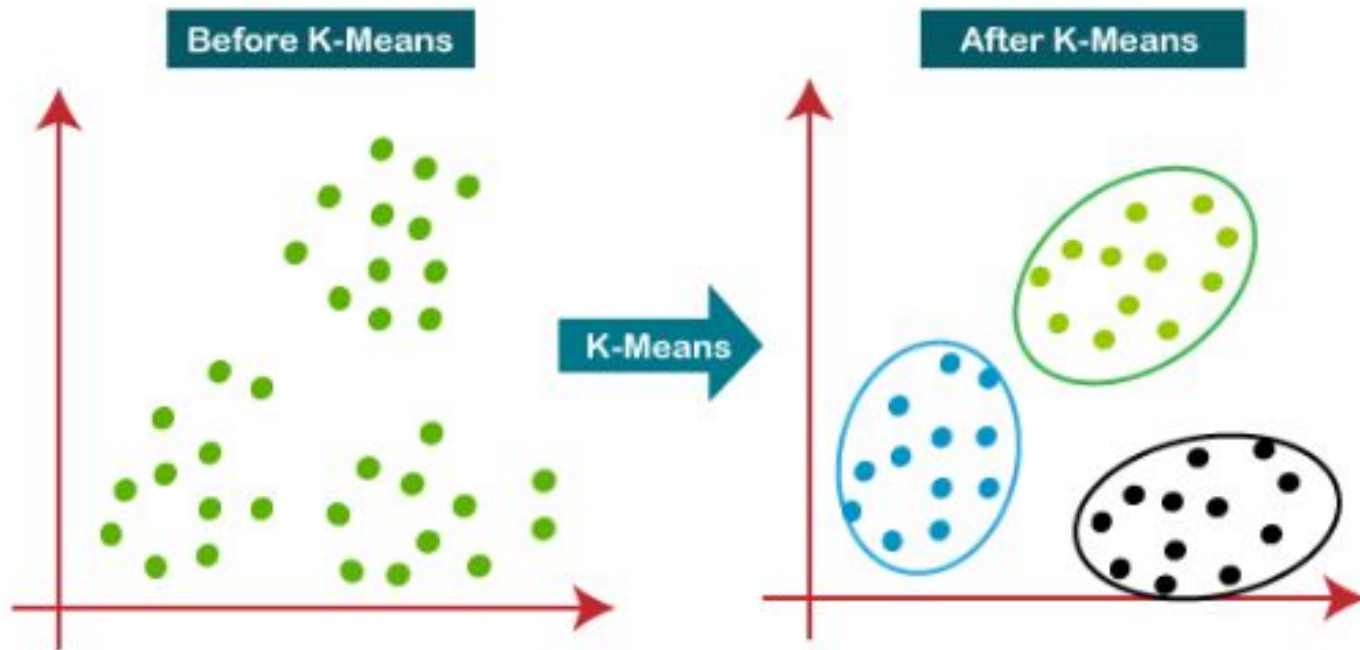
- K-Means Clustering is an [Unsupervised Learning algorithm](#), which groups the unlabeled dataset into different clusters.
- Here K defines the number of pre-defined clusters that need to be created in the process, as if $K=2$, there will be two clusters, and for $K=3$, there will be three clusters,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

The k-means [clustering](#) algorithm mainly performs two tasks:

1. Determines the best value for K center points or centroids by an iterative process.
2. Assigns each data point to its closest k-center. Those data points which are near to the particular k-center, create a cluster.

K Means Clustering Algorithm

The below diagram explains the working of the K-means Clustering Algorithm:



K Means Clustering Algorithm

How does the K-Means Algorithm Work?

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 each data point to their closest centroid, which will form the predefined K clusters.

Step-4: Calculate the variance and place a new centroid of each cluster.

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

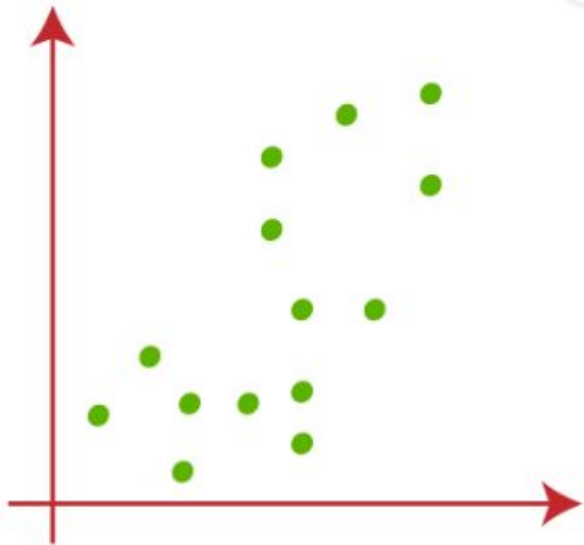
Step-6: If any reassignment occurs, then go to step-4 else go to FINISH.

Step-7: The model is ready.

