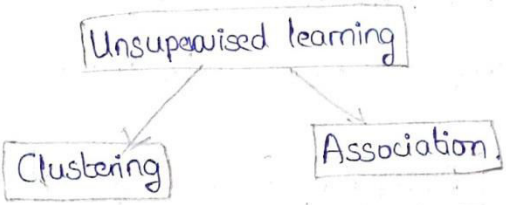
UNIT-IV

Unsupervised Learning

As the name suggests, unsupervised learning is a machine learning technique in which models are not supervised using training data set.

* Simply we say unsupervised learning is a type of machine learning in which models are trained using unlabelled dataset and are allowed to action that data without any supervision.

Types of unsupervised learning:-



Unsupervised learning algorithms:-

1. K-means clustering.
2. Hierarchical Cluster Analysis (HCA).
3. DBSCAN - (Density Based Special Clustering of Applications with Noise)
4. Principal Component Analysis (PCA)
5. KNN (K-Nearest Neighbour)
6. Neural Networks.

Clustering:-

A cluster refers to a collection of data points aggregated together because of certain similarities.

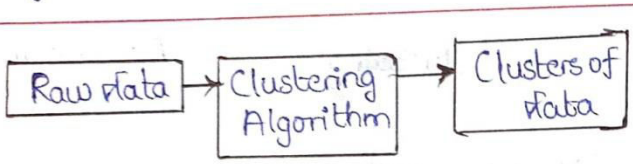
Clustering is a method of grouping the objects into classes such that objects with most similarities with the remains into a group and

has less or no similarities with the objects of another group.

* Cluster analysis find the commonalities b/w the data objects and categories them as per the presence and absence of these commonalities.

* The clustering technique is commonly used for "stastical data analysis."

Note:- Clustering is some where similar to the classification algorithm, but the difference is type of data set, that we are using.
In classification, we works with the labelled dataset, where as in clustering, we work with the unlabelled dataset.



Types of Clustering:-

1. Hard Clustering:- Data points belongs to only one group.
2. Soft Clustering:- Data points can belong to another group o.

Note:- The maximum possible no. of clusters will be equal to the no. of observations in the data set.

Clustering Methods:-

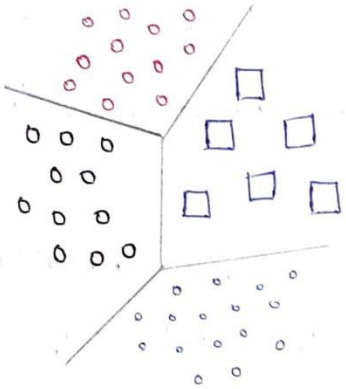
1. Partitioning Clustering / Centroid-based method.
2. Density-Based Clustering.
3. Distribution-Model Based Clustering.
4. Hierarchical Clustering.

Note:- Instances (datapoints) centered around a particular point is called centroid.

Partitioning Clustering:-

It is a type of clustering it divides the data into non-hierarchical groups. It is also known as the centroid-based method.

* The most common example of partitioning cluster is the "K-means Clustering Algorithm."



Main applications of clustering algorithm:-

1. Customer segmentation.
2. Data analysis
3. Dimensionality reduction
4. Feature Engineering
5. Outlier detection (Anonymous detection).
6. Semi Supervised learning
7. Search Engines
8. Image segmentation.

1. Customer segmentation:-

We can cluster our customers based on their purchases and their activities on our website. Customer representation can be useful in "recommender systems" to suggest content that other users in same cluster.

Data Analysis:-

When you analyze a new dataset, it can be helpful to run a clustering algorithm, and then analyze each cluster seperately.

Dimensionally reduction:-

Once a dataset has been clustered, it is usually possible to measure each instances affinity with each other cluster, affinity is any measure of how well an instance fits into a cluster, if there are k-classes then this vector is k-dimensional.

Feature Engineering:-

The cluster affinities can effort the useful as extra features and they helped us get their better performance.

Anomaly detection:- (also called outlier detection)

Any instance that has a low affinity to all the clusters is likely to be an anomaly.

Image segmentation:-

By clustering pixels according to their color, then replacing each pixel colors with the mean color of its clusters, it is possible to considerily reduce the no. of different colors in an image.

Image segmentation is used in many direct detection and tracking systems.

Search engines:-

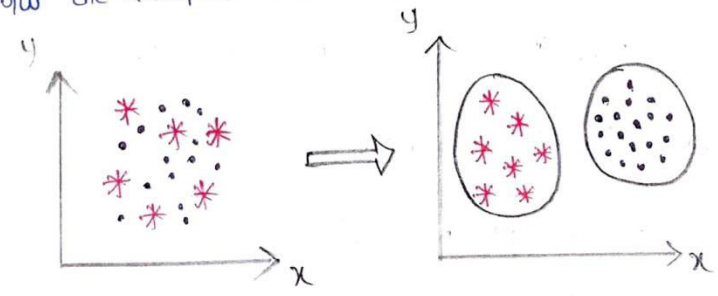
Some search engines lest us search for images that are similar to a reference image. To build such a system, you would first apply a clustering algorithm for all the images in a data base; Similar images would end up in the same clusters.

Semi-Supervised Learning:-

If you only have a few labels, you could perform clustering and propagate the labels to all the instances in the same cluster. This technique can greatly increase the no. of labels available for a subsequent supervised learning algorithm, and thus improve its performance.

K-means Clustering:-

* Proposed by "Stuart Lloyd" in "1957" and published by "Edward W. Forgy".
* K-means clustering is a distance based unsupervised clustering Given dataset of items, which certain features and values for these features, the algorithm will categorize the items into K groups or cluster of similarity.
* To calculate the similarity, we can use the Euclidean distance.
* It is an "iterative algorithm" that divides the unlabelled dataset into K different clusters in such a way that each dataset belongs only one group that has similar properties.
* It is also a "centroid-based" algorithm, where each cluster is associated with a associative centroid. Centroid can be thought as the "point that is most representative of the cluster."
* The main aim of this algorithm is to minimize the sum of distances blw the data point and corresponds clusters.



Note:- K-map is sometimes referred to as th Lloyd-progy algorithm.

Algorithms:-

Step-1:- Randomly select the 'K' cluster centers denoted by $K_1, K_2, \ldots \vec{K_n}$.

Step-2:- Calculate the distance between each datapoint 'as' and each cluster center $\vec{K_n}$.

Step-3:- Assign each datapoint is to the cluster centre $\vec{K_n}$ for which the distance is "minimum".

