Discussion 3: Machine Learning

Problem Statement - Reflection Report of Guest Lecture.

1. Aspects of Unsupervised Machine Learning.
2. Some Hands-on with Python(Real World examples of Unsupervised Learning)

Submitted by: Atharva Puranik & Vidushi Yadav

Speaker - Sandeep Bidwai
Institution - Army Institute of Technology, Pune, Maharashtra
Department - Electronics and Telecommunication.
Title of the session - Aspects of Unsupervised Machine Learning
Date of the session - Day- 06th Jan 2023 (10.00 am to 12:00 pm)

Aspects of Unsupervised Machine Learning. | A session by Dr. Sandeep Bidwai

CLUSTERING IN UNSUPERVISED LEARNING, K-MEANS, C- MEANS, HIERARCHICAL CLUSTERING, AND INSTANCE-BASED LEARNING (HANDS-ON WITH PYTHON)

"While designing the software we must think from a users perspective"

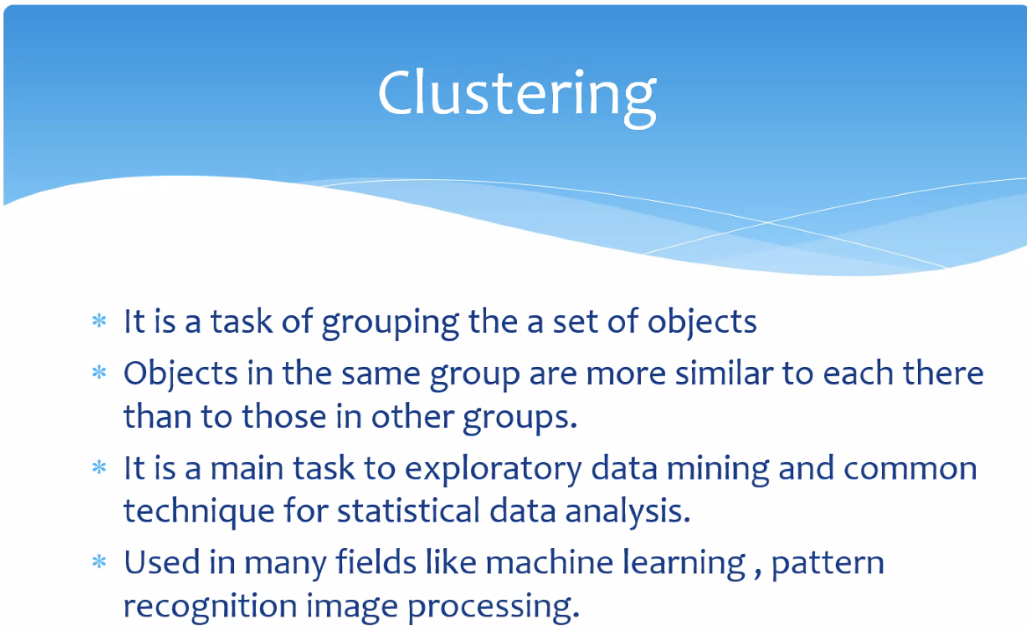
It's necessary to understand, preplan, and process the data.

Clustering

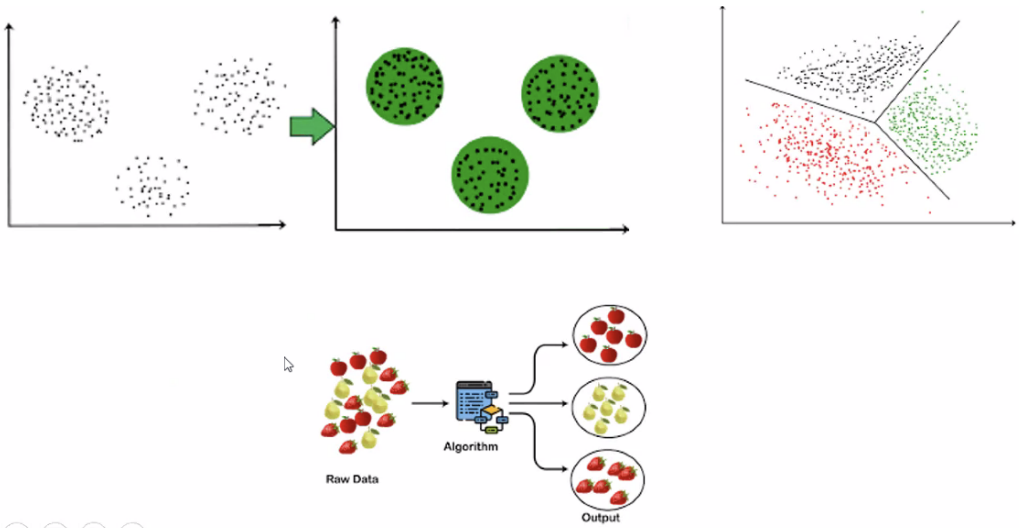
Aim : To make the available raw data into meaningful clusters.

One of the methods of grouping sets of objects into different domains. Different aspects are being considered for grouping, it is based on certain parameters.

In a large amount of data, we make groups, by having the common properties.



Output : to have differentiation in terms of similarities.