Let's understand the above steps by considering the visual plots:

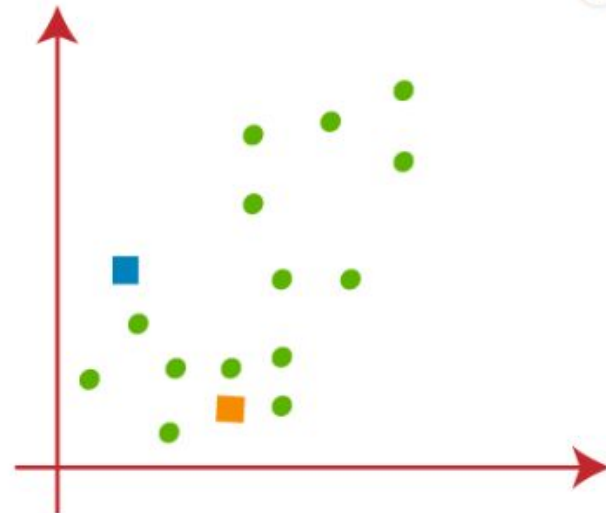
K Means Clustering Algorithm

Suppose we have two variables M1 and M2. The x-y axis scatter plot of these two variables is given below:



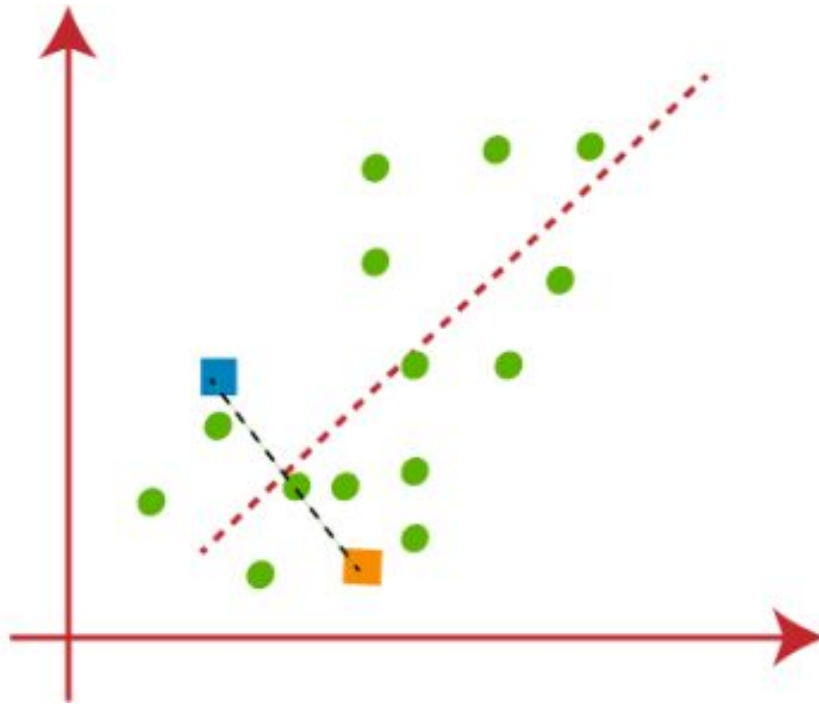
Let's take number k of clusters, i.e., $K=2$, to identify the dataset and to put them into different clusters. It means here we will try to group these datasets into two different clusters.

We need to choose some random k points or centroid to form the cluster. These points can be either the points from the dataset or any other point. So, here we are selecting the below two points as k points, which are not the part of our dataset. Consider the below image:



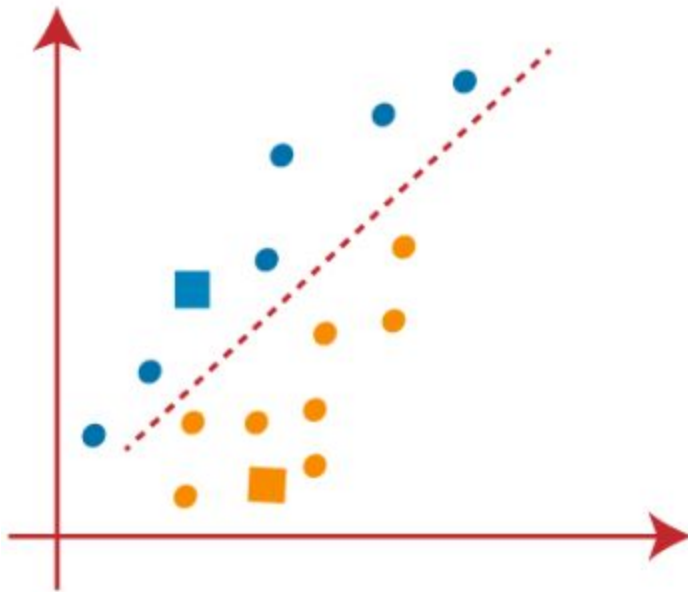
K Means Clustering Algorithm

Now we will assign each data point of the scatter plot to its closest K-point or centroid. We will compute it by applying some mathematics that we have studied to calculate the distance between two points. So, we will draw a median between both the centroids. Consider the below image:

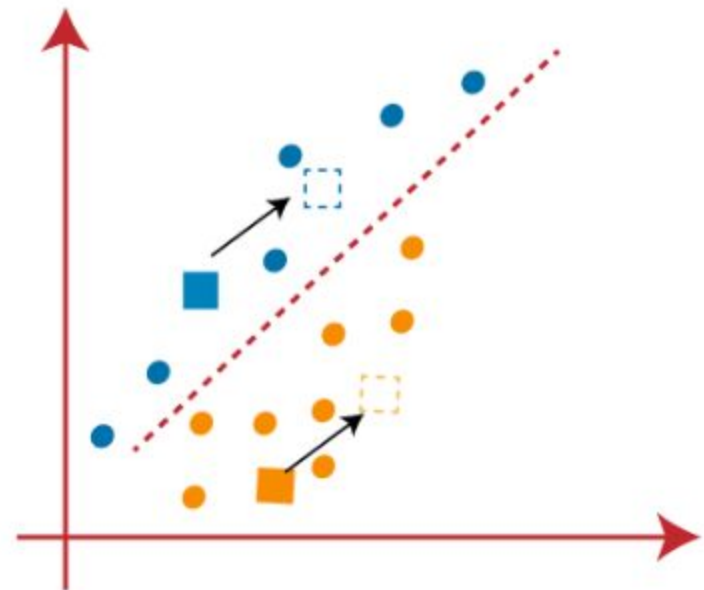


K Means Clustering Algorithm

From the above image, it is clear that points left side of the line is near to the K1 or blue centroid, and points to the right of the line are close to the yellow centroid. Let's color them as blue and yellow for clear visualization.