Step-4:- Recalculate each cluster centre/centroid by taking the average/mean of "cluster."

Step-5:- Repeat step-2 to step-5 untill the recalculated cluster center values are same (or) Datapoints in the cluster are same.

n data items/objects → K partitions/parts/groups.

each cluster. ↓ Represent cluster

$K \leq n$

↓

no. of total records.

Note:-

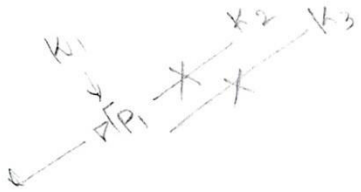
1. The K-means algorithm identifies the K no. of centroids and then allocated every data point to the nearest cluster, while keeping the centroids as small as possible.

2. The means in the K-means algorithm refers to averaging of the data, that is finding centroid.

Partitions should satisfy 2 rules:-

1. Each partition → atleast one object.

2. Each object should belong to only 1 partition only.



Ex:- Using K-means clustering algorithm form two classes for given data.

Height	185	170	168	179	182	188	180	180	183	180	180	177
Weight	72	56	60	68	72	77	71	70	84	68	64	76.

Divide the data into (2) clusters

cluster - centroid

first time - We select any 2 values as centroid randomly.

Here, I've are taking 1, 2 pairs

(185, 72) (170, 56)
K_1 K_2

So, now we start our clustering from 3rd one onwards.

Step-1:- Decide Centroids

K_1: (185, 72) K_2: (170, 56)

(K_1) (K_2)

Step-2:- Find the shortest distance between Euclidean distance

[ED] $ED = \sqrt{(x_0-x_1)^2+(y_0-y_1)^2}$ | $\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}$.

x_0, y_0 = observe values.

x_c, y_c = centroid values.

For data points (3) → (168, 60)

$K_1 = \sqrt{(168-185)^2+(60-72)^2} = 20.80$

$K_2 = \sqrt{(168-170)^2+(60-56)^2} = 4.48$ (Minimum distance)

$K_2 \leq K_1$

Step-3:- Now calculate new centroid distance. calculation

$K_2 = \left[\frac{170+168}{2}, \frac{56+60}{2}\right] = (169, 58)$ → new centroid

previous (170,56)

Datapoint 3 (168,60).

Updated centroids

K_1:(185,72) K_2:(169,58)

↑ K_1 ↑ K_2

Step-4:- Find the shortest distance for datapoint (4).

$K_1 = \sqrt{(179-185)^2 + (68-72)^2} = 7.21$ (Minimum Distance)

$K_2 = \sqrt{(179-169)^2 + (68-58)^2} = 14.14$

$K_1 < K_2$

data point (4) belongs to K_1

Repeat step-3:- For calculate new centroid for K_1, previous centroid (185,72)

data move point is → (4).

(179,68)

new centroid = $\left(\frac{179+185}{2}, \frac{68+72}{2}\right)$

$K = (182,70)$

for 5, $K_1 = (182,70)$, $K_2 = (169,58)$

(182,72) $K_1 = \sqrt{(182-182)^2 + (72-70)^2} = \sqrt{4} = 2.$

$K_2 = \sqrt{(182-169)^2 + (72-58)^2} = \sqrt{169+196} = 110.$

$K_1 < K_2.$

for 5, datapoint is → (5)

(182,72) new centroid = $\left(\frac{182+182}{2} + \frac{70+72}{2}\right)$

$K_1 = (182,71).$

for (6) $K_1 = (182,71)$, $K_2 = (169,58)$

(188,74)

$K_1 = \sqrt{(188-182)^2 + (74-71)^2} = \sqrt{36+36} = \sqrt{72} = 8.48.$

$K_2 = \sqrt{(188-169^2) + (74-58)^2} = \sqrt{1024+9^2} = \sqrt{361+36)} = 26.87$

$K_1 \leq K_2$.

New centroid $K_1 = \left(\frac{182+188}{2}, \frac{77+71}{2}\right) = (185, 74)$

Now, $K_1 = (185, 74)$.

Data point belongs to K_1.

Now $K_1 = (185, 74)$, $K_2 = (169, 58)$

Datapoint (7)

(180, 71) $K_1 = \sqrt{(180-185)^2+(71-74)^2} = \sqrt{25+9} = \sqrt{34} = 5.83$

$K_2 = \sqrt{(180-169)^2+(71-58)^2} = \sqrt{121+169} = \sqrt{290} = 17.029.$

Minimum distance $K_1 = 5.83 \rightarrow$ So datapoint belongs to K_1.

Now new centroid of K_1 is

(185, 74) Data (180, 71) New centroid $= \left(\frac{185+180}{2}, \frac{74+71}{2}\right) = (182.5, 72.5)$

$K_1 = (182.5, 72.5)$, $K_2 = (169, 58)$.

Datapoint (8)

(180, 70) $K_1 = \sqrt{(180-182.5)^2+(70-72.5)^2} = \sqrt{(2.5)^2+(2.5)^2} = \sqrt{12.5} = 3.53$

$K_2 = \sqrt{(180-169)^2+(70-58)^2} = \sqrt{121+144} = \sqrt{265} = 16.27.$

Point 8 $\in K_1$.

Now new centroid of K_1 is

Previous Data

(182.5, 72.5) (180, 70).

New centroid $= \left(\frac{182.5+180}{2}, \frac{72.5+70}{2}\right) = (181.25, 71.25)$.

Datapoint (9)

(183, 84) $K_1 = (181.25, 71.25)$, $K_2 = (169, 58)$

$K_1 = \sqrt{(183-181.25)^2+(84-71.25)^2} = \sqrt{3.0625+162.5625} = 12.89.$

$K_2 = \sqrt{(183-169)^2+(84-58)^2} = \sqrt{196+676} = \sqrt{872} = 29.529.$

point (9) $\in K_1$.

Now new centroid of K_1 is

previous Data

(182.5, 71.25) (83, 84)

$$K_1 = \left(\frac{181.25+180}{2}, \frac{71.25+70}{2}\right) = \left(\frac{361.25}{2}, \frac{141.25}{2}\right)$$

$$= (180.625, 70.625)$$

datapoint (10)

(180, 68) $K_1 = (180.625, 70.625)$ $K_2 = (169, 58)$

$$K_1 = \sqrt{(180-180.625)^2 + (68-70.625)^2} = \sqrt{(0.625)^2 + (-2.625)^2}$$

$$= 2.698.$$

$$K_2 = \sqrt{(180-169)^2 + (168-58)^2} = \sqrt{121+100} = \sqrt{221} = 14.866.$$

(10) $\in K_1$

New centroid of K_1.