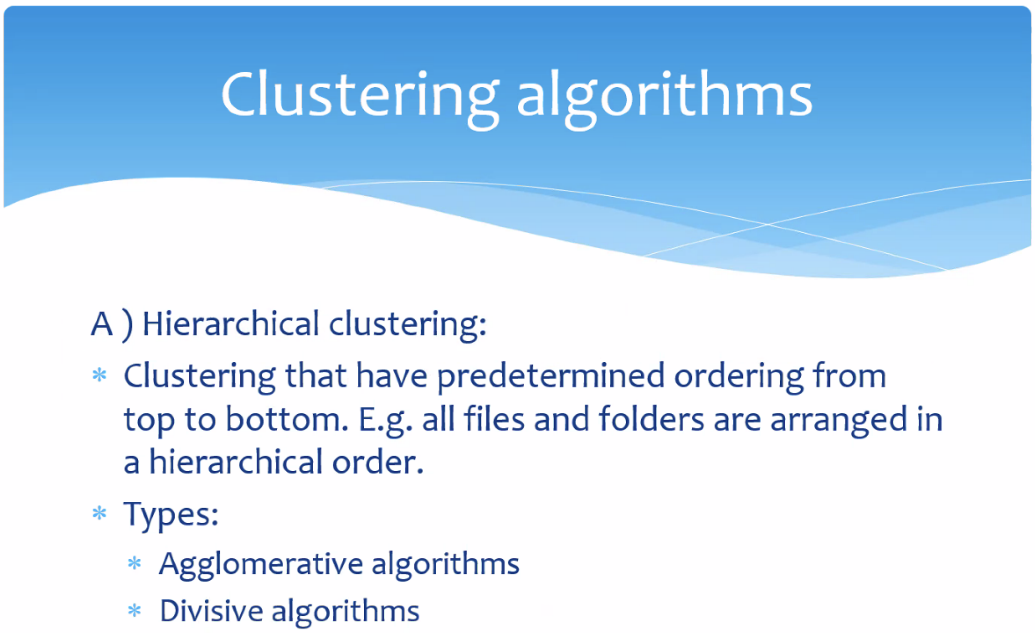
Clustering Algorithm

Tree based structure and is based on priority, it has predefined order.

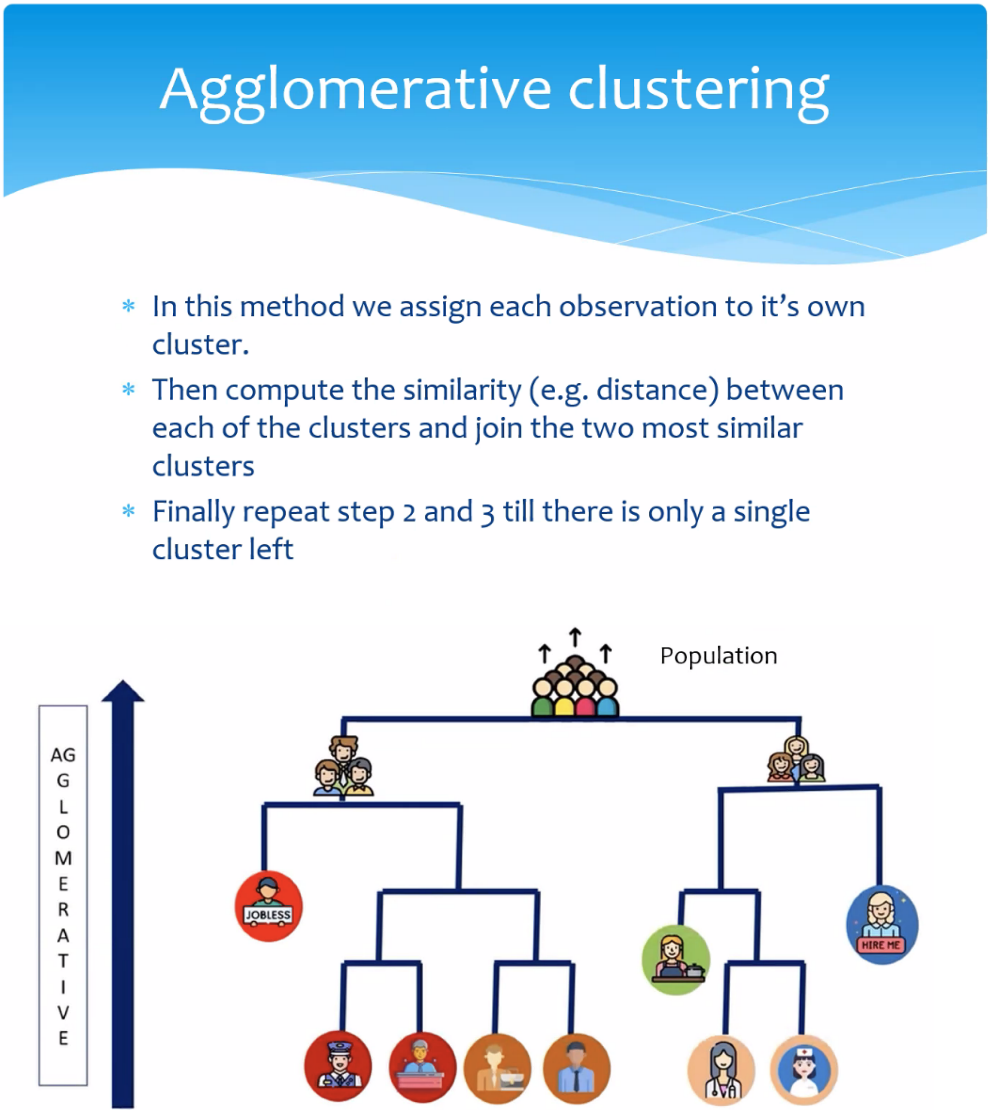
A) Hierarchical C: It has predetermined ordering from top to bottom

Ex: all files and folders are arranged in a hierarchical order.

Types:



1) **Agglomerative Clustering**
Working :We assign each observation to its own cluster, it's necessary which type of cluster we have.