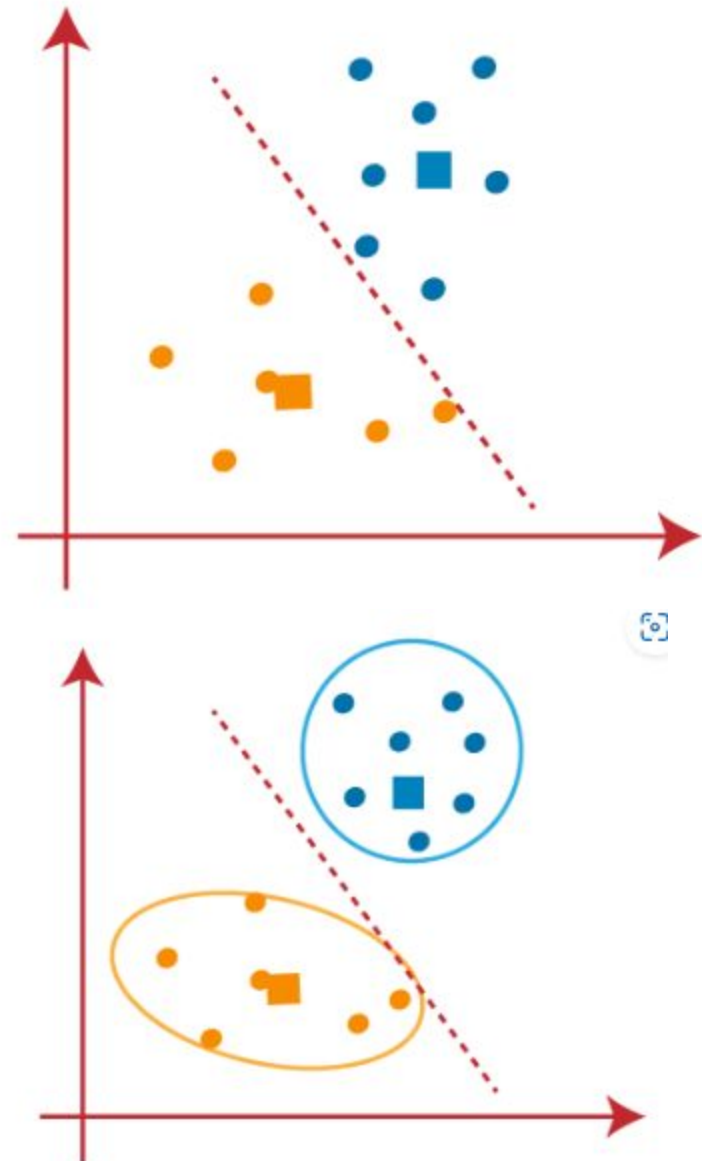
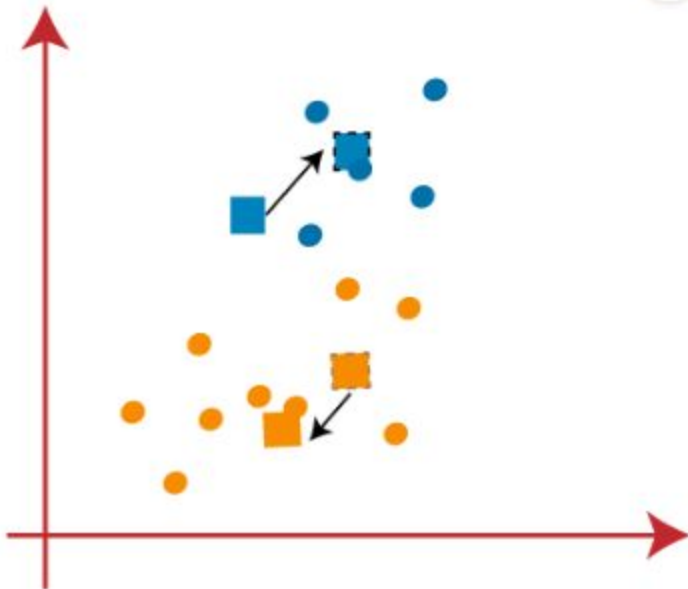
As we need to find the closest cluster, so we will repeat the process by choosing a **new centroid**. To choose the new centroids, we will compute the center of gravity of these centroids, and will find new centroids as below:



K Means Clustering Algorithm

As reassignment has taken place, so we will again go to the step-4, which is finding new centroids or K-points.

We will repeat the process by finding the center of gravity of centroids, so the new centroids will be as shown in the below image:



K Means Clustering Algorithm

How to choose the value of "K number of clusters" in K-means Clustering?

The Elbow method is one of the most popular ways to find the optimal number of clusters. This method uses the concept of WCSS value. **WCSS** stands for **Within Cluster Sum of Squares**, which defines the total variations within a cluster. The formula to calculate the value of WCSS (for 3 clusters) is given below:

$$WCSS = \sum_{P_i \text{ in Cluster1}} \text{distance}(P_i, C_1)^2 + \sum_{P_i \text{ in Cluster2}} \text{distance}(P_i, C_2)^2 + \sum_{P_i \text{ in Cluster3}} \text{distance}(P_i, C_3)^2$$

In the above formula of WCSS,

$\sum_{P_i \text{ in Cluster1}} \text{distance}(P_i, C_1)^2$: It is the sum of the square of **the distances between each data point and its centroid within a cluster1** and the same for the other two terms.