(180.625, 70.625) Data (180, 68)

$$K_1 = \left(\frac{180.625+180}{2}, \frac{70.625+68}{2}\right)$$

$$K_1 = \left(\frac{360.625}{2}, \frac{138.625}{2}\right)$$

$$= (180.3125, 69.3125).$$

datapoint (11)

(180, 67) $K_1 = (180.3125, 69.3125)$ $K_2 = (169, 58)$

$$K_1 = \sqrt{(180-180.3125)^2 + (67-69.3125)^2}$$

$$= 0.09765.$$

$$K_2 = \sqrt{(180-169)^2 + (67-58)^2} = \sqrt{121+81} = \sqrt{202} = 14.21.$$

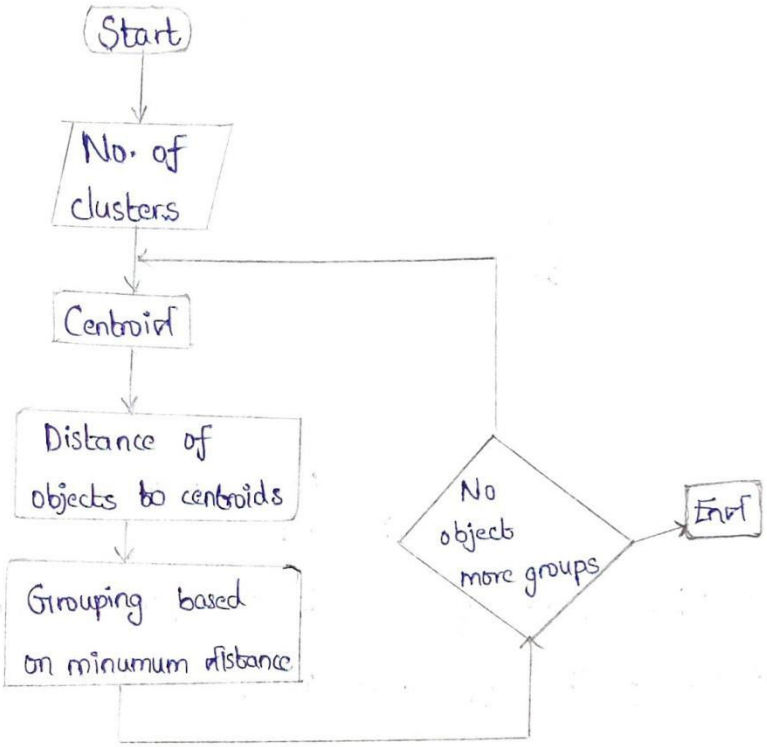
Finall clustering are:-

$$K_1 = \{1, 4, 5, 6, 7, 8, 9, 10, 11, 12\}$$

$$K_2 = \{2, 3\}$$

Now the data is derived into (2) different clusters.

K-means algorithm for clustering :-



The value of k is very crucial for optimal outcomes from the algo-rithm. There are several techniques to choose the optimal value for k including :-

1. Cross-Validation
2. Silhoutte Method
3. G-means Algorithm
4. Elbow method.

Here we will implement the elbow method to find the optimal value for k.

Elbow Method :-

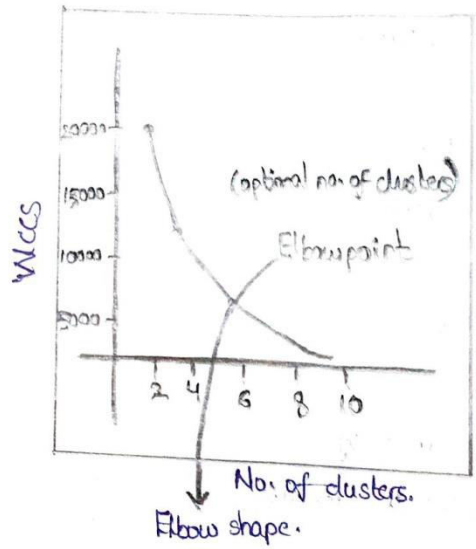
Step-1 :- Apply k-means clustering algorithm and form the clusters for different values for k.

Step-2 :- For each k calculate the Within Cluster Sum of Squares (WCCS).

$$WCSS = \sum_{Ck}^{Cn} \left(\sum_{di in ci}^{dm} distance(di, c_k)^2 \right)$$

Here c is the cluster centroids and di is the data point in each cluster.

Step-3:- Plot curve of WCCS according to the no. of clusters.



Step-4:- The location of bend in the plot is generally considered an indicator of the approximate no. of clusters.

Implementation:-

```
from sklearn.cluster import Kmeans
wcss = {}
for i in range (1,11);
        Kmeans = KMeans(n_clusters=i; int='Kmean++', random_state:42).
                                    n_init = 10);
        Kmeans. fit(x)
        wcss.append( K-means,interia_)
        plt.plot (range (1,11), WCSS)
        plt.ptlc ('The Elbow method')
        plt.xlabel ('No. of clusters')
        plt.ylabel ('WCSS')
        plt.show ( )
```

fitting K-means to the dataset K.

```
Kmeans = Kmeans (n_clusters = 5, init = 'K-mean++', random state=4);
y_kmeans = Kmeans.fit_predict(x).
```

n_clusters :- no. of clusters that we want, for this we follow the elbow menthod. default = 8.

init - 'kmean++', - Selects initial cluster centroids'

n_init = No. of times K-means algorithm is run with different centroid seeds.

n_init = "auto" / n_init = '10' - default.

inertia = Sum of squared distances of samples to their closet cluster center, weighted by the sample weights if provided.

Silhoutte Coefficient :-

⇒ The Silhoutte Coefficient or Sihhoutte Score is a measure of how similar data point is within cluster (cohesion) compared to other clusters (seperation).

⇒ for different value of K (say 1 - 10) calculate the silhoutte coefficient.

⇒ Plot silhoutte coefficient of the data point i.

⇒ The equation for calculating the sihoutte coefficient for a particular data point :-

$$S_{(i)} = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

i = one datapoint in dataset.

$S_{(i)}$ = is the silhoutte coefficeient of data point i

$a_{(i)}$ = is the average distance blw i and the other datapoints in the cluster to which i belongs to.