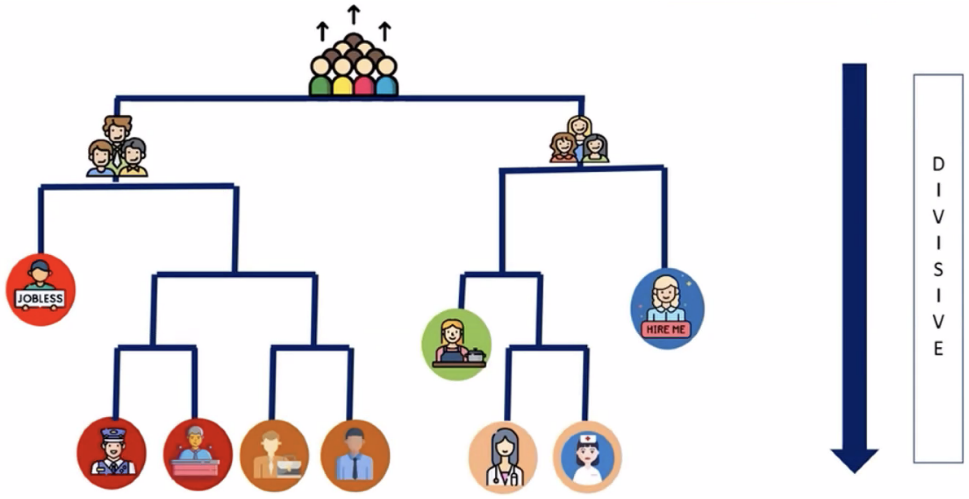


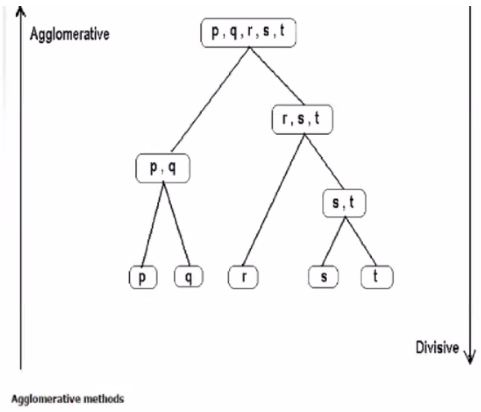
2) **Divisive:** We assign all the observations to a single cluster, and then partition the cluster to 2 least similar clusters.

- Then we proceed recursively on each cluster until there is one cluster for each observation,
- Example : K means clustering.

Divisive Algorithm

* In this method we assign all of the observations to a single cluster and then partition the cluster to two least similar clusters.
* Finally we proceed recursively on each cluster until there is one cluster for each observation.

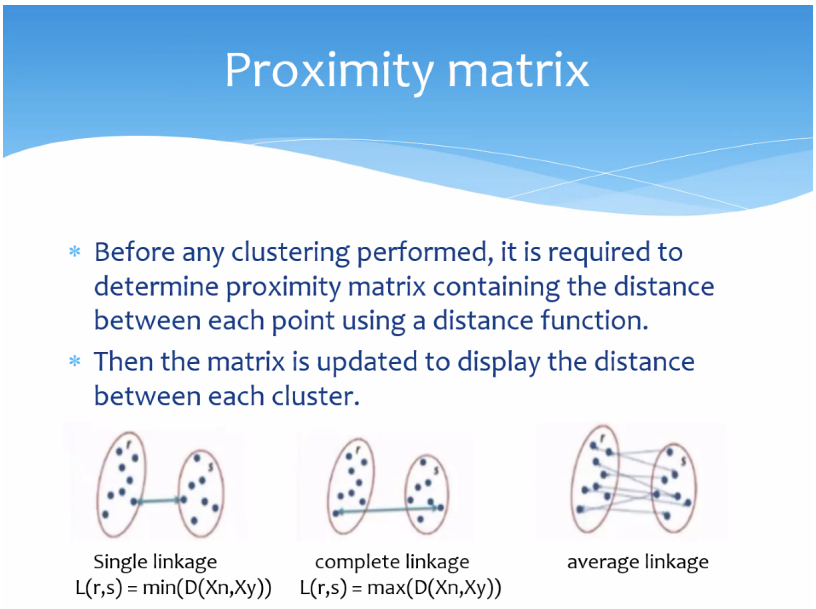




Agglomerative methods

Characteristics :

R & S are 2 clusters, same linkage, the length is based on min distance by 2 data points.

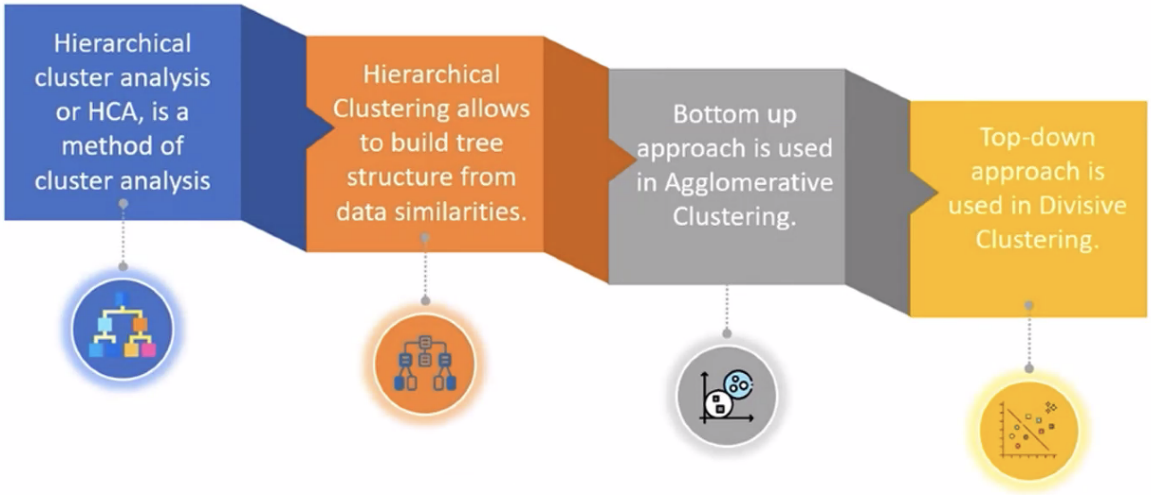


B) Partitional Clustering :