To measure the distance between data points and centroid, **we can use any method such as Euclidean distance or Manhattan distance.**

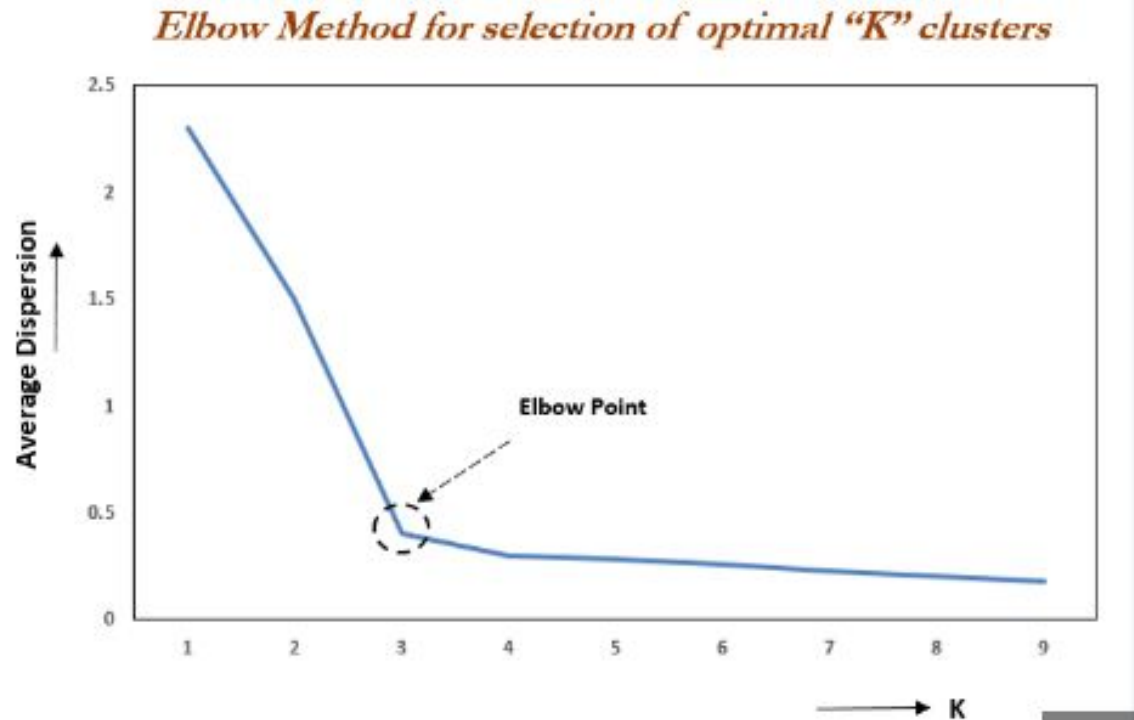
To find the optimal value of clusters, the elbow method follows the below steps:
It executes the K-means clustering on a given dataset for different K values (ranges from 1-10).

For each value of K, calculates the WCSS value.

Plots a curve between calculated WCSS values and the number of clusters K.

The sharp point of bend or a point of the plot looks like an arm, then that point is considered as the best value of K.

K Means Clustering Algorithm



K Means Clustering Algorithm

Python Implementation of K-means Clustering Algorithm

In the above section, we have discussed the K-means algorithm, now let's see how it can be implemented using [Python](#).

Before implementation, let's understand what type of problem we will solve here. So, we have a dataset of **Mall_Customers**, which is the data of customers who visit the mall and spend there.

In the given dataset, we have **Customer_Id**, **Gender**, **Age**, **Annual Income (\$)**, and **Spending Score** (which is the calculated value of how much a customer has spent in the mall, the more the value, the more he has spent). From this dataset, we need to calculate some patterns, as it is an unsupervised method, so we don't know what to calculate exactly.

K Means Clustering Algorithm

- **Importing Libraries**

As we did in previous topics, firstly, we will import the libraries for our model, which is part of data pre-processing. The code is given below:

```
# importing libraries
import numpy as nm
import matplotlib.pyplot as mtp
import pandas as pd
```

- **Importing the Dataset:**

Next, we will import the dataset that we need to use. So here, we are using the Mall_Customer_data.csv dataset. It can be imported using the below code:

```
# Importing the dataset
dataset = pd.read_csv('Mall_Customers_data.csv')
```

K Means Clustering Algorithm

dataset - DataFrame

Index	CustomerID	Genre	Age	Annual Income (k\$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40
5	6	Female	22	17	76
6	7	Female	35	18	6
7	8	Female	23	18	94
8	9	Male	64	19	3
9	10	Female	30	19	72
10	11	Male	67	19	14
11	12	Female	35	19	99
12	13	Female	58	20	15
13	14	Female	24	20	77
14	15	Male	37	20	13
15	16	Male	22	20	79

Format Resize ☒ Background color ☒ Column min/max Save and Close Close

K Means Clustering Algorithm

Finding the optimal number of clusters using the elbow method:

#finding optimal number of clusters using the elbow method

from sklearn.cluster **import** KMeans

wcss_list= [] #Initializing the list **for** the values of WCSS

#Using **for** loop **for** iterations from 1 to 10.

for i in range(1, 11):

 kmeans = KMeans(n_clusters=i, init='k-means++', random_state= 1)

 kmeans.fit(x)

 wcss_list.append(kmeans.inertia_)

mtp.plot(range(1, 11), wcss_list)

mtp.title('The Elbow Method Graph')

mtp.xlabel('Number of clusters(k)')

mtp.ylabel('wcss_list')

mtp.show()

K Means Clustering Algorithm

Training the K-means algorithm on the training dataset

As we have got the number of clusters, so we can now train the model on the dataset.