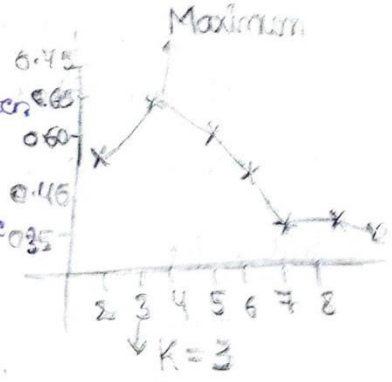
$b(i)$ = is the average distance from i to all clusters which i doesnot belong.

Note :- The value of the silhoutte coefficient is blw $\{-1, 1\}$.

⇒ A score 1 denotes the best, meaning that the data point is is very compact within the cluster to which it means and far away from other clusters.

→ The worst value is -1.

→ Value near '0' denote overlapping cluster.



Implementation :-

from sklearn. metrics import silhoutte_score

silhoutte_score (x, kmeans, labels).

Advantages of K-means algorithm:-

1. K-means algorithm is simple, easy to understand & easy to implement.
2. No other clustering algorithm performs better than K-means.
3. It is also coefficient, in which the time take to cluster k-means rises linearly with the no. of data points.

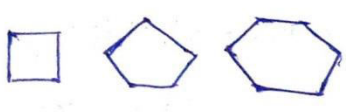
Disadvantages of K-means algorithm | Limits of K-means:-

1. The user needs to specify an initial value of K.
2. The process of finding the clusters may not converge.
3. It is not suitable for discovering clusters that are not hyper ellipsoids or hyper spheres.
4. K-means does not behave very well when the clusters have varying sizes, different densities or non-spherical shapes.

* To overcome K-mean algorithm, "David Arther & Sergi Vassilvitzsil" (2006) proposed K-mean ++.

* It works like a smarter initialization step that tends to select centroids.

* For different densities or non-converse shapes dataset "Gaussian mixture Models" work great.



Using Clustering for image segmentation:-

Segmentation:- Image segmentation is the task of partitioning an image into multiple segments. An image is a collection (or) set of different pixels we group together the pixels that have simi-lar attributes using image segmentation.

1. Color segmentation:- Pixels with a similar color get assigned to the same segment.

Ex:- If we want to analyze satellite images to measure how much total forest are a there is in a region, color segmentation may be is it fine.

2. Semantic segmentation:- All pixels that are part of the same object type get assigned to the same segment.

⇒ It is simply the task of assigning a class label to every single pixel of an input image.

3. Instance Segmentation:- All pixels that are part of the same individual object are assigned to the same segment.

⇒ Instance segmentation is very similar to semantic segmentation.

⇒ The only difference is that it distinguishes b/w different objects of the same class.

4. Panoptic segmentation:-

⇒ While semantic segmentation tells us the class label of every pixel in the images, instance segmentation differentiates b/w different objects of the same class.

⇒ Neither of them provides a complete understanding of the image.

* Semantic Segmentation is suited to labeling uncountable objects as "sky" or "ocean".

* Instance Segmentation is well suited for understanding countable objects.

* All the objects of an image can be classified this varying "countable" or "uncountable" classes. To address both types, Panoptic Segmentation comes to help.

* The state of art in semantic or instance segmentation to day is achieved using complete or complex architectures based on convolutional neural networks.

* The image may be in the format of jpg, png, ... and the grouping of segments can be done based on colours.

* If the picture is classified in sabillites it may have more than 3 colours → 3 channels.

* If the picture is captured by single colours → it is also called as gray scale → 1 channel.

* If the image consists of '3' colours i.e, Red, Blue, Green (RGB). it consists of '3' channels.

* We can identify the image by:

1. Height of image.
2. Width of image.
3. No. of channels.

} We will get by using shape().

Implementation:-

```
from mat.plotlib. image import imread
from sklearn. cluster import kmeans.
import matplotlib.pyplot as plt.
image = imread ("image.jpg")
image.shape
x.image.reshape (-1,3) // reshape the array get along list of RGB colors.
```

// (180, 281, 3) — height, width, channel/colors

x.shape() // (50, 580, 3) ← 180X281

pixels in image

```
Kmeans = kmeans (n_clusters = 8) fit(x)
segmented_img = kmeans.cluster_centers [kmeans.labels].
segmented_img2 = Segmented_img .reshape (image.shape)
plt.imshow (segmented_img2 , as type ("units"))
```

Image Segmentation Techniques

- Traditional
  - Region bage
  - Thresholding
  - Edge detection
  - Clustering
- Deep Learning
  - Uref
  - Mask R CNN
  - DeepLab
  - Interactive segmentals
  - SAM (Segment Anything Model)

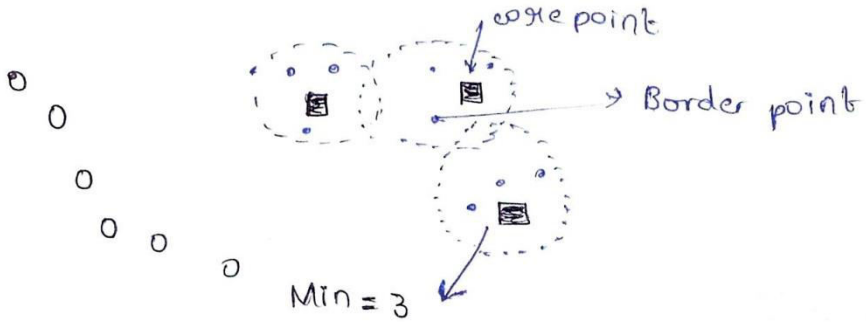
DBSCAN:-

DBSCAN stands for "Density Based Spatial Clustering of Application with Noise."

* It was proposed by "Martin Estern Etal" in "1996."
* DBSCAN is a density-based clustering algorithm that works on the assumption that clusters are dense regions in space seperated by regions of lower density.
* It can be density clusters in large spatial dataset by looking at the local desity, of the datapoints.
* It doesnot require the no. of classes to be told before hand, unlike K-means, where we have to specify the no. of centroids.
* DBSCAN requires only two parameters:-
  - Epsilon.
  - Minpoints.

Epsilon:- It is the radius of the circle to be created around each datapoint to check the density.

Min-points:- It is the minimum no. of datapoints required inside the circle of for that data point to be classified as a core point.



Types of Data points:-

core point :- It should satisfy the condition of has at least min points.

border point :- This is a point that has at least one core point of distance n.

noise point :- This is a point that is neither a core or not a border.

Ex:- Apply DBSCAN algorithm to given data points & create clusters with minpoints = 4 & Epsilon (E) = 1.9.