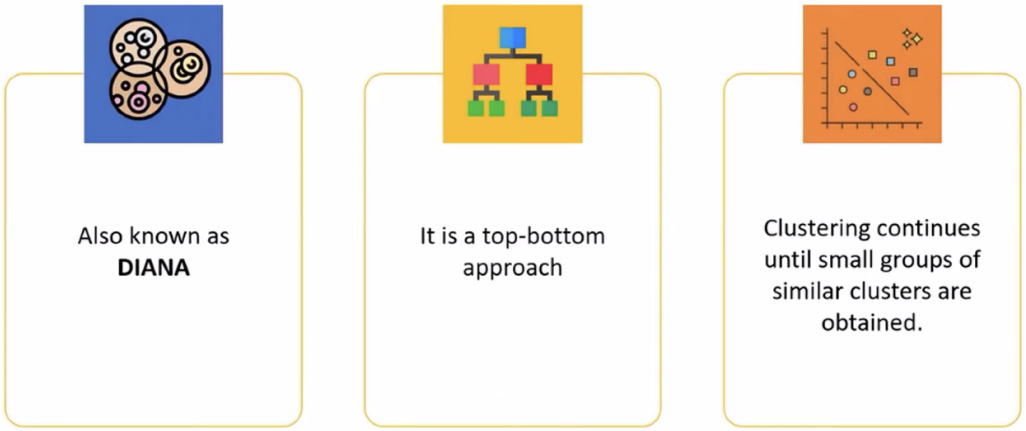
Decompose the data set into disjoint clusters.

K-mean algorithm.

Summary:



Divisive Clustering



K - Means clustering

It's an unsupervised learning algorithm that is used to solve the clustering problems in machine learning or data science.

It groups unlabeled datasets into clusters.
K = no. of predefined clusters that need to be created in the process.
Ex. : if K=2 ; there are 2 clusters.

Grouping based on similar properties needs to be done.

* It is an iterative algorithm that divides the unlabeled dataset into k different clusters in such a way that each dataset belongs only one group that has similar properties.
* It allows us to cluster the data into different groups and a convenient way to discover the categories of groups in the unlabeled dataset on its own without the need for any training.

Clustering Algorithm:

It is a centroid - base algorithm, where each cluster is associated with centroid.
Aim: Minimize the sum of distances between the data points and their corresponding clusters

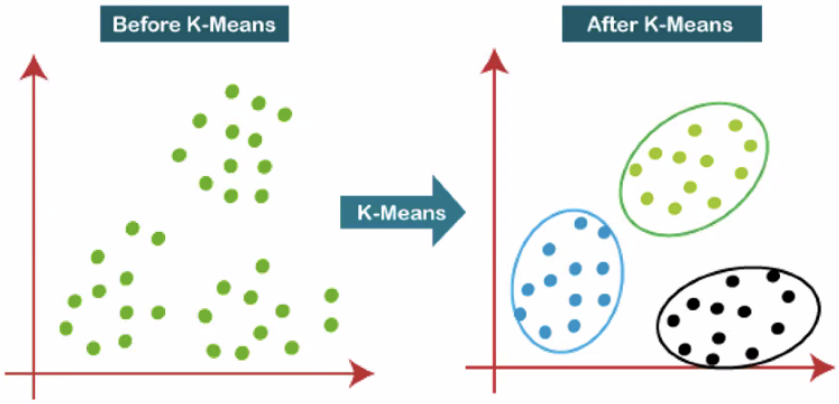
- Take unlabeled data
- Decide input in k no. of data.
- Repeat the process until it does not find the best clusters.

K needs to be predetermined in this algorithm.

Tasks:
Determined the best value for k center points or centroid

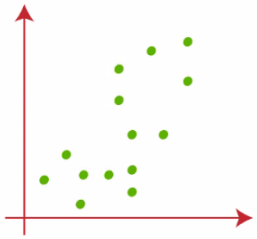
How it works:

Step 1: Select no. of k, i.e. decide the no. of clusters.
Step 2: Select random k points or centroids.
Step 3: Assign each data point to their closed centroid, which will form the predefined k clusters.
Step 4: Calculate the variance and place a new centroid of each clusters
Step 5: Repeat step 3
Step 6: If any reassignment occur, then back to steo4 else go to finish
Step 7: MODEL READY!!