To train the model, we will use the same two lines of code as we have used in the above section, but here instead of using `i`, we will use `5`, as we know there are 5 clusters that need to be formed. The code is given below:

```
#training the K-means model on a dataset
```

```
kmeans = KMeans(n_clusters=5, init='k-means++', random_state= 42)
```

```
y_predict= kmeans.fit_predict(x)
```


K Means Clustering Algorithm

Step-4: Visualizing the Clusters

The last step is to visualize the clusters. As we have 5 clusters for our model, so we will visualize each cluster one by one.

To visualize the clusters will use scatter plot using `mtp.scatter()` function of `matplotlib`.

`#visualizing the clusters`

```
mtp.scatter(x[y_predict == 0, 0], x[y_predict == 0, 1], s = 100, c = 'blue', label = 'Cluster 1') #f  
or first cluster
```

```
mtp.scatter(x[y_predict == 1, 0], x[y_predict == 1, 1], s = 100, c = 'green', label = 'Cluster 2') #  
for second cluster
```

```
mtp.scatter(x[y_predict == 2, 0], x[y_predict == 2, 1], s = 100, c = 'red', label = 'Cluster 3') #for  
third cluster
```

```
mtp.scatter(x[y_predict == 3, 0], x[y_predict == 3, 1], s = 100, c = 'cyan', label = 'Cluster 4') #f  
or fourth cluster
```

```
mtp.scatter(x[y_predict == 4, 0], x[y_predict == 4, 1], s = 100, c = 'magenta', label = 'Cluster 5  
)' #for fifth cluster
```

```
mtp.scatter(kmeans.cluster_centers_[0], kmeans.cluster_centers_[1], s = 300, c = 'yellow', label = 'Centroid')
```

```
mtp.title('Clusters of customers')
```

```
mtp.xlabel('Annual Income (k$)')
```

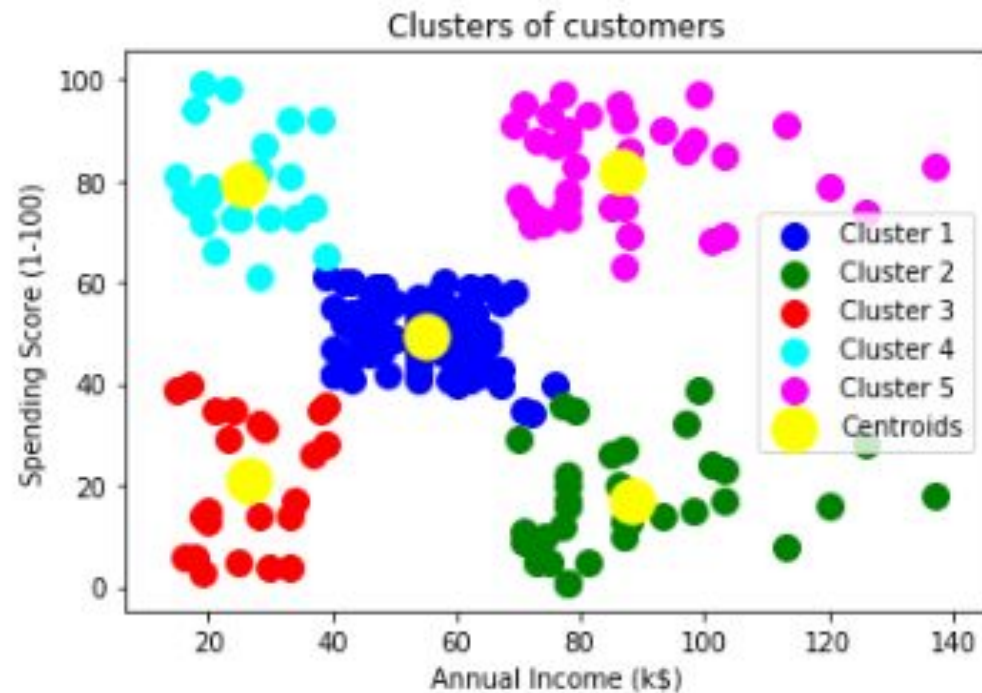
```
mtp.ylabel('Spending Score (1-100)')
```

```
mtp.legend()
```

```
mtp.show()
```

K Means Clustering Algorithm

Output:



Unit 4 K-Meloids clustering Algorithm



OUTLINE:

- ❖ K-Meloids clustering Algorithm

K-Meloids clustering Algorithm

In the K-Means algorithm, given the value of k and unlabelled data:

- Choose k number of random points (Data point from the data set or some other points). These points are also called "**Centroids**" or "**Means**".
- Assign all the data points in the data set to the closest centroid by applying any distance formula like **Euclidian distance**, **Manhattan distance**, etc.
- Now, choose new centroids by calculating the mean of all the data points in the clusters and goto step 2
- Continue step 3 until no data point changes classification between two iterations.
- The problem with the K-Means algorithm is that the algorithm needs to handle outlier data. An outlier is a point different from the rest of the points. All the outlier data points show up in a different cluster and will attract other clusters to merge with it. Outlier data increases the mean of a cluster by up to 10 units. Hence, **K-Means clustering is highly affected by outlier data.**

K-Meloids clustering Algorithm

K-Medoids:

- Medoid:** A Medoid is a point in the cluster from which the sum of distances to other data points is minimal.
- (or)**
- A Medoid is a point in the cluster from which dissimilarities with all the other points in the clusters are minimal.
- Instead of centroids as reference points in K-Means algorithms, the K-Medoids algorithm takes a Medoid as a reference point.

There are three types of algorithms for K-Medoids Clustering:

PAM (Partitioning Around Clustering)

CLARA (Clustering Large Applications)

CLARANS (Randomized Clustering Large Applications)

K-Meloids clustering Algorithm

PAM is the most powerful algorithm of the three algorithms but has the disadvantage of time complexity. The following K-Medoids are performed using PAM. In the further parts, we'll see what CLARA and CLARANS are.