$P_1(3,7)$, $P_2(4,6)$, $P_3(5,5)$, $P_4(6,4)$, $P_5(7,3)$, $P_6(6,2)$, $P_7(4,2)$, $P_8(8,4)$, $P_9(3,3)$, $P_{10}(2,6)$, $P_{11}(3,5)$, $P_{12}(2,4)$.

	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}	P_{11}	P_{12}
$P_1(3,7)$	0											
$P_2(4,6)$	1.41	0										
$P_3(5,5)$	2.82	1.41	0									
$P_4(6,4)$	4.242	2.82	1.41	0								
$P_5(7,3)$	5.658	4.24	2.83	1.41	0							
$P_6(6,2)$	5.83	4.47	3.16	2.00	1.41	0						
$P_7(7,2)$	6.40	5.00	3.61	2.24	1.00	1.00	0					
$P_8(8,4)$	5.83	4.47	3.16	2.00	1.41	2.83	2.24	0				
$P_9(3,3)$	4.00	3.16	2.83	3.16	4.00	3.16	4.12	5.10	0			
$P_{10}(2,6)$	1.41	2.00	3.16	4.44	5.83	5.66	6.40	6.32	3.16	0		
$P_{11}(3,5)$	2.00	1.41	2.00	3.16	4.47	4.24	5.00	5.10	2.00	1.41	0	
$P_{12}(2,4)$	3.16	2.83	3.16	4.00	5.10	4.47	5.39	6.00	1.41	2.00	1.41	0

for finding nearest points, for this we check horizontally → vertically find $q \leq 19$ values. minpht = 4 & Epsilon (E) = 1.9.

P_1 : P_3, P_{10}

P_2 : P_3, P_{11}

P_3 : P_2, P_4

P_4 : P_3, P_5

P_5 : P_4, P_6, P_7, P_8

P_6 : P_5, P_7

P_7 : P_5, P_6

P_8 : P_5

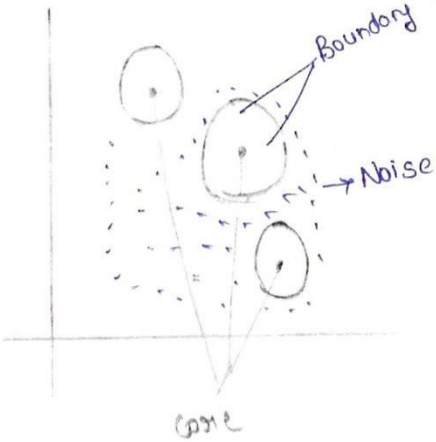
P_9 : P_{12}

P_{11} : P_2, P_{10}, P_{12}

P_{12} : P_9, P_{11}.

Finding core & Noise & Border points.

P_1	Noise	Border
P_2	Core	
P_3	Noise	Border
P_4	Noise	Border
P_5	Core	
P_6	Noise	Border
P_7	Noise	Border
P_8	Noise	Border
P_9	Noise	
P_{10}	Noise	Border
P_{11}	Core	
P_{12}	Noise	Border



Gaussian Mixtures:-

A Gaussian Mixture Model (GMM) is a probabilistic model that assumes that the instances were generated from a mixture of several gaussian distribution whose parameters are unknown.

All the instances generated from a single gaussian distribution form a cluster that typically looks like an ellipsoid. Each cluster can have a different ellipsoidal shape, size, density and orientation.

For each instance, a cluster is picked randomly form among K-cluster. The probability of choosing the ith cluster is the weight (θ_i). The index of the cluster choosen for the i^{th} instance is noted $z(i)$.

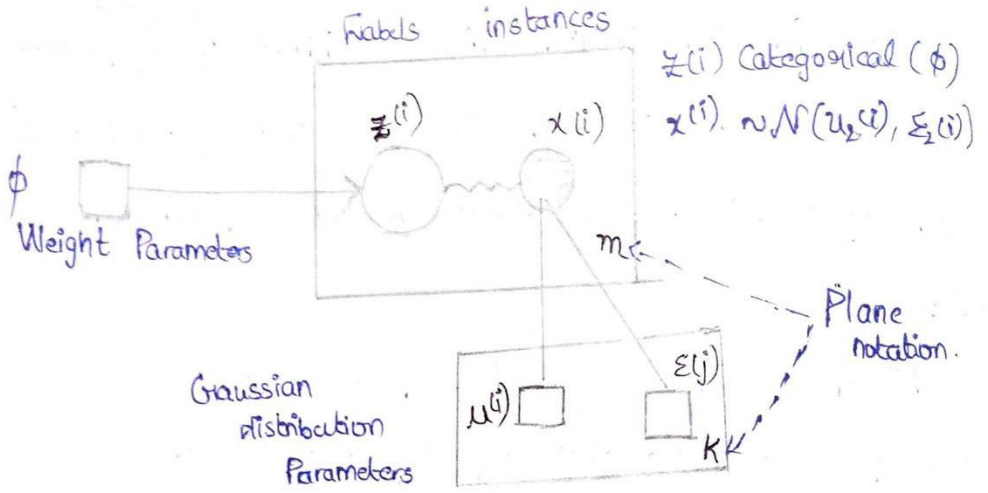
If the i^{th} instance was assigned to the i^{th} cluster ($i.e\ z(i)=j$), then the location $x(i)$ of this instance is sampled randomly form Gaussian distribution with mean $\mu^{(j)}$ & convariance matrix (Σ_j).

This noted

$$x^{(i)} \sim N(\mu^{(j)}, \Sigma^{(j)})$$

Comparison with K-means algorithm:-

Assigned each instance to exactly one cluster of clusters are overlapping? hard to tell which cluster is right K-means is a distance based model. Use the Euclidian distance. If the cluster has a non-circular shape it does not work properly.



⇒ It is also a "generative model" meaning you can actually sample new instances from it (note that they are ordered by cluster index).

⇒ It is also possible to estimate the desity of the model at any given location. This is achieved using the "score_sample()" method; for each instance, it is given that this method estimates the log of the "probability density function (PDF)" at that location. The greater scores the higher the density.

⇒ In Gaussian model mixture (GMM) we have covariance type hyper-parameter to one of the following values.

Spherical:- All clusters must be spherical, but they can have different parameters (i.e, different variances).

diag:- clusters can take on any ellipsoidal shape of any image, but the ellipsoids areas must be parallel to the coordinate axis (i.e, the covariance matrixes must be diagnol).

tied:- all clusters must have the ellipsoidal shape, size and orientation (ie, all clusters, share the same covariance matrix).