Working of K-Means clustering

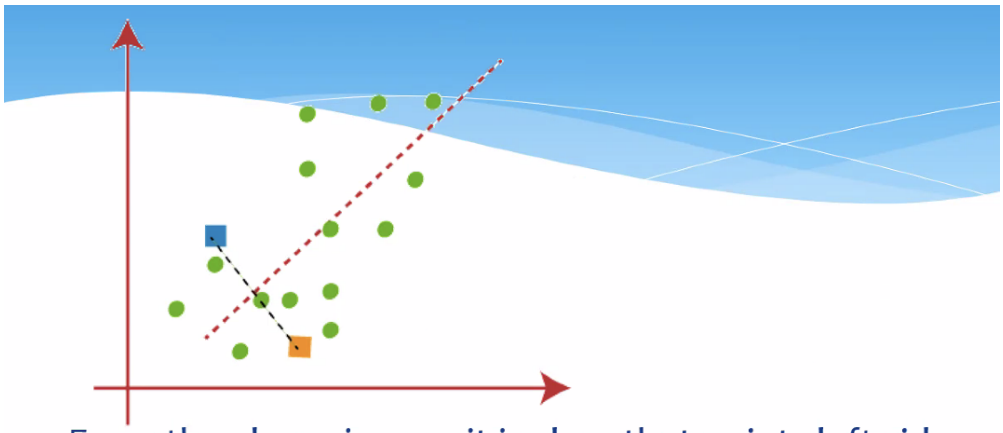
* Let's understand the above steps by considering the visual plots:
* Suppose we have two variables M1 and M2. The x-y axis scatter plot of these two variables is given below:



Working of K-Means clustering

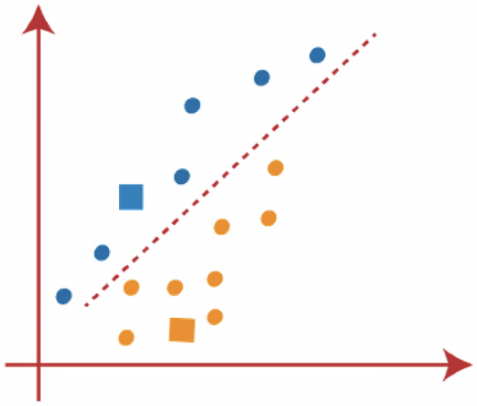
* Let's take number k of clusters, i.e., K=2, to identify the dataset and to put them into different clusters.
* It means here we will try to group these datasets into two different clusters.
* We need to choose some random k points or centroid to form the cluster.
* These points can be either the points 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:

* 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:

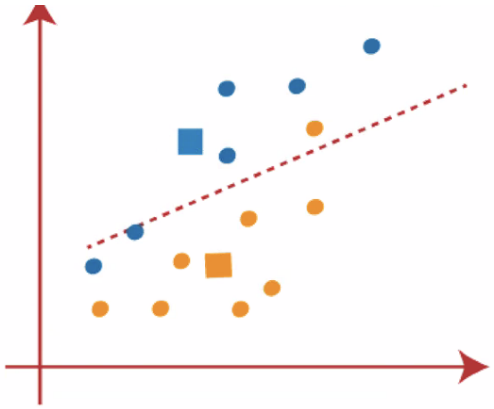


* From the above image, it is clear that points left side of the line is near to the K1 or blue centroid, and points to the right of the line are close to the yellow centroid. Let's color them as blue and yellow for clear visualization.

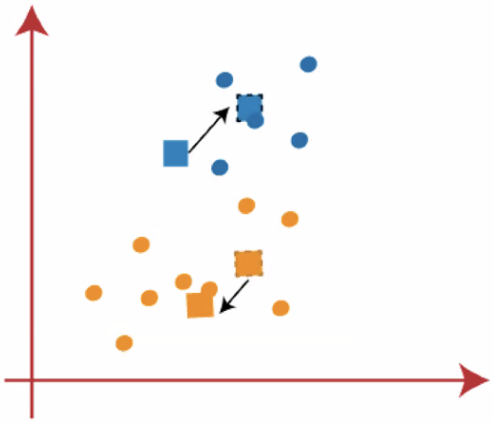
* As we need to find the closest cluster, so we will repeat the process by choosing **a new centroid**.
* To choose the new centroids, we will compute the center of gravity of these centroids, and will find new centroids as below:



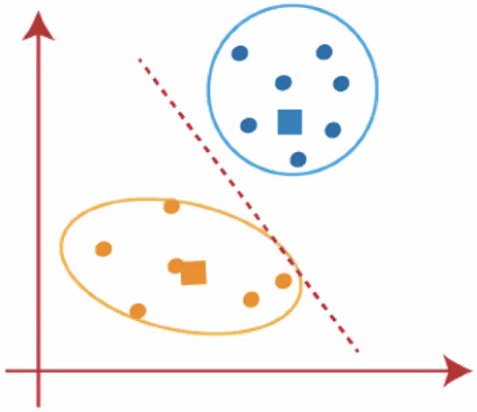
* From the image, we can see, one yellow point is on the left side of the line, and two blue points are right to the line.
* So, these three points will be assigned to new centroids.



* 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:



* We can see in the above image; there are no dissimilar data points on either side of the line, which means our model is formed. Consider the below image:



How to choose the value of k?
Different ways:

1) **Elbow method**: concept of WCSS Value.
   Within the cluster Sum of Squares.

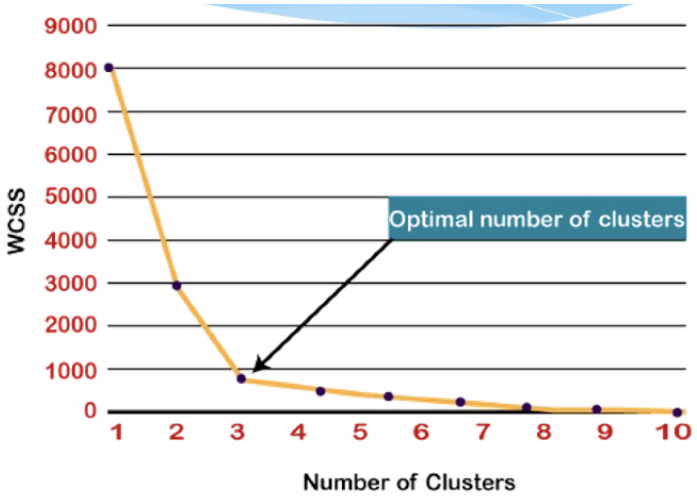
   Formula:

$$WCSS = \sum_{Pi \text{ in Cluster}_1} distance(P_i\ C_1)^2 + \sum_{Pi \text{ in Cluster}_2} distance(P_i\ C_2)^2 + \sum_{Pi \text{ in CLuster}_3} distance(P_i\ C_3)^2$$

$\sum_{Pi \text{ in Cluster}_1} distance(P_i\ C_1)^2$:

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.