Algorithm:

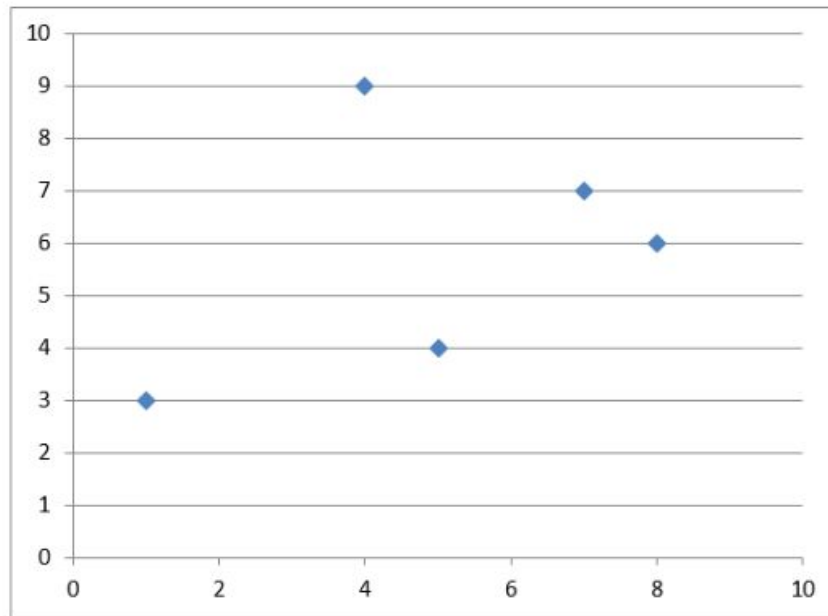
Given the value of k and unlabelled data:

- Choose k number of random points from the data and assign these k points to k number of clusters. These are the initial medoids.
- For all the remaining data points, calculate the distance from each medoid and assign it to the cluster with the nearest medoid.
- Calculate the total cost (Sum of all the distances from all the data points to the medoids)
- Select a random point as the new medoid and swap it with the previous medoid. Repeat 2 and 3 steps.
- 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.
- If the total cost of the new medoid is greater than the cost of the previous medoid, undo the swap and repeat step 4.
- The Repetitions have to continue until no change is encountered with new medoids to classify data points.

K-Meloids clustering Algorithm

Data set:

	x	y
0	5	4
1	7	7
2	1	3
3	8	6
4	4	9



K-Medoids clustering Algorithm

If k is given as 2, we need to break down the data points into 2 clusters.

1. **Initial medoids: M1(1, 3) and M2(4, 9)**

2. Calculation of distances

Manhattan Distance: $|x_1 - x_2| + |y_1 - y_2|$

	x	y	From M1(1, 3)	From M2(4, 9)
0	5	4	5	6
1	7	7	10	5
2	1	3	-	-
3	8	6	10	7
4	4	9	-	-

Cluster 1: 0

K-Meloids clustering Algorithm

Cluster 2: 1, 3

1. Calculation of total cost:

$$(5) + (5 + 7) = 17$$

2. Random medoid: (5, 4)

M1(5, 4) and M2(4, 9):

	x	y	From M1(5, 4)	From M2(4, 9)
0	5	4	-	-
1	7	7	5	5
2	1	3	5	9
3	8	6	5	7
4	4	9	-	-

Cluster 1: 2, 3

K-Meloids clustering Algorithm

Cluster 2: 1

1. Calculation of total cost:
 $(5 + 5) + 5 = 15$
Less than the previous cost
New medoid: (5, 4).
2. Random medoid: (7, 7)

M1(5, 4) and M2(7, 7)

	x	y	From M1(5, 4)	From M2(7, 7)
0	5	4	-	-
1	7	7	-	-
2	1	3	5	10
3	8	6	5	2
4	4	9	6	5

Cluster 1: 2

K-Meloids clustering Algorithm

Cluster 2: 3, 4

1. Calculation of total cost:
 $(5) + (2 + 5) = 12$
Less than the previous cost
New medoid: (7, 7).
2. Random medoid: (8, 6)

M1(7, 7) and M2(8, 6)

	x	y	From M1(7, 7)	From M2(8, 6)
0	5	4	5	5
1	7	7	-	-
2	1	3	10	10
3	8	6	-	-
4	4	9	5	7