By default, the covariance type is equal to "full", which means

that each cluster can take on any shape, size and orientation (it has its own unconstrained covariance matrix).

The Expectation Maximization (EM):-

EM has algorithm, has many similarities with K-means algorithm It also initializes the cluster, parameters randomly, then repeats two steps until convergence.

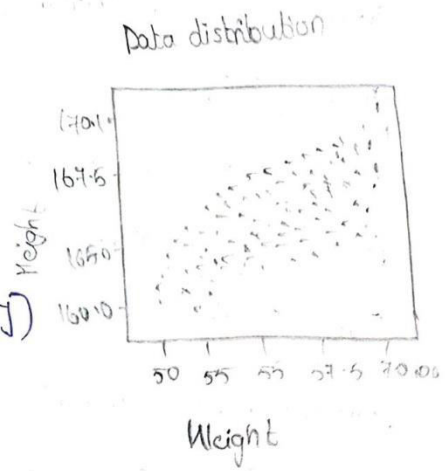
⇒ First assigning instances to clusters (this is called the "Extension step").

⇒ and updating the clusters (this is called "maximization step").

⇒ Unlike, K-means, EM uses "soft cluster" assignments rather than the "hard assignments."

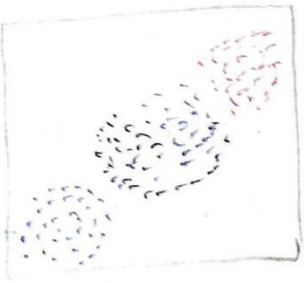
Comparison to K-means and Gaussian Mixture Model (GMM) in implementation:-

```
import pandas as Pd
import matplotlib.pyplot as plt
data = pd.read_csv("Clustering_gmm.csv")
plt.figure (figsize (7,7))
plt.scatter (data ["weight"], data ["Height"])
plt.xlabel ("weight")
plt.ylabel ("Height")
plt.title ("Data Distribution")
plt.show ().
```



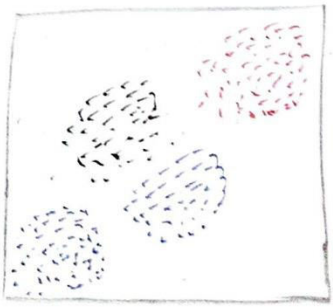

```
# training K means model: ①:-
from sklearn.cluster import Kmeans
Kmeans = Kmeans (n_clusters = 4)
```

```
Kmean fit (data)
pred = Kmeans.predict (data)
frame = Pd.Dataframe (data)
frame ['cluster'] = pred
frame.columns = ['weight','height','cluster']
colors = {'blue','green','cyan','black'}
for k in range (0,4);
    data = frame [frame {'cluster'} == k]
    plt.scatter (data ["weight"], data ["height"], c = colors [k])
plt.show().
```



train Gaussian Mixture Model:-

```
from sklearn.Mixture import GaussianMixture
gmm = GaussianMixture (n_components = 4)
gmm.fit (data)
labels = gmm.predict (data)
frame = pd.dataframe (data)
frame ['cluster'] = labels
frame.columns = {'weight','Height','cluster'}
colors = {'blue','green','cyan','black'}
for k in range of (0,4);
    data = frame {frame ['cluster'] == k}
    plt.scatter (data {'weight'}, data {'height'}, c = colors [k]}
plt.show().
```



Dimensionality Reduction:-

The no. of input features, variables and columns present in a given dataset is known as "dimensionality" and the process of reduce these features is called dimensionality reduction.

⇒ Dimensionality reduction is a way of converting the higher dimensions dataset into lesser dimensions data set ensuring that it provides similar information.

⇒ Simply we say it is a technique used to reduce no. of features in a dataset while retaining as much of the important information as possible.

⇒ This technique allows you to find small set of most impactfull features among a large no. of features with this small set of principal features, we can run our prediction algorithms easily with better accuracy.

⇒ It is commonly used in the fields that deal with high-dimensional data such as Speed recognization, Signal processing, Bio-informatics, Data visualization, Noice Prediction.

⇒ Dimensionaly reduction is also extremely useful for data visualization.

The main approaches to dimensionality reduction.

* Projection.
* Mainfold learning.

Note:-

Dimensionally refers to an minimum no. of coordinates needed to specify any point within a space or a object.

The most popular dimensionality reduction techniques:-

1. PCA (Principal Component Analysis) → Projection (linear dimensionality)
2. Kernel PCA
3. Random Projection
4. Locally Linear Embedding (LLE) → Manifold learning (Non-linear dimensionality)

The Curse of dimensionality:-

This learn was first introduced by Richard E. Bellman.

Curse of dimensionality refers to a set of problems that arise when working with high dimensionality data.

The dimensions of a data set corresponds to the no. of attributes / features that exist in a dataset. A dataset with a large no. of attributes, generally of the order of a hundred or more, is referred to as high dimensional data.

> **Note:-**
> High dimensional data is when a dataset a no. of features (p) that is bigger than the no. of observations (N). High dimension data is the problem that leads to the "curse of dimensionality. The equation of high dimensional data is usually written like $p \gg N$.