- Range : 1 - 10, values of k
- For each k, calculate WCSS Value.
- Plot a curve between calculated CSS Values and no. of clusters k.
- Bend of the plot look like an arm, then that point is considered as the best value of k
- Example:



Implementation in Python

Step 1:

- **Importing Libraries:**

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

Matplotlib: for plotting the graph
Numpy: performing mathematical calculations.
Pandas: managing the dataset.

- **Importing data:**

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

- **Data:**
  Find patterns in dataset, need some patterns.

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

- **Extracting independent variables.**

Step 2:

Wcss - Y axis
Clusters: X axis

```
#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= 42)
   kmeans.fit(x)
   wcss_list.append(kmeans.inertia_)
mtp.plot(range(1, 11), wcss_list)
mtp.title('The Elobw Method Graph')
mtp.xlabel('Number of clusters(k)')
mtp.ylabel('wcss_list')
mtp.show()
```

we have used **the KMeans** class of sklearn. cluster library to form the clusters.

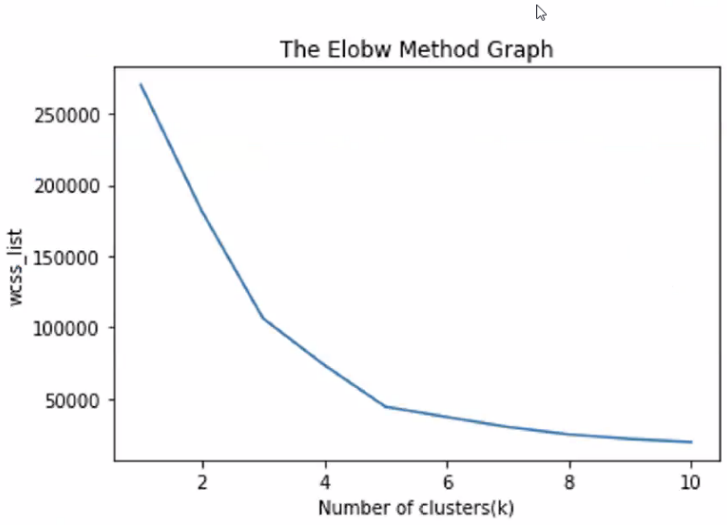
Next, we have created the **wcss_list** variable to initialize an empty list, which is used to contain the value of wcss computed for different values of k ranging from 1 to 10.

After that, we have initialized the for loop for the iteration on a different value of k ranging from 1 to 10; since for loop in Python, exclude the outbound limit, so it is taken as 11 to include 10^{th} value.

we have fitted the model on a matrix of features and then plotted the graph between the number of clusters and WCSS.

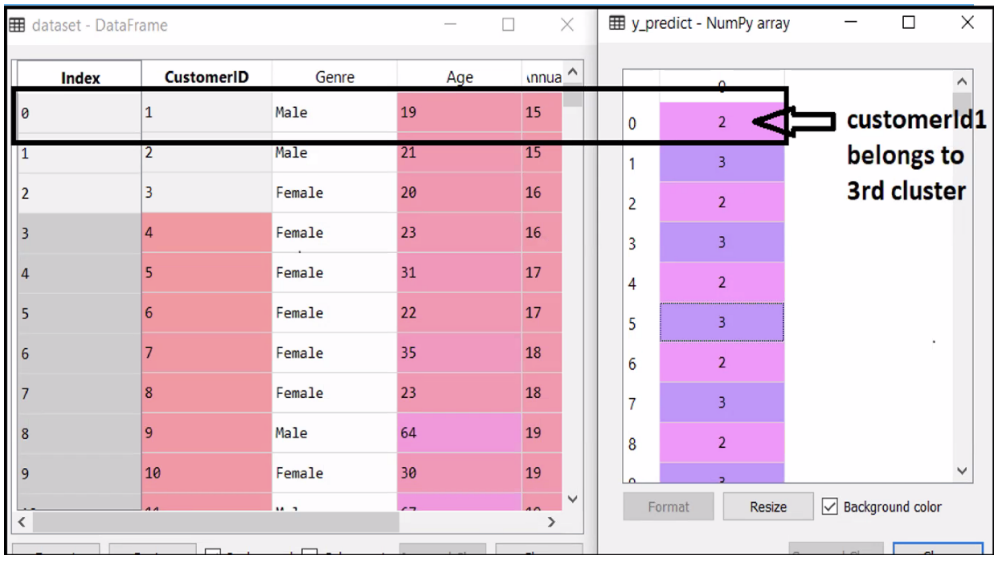
Output: After executing the above code, we will get the below output:

Output:



Training Set

```
#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)
```



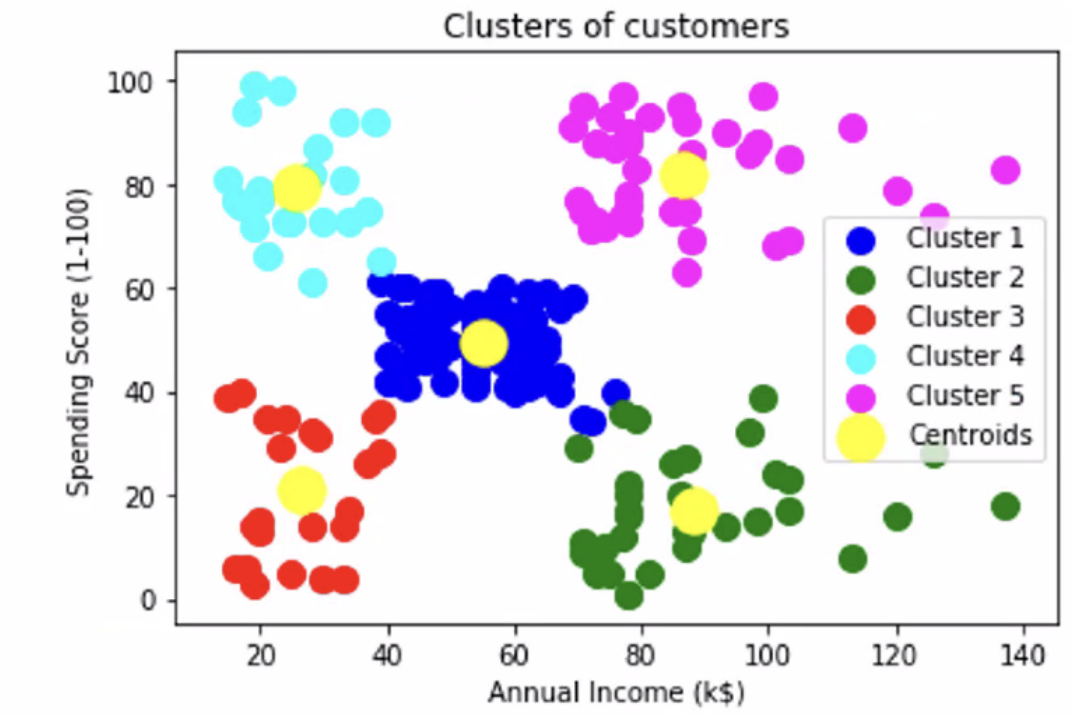
Step 4:

Visualizing the clusters

mtp.scatter() function of matplotlib

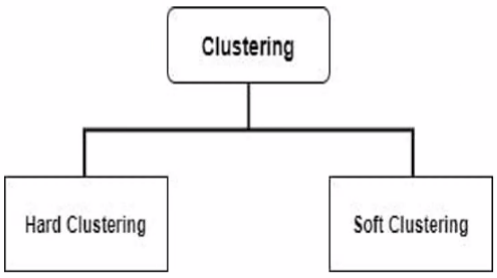
```
#visulaizing the clusters
mtp.scatter(x[y_predict == 0, 0], x[y_predict == 0, 1], s = 100, c = 'blue', label = 'Cl
uster 1') #for first cluster
mtp.scatter(x[y_predict == 1, 0], x[y_predict == 1, 1], s = 100, c = 'green', label = 'Cl
uster 2') #for second cluster
mtp.scatter(x[y_predict== 2, 0], x[y_predict == 2, 1], s = 100, c = 'red', label = 'Clus
ter 3') #for third cluster
mtp.scatter(x[y_predict == 3, 0], x[y_predict == 3, 1], s = 100, c = 'cyan', label = 'Cl
uster 4') #for 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()
```

Here,
5 different clusters with different colors
Clusters are formed between 2 parameters.

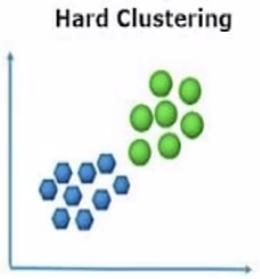


C- Means Clustering

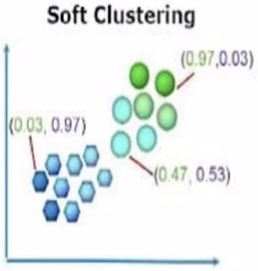
- Unsupervised ML technique
- Divide population in clusters, data points in the same group are similar to each other and in different groups are dissimilar.
- C-means clustering or fuzzy is a soft clustering technique in ML
- Each data point is separated into different clusters and then assigned a probability score for being in that cluster.
- Fuzzy c- means clustering gives better results for overlapped data sets compared to k-means clustering.



Hard Clustering:
For each datapoint it may either completely belong to a cluster or not
k-means clustering is a hard clustering algorithm



Soft Clustering:
Instead of putting each datapoint into separate clusters, the probability of that point is assigned to a probable cluster.



Fuzzy C-means clustering

Its is a clustering form
One data point can potentially belong to multiple clusters.
Outcome is probability based..
Has better accuracy as in most scenario the data points overlap
Ex:
Customer classification in marketing sector

Objective:
To reduce the number of units that have % in an image. It helps to go for a higher level of compression without losing the important useful info.
It helps for classification of large amount of image data

Importing python libraries

```
In [ ]: import numpy as np
        import pandas as pd
        from fcmeans import FCM
        from matplotlib import pyplot as plt
```

```
In [ ]: fcm = FCM(n_clusters=2)
        fcm.fit(X)
```

```
In [ ]: # outputs
        fcm_centers = fcm.centers
        fcm_labels = fcm.predict(X)
```