Cluster 1: 4

K-Meloids clustering Algorithm

Cluster 2: 0, 2

1. Calculation of total cost:

$$(5) + (5 + 10) = 20$$

Greater than the previous cost

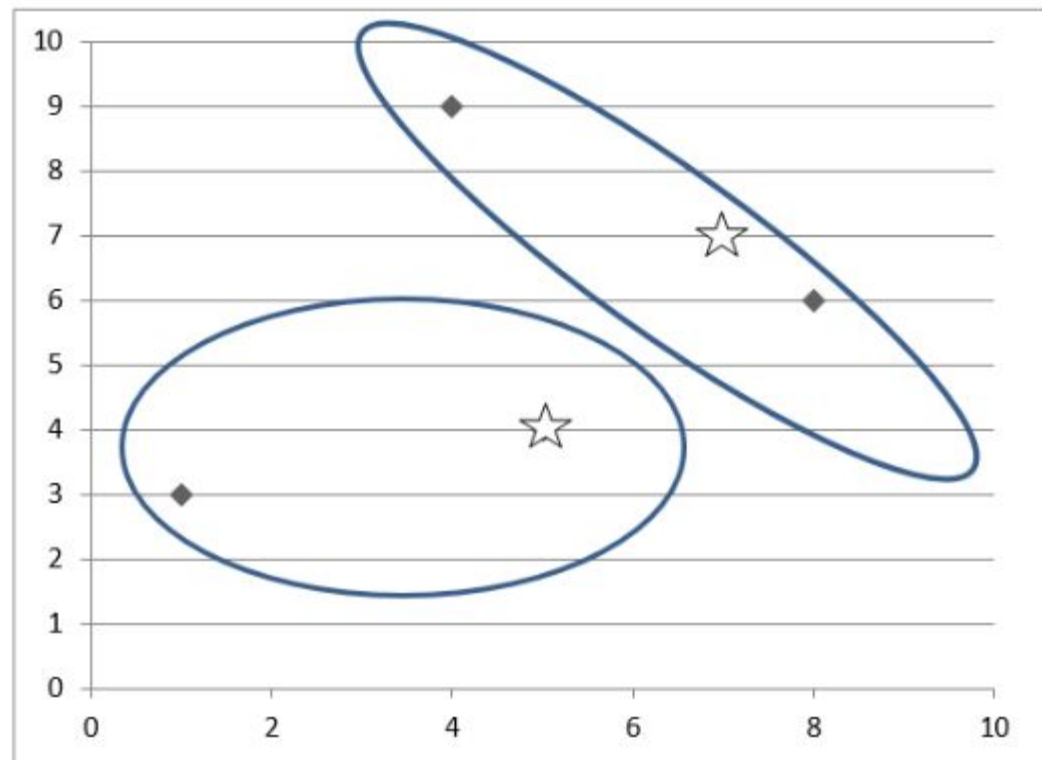
UNDO

Hence, the final medoids: **M1(5, 4) and M2(7, 7)**

Cluster 1: 2

Cluster 2: 3, 4

Total cost: 12



K-Medoids clustering Algorithm

CLARA:

It is an extension to PAM to support Medoid clustering for large data sets. This algorithm selects **data samples from the data set, applies Pam on each sample, and outputs the best Clustering out of these samples**. This is more effective than PAM. We should ensure that the selected samples aren't biased as they affect the Clustering of the whole data.

CLARANS:

This algorithm selects a sample of neighbors to examine instead of selecting samples from the data set. In every step, it examines the neighbors of every node. The time complexity of this algorithm is $O(n^2)$, and this is the best and most efficient Medoids algorithm of all.

K-Meloids clustering Algorithm

K-Means	K-Medoids
Both methods are types of Partition Clustering.	
Unsupervised iterative algorithms	
Have to deal with unlabelled data	
Both algorithms group n objects into k clusters based on similar traits where k is pre-defined.	
Inputs: Unlabelled data and the value of k	
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



OUTLINE:

- ❖ Image Segmentation using K means Clustering

- ## OUTLINE:
- ❖ Image Segmentation using K means Clustering

Image Segmentation using K Means Clustering

- **Image Segmentation:** In computer vision, image segmentation is the process of partitioning an image into multiple segments.
- The goal of segmenting an image is to change the representation of an image into something that is more meaningful and easier to analyze. It is usually used for locating objects and creating boundaries.
- It is not a great idea to process an entire image because many parts in an image may not contain any useful information. Therefore, by segmenting the image, we can make use of only the important segments for processing.
- An image is basically a set of given pixels. In image segmentation, pixels which have similar attributes are grouped together. Image segmentation creates a pixel-wise mask for objects in an image which gives us a more comprehensive and granular understanding of the object.

Image Segmentation using K Means Clustering

Autonomous vehicles – During the design of the perception system of self-driving vehicles semantic segmentation is used to separate objects from the surrounding in the vicinity of the car.



Image Segmentation using K Means Clustering

Imaging in Medical Field – Clustering is used to separate and identify types of tissues in cancerous cell images.

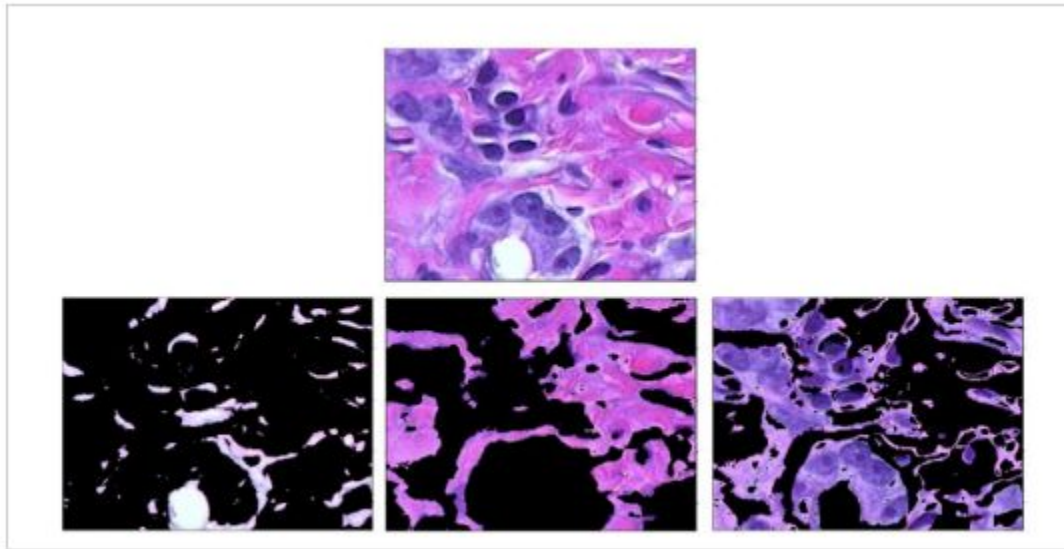


Image Segmentation using K Means Clustering

K-Means clustering for Image Segmentation:

- K-Means is an agglomerative type of clustering where we don't have the labels of pixels/data points beforehand. The number of clusters or groups formed is taken as K. These clusters are derived based on some of the similar or common characteristics between the pixels or data points.
- The steps involved in K-Means clustering are
 - Select a particular value of K
 - A feature is taken in each pixel like RGB value, etc.
 - Similar pixels are grouped by using distances like Euclidean distance.
 - K-Means is used with the center of the cluster.
 - Generally, a threshold is defined within which if the calculated size falls it is grouped into a single cluster.