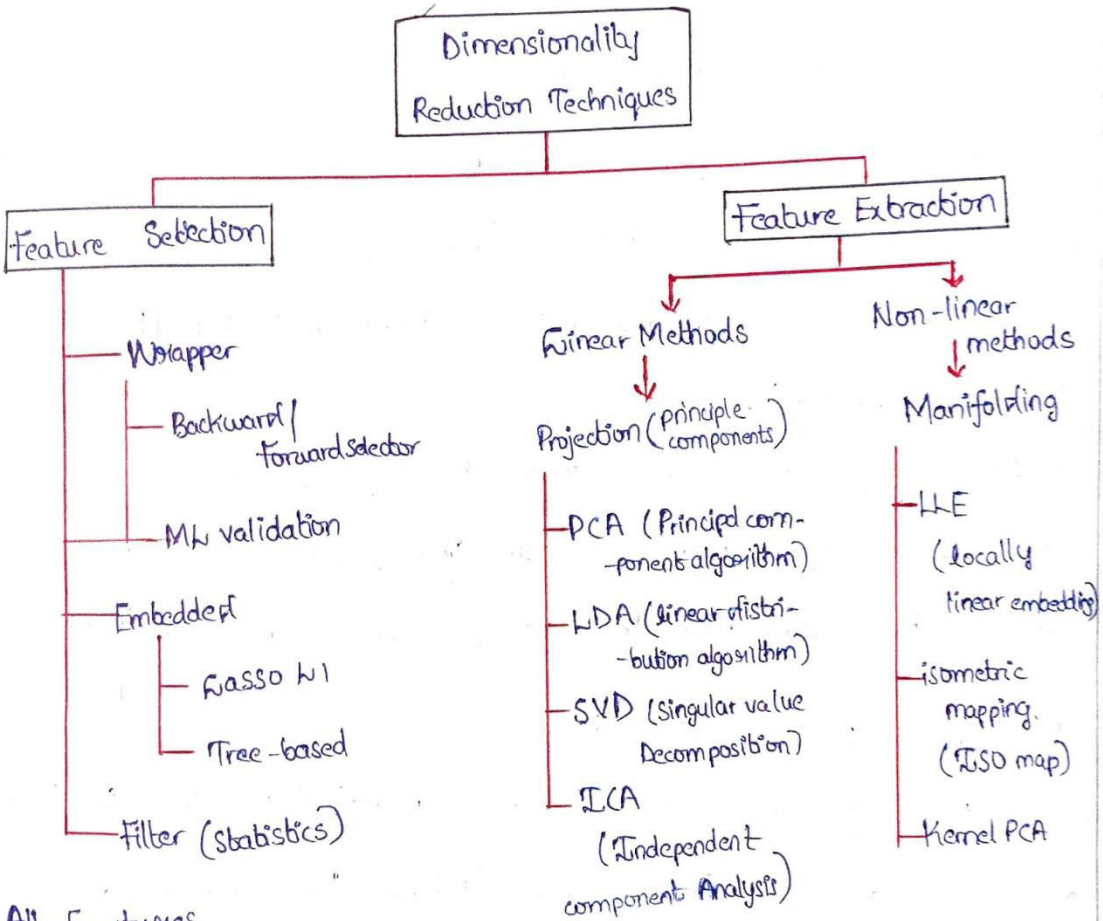
⇒ The difficulties related to training machine learning models due to high dimensional data is referred to as "Curse of dimensionality"

⇒ The popular aspects of the curse of dimensionality, "data sparsity" and "distance concentration."

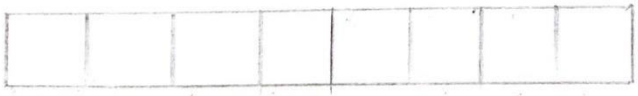
Main Approaches for Dimensionality Reduction:-

Generally, they are 2 main approaches to dimensionality reduction:-

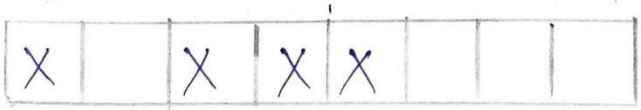
1. Projection
2. Manifold learning.



All features

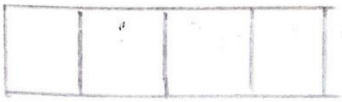


Feature Selection



Choosing most important features of data

Feature extraction.



Combining features to create new superfeatures.

Projection:-

Sometimes feature extraction is also called as feature projection, where in multidimensional space is converted into an space with lower dimensions.

⇒ Most of the train training instances ble close to each lower dimensional space of the high-dimensional space.

⇒ The idea behind projection is "to find the best surface and project each" of the "data instances" onto that lower-dimensional space

⇒ One of the most common projection technique is PCA (Principal Component Analysis).

⇒ However, projection is not always the best approaches, as in many cases, the subspace twist and turn. One famous example is the skill roll toy dataset. In these cases projection is not good technique.

Manifold Learning (ML):-

Recently a new approach named manifold learning (ML) or nonlinear dimensionality reduction (NDLR) has gained a great attention in the content of dimensionality reduction.

⇒ Manifold learning is based on the assumption that high-dimensional data are embedding in a manifold non-linear of lower dimension.

⇒ Non-linear dimensionality reduction, also known as manifolding refers to various related techniques that aim to project high-dimensional data onto lower dimensional "latent-manifolds."

⇒ The general idea behind manifold learning is to model the mani-

-fold on which the training model points inside. It takes an unsupervised approach that learns the high-dimensional structure of the data from the data itself.

⇒ A dimensional manifold is a part of an n-dimensional space (d<n) that resembles an d-dymensional hyperplane.

⇒ The methods based on manifold learning are a set of non-linear techniques for feature extraction. They rule on the premise that a set of values of high dimensionality is i need a body with a set of lower no. of dimensions, whose shape has been maximable to return in the latter. This can be seen as a 2D plane, which has been rolled up to result, to a time in a three dimensional-structure.

⇒ Many dimensionality reduction algorithms work by modeling the manifold on which the training instances be, this called manifold learning. It relies on the manifold "assumption" also called the "manifold hypothesis", which holds that most real-world high-dimensional dataset lie close to each lower-dimensional manifold.

⇒ The manifold assumption is often accompained by another implicit assumption; that the task at hand (eg:- classification or regression) will be simpler if expressed in the lower dimensional space of the manifold.

PCA (Principal Component Algorithm or Analysis) :-

Principal Computer Analysis, or PCA is a dimensionality reduction method that is often used to reduce the dimensionality of large datasets, by transforming a large set of variables / features. These new transferred features called principal components. It still contains most of the information in the large set.

> **Note:-** The idea of PCA is simple- reduce the no. of variables of a dataset while preserving as much information as possible.

⇒ PCA performs a linear direction or transformation of data, seeking directions of maximum variance.

⇒ The main aim of PCA is to find such principal components which can describe the data points with set of ... Well, principal components.

⇒ Principal components remove noise by reducing a large no. of features to just a couple of principal components.