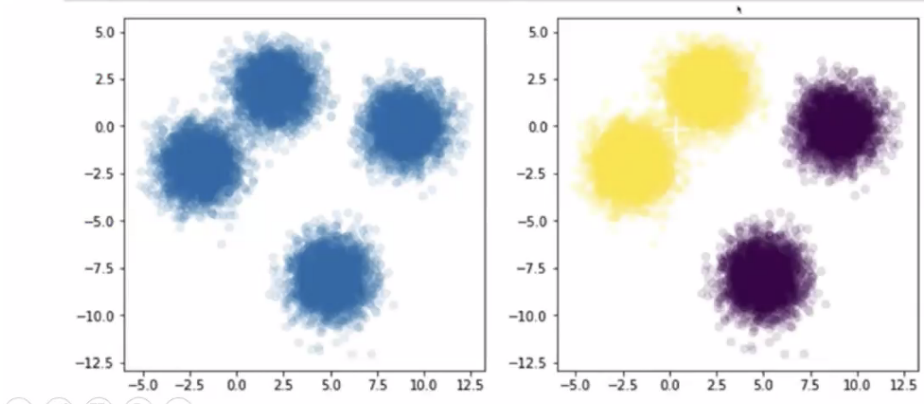
```
In [5]: fcm_centers
```

```
Out[5]: DeviceArray([[ 6.7439976 , -4.1813655 ],
                     [ 0.36480576, -0.18070485]], dtype=float32)
```

```
In [6]: np.unique(fcm_labels)
```

```
Out[6]: array([0, 1], dtype=int32)
```

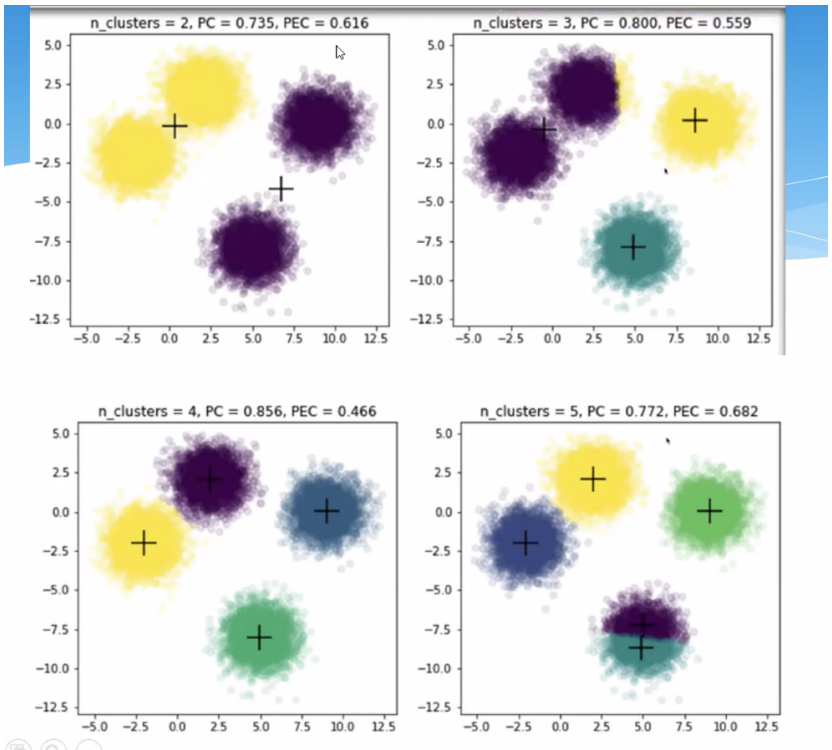
```
In [ ]: # plot result
        f, axes = plt.subplots(1, 2, figsize=(11,5))
        axes[0].scatter(X[:,0], X[:,1], alpha=.1)
        axes[1].scatter(X[:,0], X[:,1], c=fcm_labels, alpha=.1)
        axes[1].scatter(fcm_centers[:,0], fcm_centers[:,1], marker="+", s=500, c
        plt.show()
```



In [ ]:
```
n_clusters_list = [2, 3, 4, 5, 6, 7]
models = list()
for n_clusters in n_clusters_list:
    fcm = FCM(n_clusters)
    fcm.fit(X)
    models.append(fcm)
```

In [9]:
```
# outputs
num_clusters = len(n_clusters_list)
rows = int(np.ceil(np.sqrt(num_clusters)))
cols = int(np.ceil(num_clusters / rows))
f, axes = plt.subplots(rows, cols, figsize=(11,16))
for n_clusters, model, axe in zip(n_clusters_list, models, axes.ravel()):
    # get validation metrics
    pc = model.partition_coefficient
    pec = model.partition_entropy_coefficient

    fcm_centers = model.centers
    fcm_labels = model.predict(X)
    # plot result
```



Difference between

K-means clustering	Fuzzy C-mean clustering
Faster than c-mean clustering	Slower than k-mean clustering (more work)
	Each point is evaluated with each cluster, and more operations are involved in each evaluation.
K-Means just needs to do a distance calculation,	whereas fuzzy c means needs to do a full inverse-distance weighting.
each point is belonging to a centroid in K-means,	but in fuzzy c-means each point can be belonging to two centroids but with different quality.

Instance Based Learning

It is instance based as it builds the hypotheses from the training instances, also known as memory - based learning.

- What total dataset it is, how your model is going to process your data, on this processing your data depends.
- Time complexity algorithm as based on instance- based learning.

Advantages:
Local approximations can be made to target functions
This algo can adapt to new data easily.

Disadvantages
Classification costs are high, if we apply in a hardware environment.
Large amount of memory is required for storing the data.

Instance- based Learning Algorithm:

- KNN
- SOM
- LVQ
- LWL
- Case- Based Reasoning.

Conclusion

It was a insightful sessions and after the session i got a deep understanding of k-means clustering, C-means and hierarchical algorithm, Instance based learning theory based and practical.