Image Segmentation using K Means Clustering

```
# KMEANS IMAGE SEGMENTATION
import numpy as np
import cv2
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
%matplotlib inline

image_1 = cv2.imread("/content/image_1.jpg", cv2.IMREAD_UNCHAN
image_2 = cv2.imread("/content/image_2.jpg", cv2.IMREAD_UNCHAN
vector_1 = image_1.reshape((-1,3))
vector_2 = image_2.reshape((-1,3))
kmeans_1 = KMeans(n_clusters=5, random_state = 0, n_init=5).fi
c = np.uint8(kmeans_1.cluster_centers_)
seg_data= c[kmeans_1.labels_.flatten()]
seg_image = seg_data.reshape((image_1.shape))
plt.imshow(seg_image)
plt.pause(1)

kmeans_2 = KMeans(n_clusters=5, random_state = 0, n_init=5).fi
c = np.uint8(kmeans_2.cluster_centers_)
seg_data= c[kmeans_2.labels_.flatten()]
seg_image = seg_data.reshape((image_2.shape))
plt.imshow(seg_image)
plt.pause(1)
```



OUTLINE:

- ## ❖ Bi-Clustering & Multi view Clustering in Machine Learning

Bi-Clustering in Machine Learning

- **Biclustering** is a data analysis technique used in machine learning and data mining to find patterns in data where both rows and columns exhibit some form of similarity or clustering.
- Unlike traditional clustering, which groups data points (rows) based on their similarity, biclustering seeks to simultaneously group both rows and columns in a way that reveals hidden structures in the data.

Bi-Clustering in Machine Learning

some key points about biclustering in machine learning:

Simultaneous Clustering: Biclustering aims to identify submatrices within a larger data matrix where the elements share some common characteristics. These submatrices represent subsets of rows and columns that are more closely related to each other than to the rest of the data.

Applications: Biclustering is used in various domains, such as gene expression analysis, text mining, and recommendation systems. For example, in gene expression analysis, biclustering can help identify groups of genes that are co-regulated across different experimental conditions.

Types of Biclustering:

Binary Biclustering: In binary biclustering, the elements in the data matrix are binary (0 or 1), and the goal is to find submatrices with similar binary patterns.

Real-Valued Biclustering: This is more general and is used with real-valued data. It aims to discover submatrices with similar numeric values.

Bi-Clustering in Machine Learning

Algorithms: Several algorithms have been developed for biclustering, such as Spectral Biclustering, Plaid Model, and Bimax. These algorithms use different techniques to identify biclusters in the data.

Evaluation: Evaluating the quality of biclustering results can be challenging. Common measures include the mean squared residue, coherence, and relevance, but the choice of evaluation metric may depend on the specific problem and data.

Challenges: Biclustering is a complex problem with many possible solutions. The choice of the appropriate algorithm and parameter settings can greatly affect the results. It can also be computationally intensive, especially for large datasets.

Interpretability: One of the benefits of biclustering is that it provides interpretable results. Biclusters represent meaningful patterns in the data, making it easier to understand the relationships between rows and columns.

Multi view Clustering in Machine Learning

- **Multiview clustering** is a machine learning technique used to cluster data that comes from multiple distinct sources or views.
- In many real-world scenarios, data can be represented in multiple ways, and each representation (view) may capture different aspects or features of the underlying information.
- Multiview clustering seeks to leverage this diversity in data representations to improve the quality of clustering results. Here's an overview of multiview clustering:

Multi view Clustering in Machine Learning

Key Concepts:

Multiview Data: In multiview clustering, the data is available in multiple views, which can be different representations or modalities of the same underlying information. For example, in a medical application, one view might contain text-based patient records, while another view could have medical images of the same patients.

View Agreement: The core idea behind multiview clustering is to find clusters that agree across multiple views. Clusters are considered more reliable when they are consistent across various data representations. In other words, the different views should complement each other, and clustering results should be consistent.

Integration Methods: Multiview clustering methods use different techniques to combine information from multiple views. Some common integration methods include co-regularization, consensus clustering, and ensemble clustering. These methods aim to exploit the relationships between clusters in different views.

Challenges: Multiview clustering introduces unique challenges. One significant challenge is view selection, i.e., determining which views are relevant and how they should be weighted during the clustering process. Another challenge is handling view-specific noise and inconsistencies.

Multi view Clustering in Machine Learning

Approaches to Multiview Clustering:

Co-Regularization: This approach encourages the clustering solutions in each view to be consistent with one another. Algorithms based on co-regularization seek a shared cluster structure across multiple views while considering the uniqueness of each view.

Consensus Clustering: In consensus clustering, multiple clustering results from each view are combined into a single consensus clustering. This is often done by taking the average, weighted average, or other measures of agreement between the views.

Ensemble Clustering: Ensemble methods treat each view as a base clustering algorithm and combine their results using techniques such as majority voting, weighted voting, or hierarchical clustering.

Subspace Clustering: Subspace clustering focuses on finding clusters within each view and then integrating these view-specific clusters to form a consensus clustering. Subspace clustering methods are particularly useful when views are high-dimensional and have substructures.