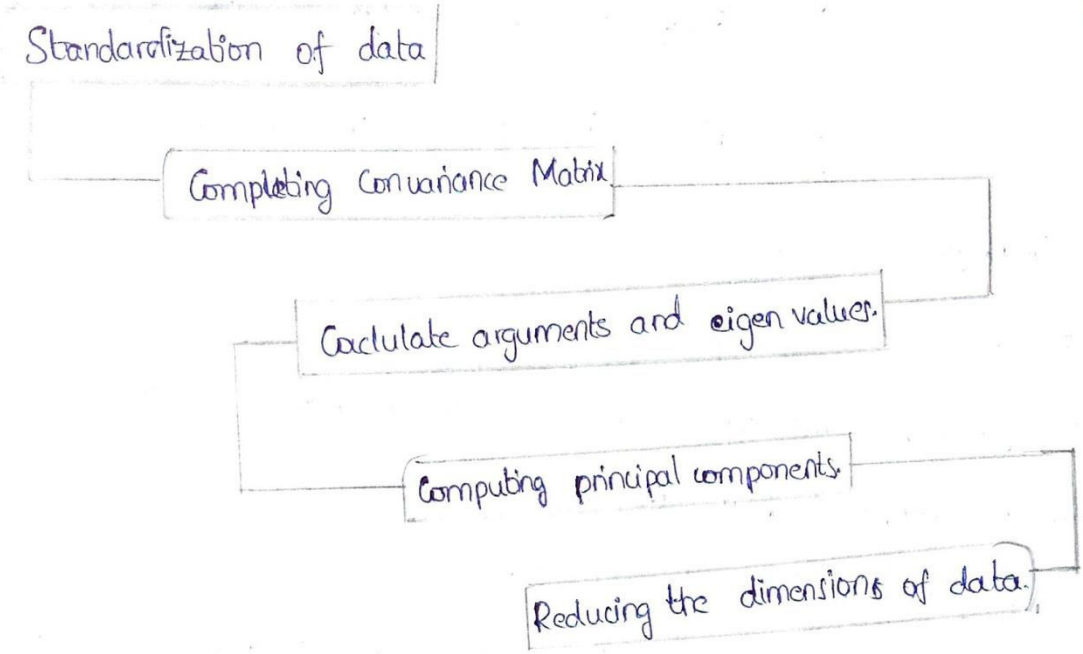
⇒ There are multiple ways to calculate PCA :-

1. Eigendecomposition of the covariance matrix.
2. Singular value decomposition of the data matrix (SVP).
3. Eigen value approximation via power iterative computation.
4. Non-linear iterative partial least squares (NIPLS) computation.

Step-by-Step of PCA:-

Personal Component analysis (PCA) is a dimensionality reduction technique that enables you to identify collerations & pattern in a

data set, so that it can be tranformed into a dataset of signifi-cantly lower dimension without loss of any import information.



1. Standardization of the data:-

Standardization is all about scaling under your data in such a way, that all the variables and their values lie within a similar range.

$$z = \frac{\text{value} - \text{mean}}{\text{Standard deviation}}.$$

2. Computing the covariance matrix:-

A covariance matrix exposes the the correlation b/w the differ-ent variables of the dataset. It is essential to identify heavily dep-erdent variables because they contain biased and redundant information which releases the overall performance of the model.

$$\begin{bmatrix} cov(a,b), & cov(a,b) \\ cov(b,a) & cov(b,b) \end{bmatrix} \longrightarrow$$

- The covariance vote denotes
- ⇒ How co-dependent two variables are
- ⇒ Negative co-variance indirectly proportional
- ⇒ Positive covariance directly proportionals

$$cov(a,b) = cov(b,a)$$

Note:-

Correlation - features how strongly they are related.

f_1 f_1 :- proportional +1 (+ve correlation)
+ +

$-f_1$ f_1 inversly -1 (-ve correlation).
+ −

Orthogonal not correlated ⇒ 0.

3. Calculating Eigen values and Eigen vectors:-

⇒ Eigen values and Eigen vectors are the mathematical constructs that must be computed from the covariance matrix in order to determine the principal components of the datasets.

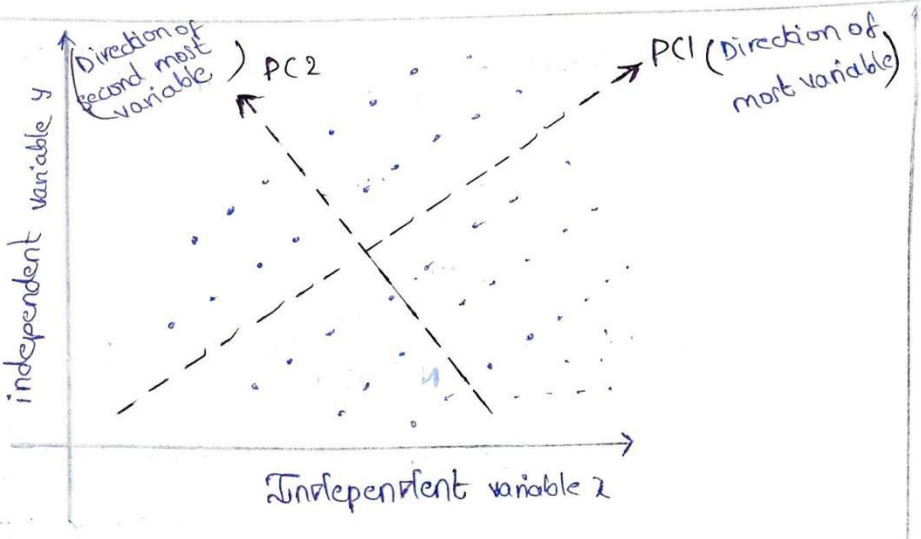
⇒ Principal components are the new set of variables that are obtained from the initial set of variables (features). They compress and possess the most of useful information that was scattered among the initial variables.

* Eigenvalues are those vectors when linear transformation is performed on them then their direction does not change.
* Eigenvalues simply denote the scalars of the respective eigenvectors.

4. Computing the principal components:-

Once we have computed the Eigen values & Eigen vectors, all we have to do is order them in the descending order, where the eigen vector with the highest eigenvalue is the most significant and this forms first personal component.

* PC1 is the most singificant and shows the maximum possible information.
* PC2 is the second most significant and stores remaining maximum info and so on.



5. Reducing the dimensions of the dataset:-

The last step performing in PCA is to re-arrange the original data with the final principal components which represents the maximum and the most significant information of the dataset.

Example:-

Given the data in the table, reduce the dimension from 2 to 1 using the principal component analysis (PCA) algorithm.

Feature	Ex 1	Ex 2	Ex 3	Ex 4
x	4	8	13	7
y	11	4	5	14

Step-1:- Calculate Mean

No. of features, $n = 2$.

No. of samples, $N = 4$.

$$\vec{x_1} = 1/4\,(4+8+13+7) = 8$$

$$\vec{x_2} = 1/4\,(11+4+5+14) = 8.5$$

x, y ⇒ n features
then no. of pairs
$= n^2 \Rightarrow 2^2 = 4$.

Step-2:- Calculation of the covariance matrix

$$S = \begin{bmatrix} cov(x_1,x_1) & cov(x_1,x_2) \\ cov(x_2,x_1) & cov(x_2,x_2) \end{bmatrix}$$

Ordered pairs are
(x,x), (x,y), (y,x), (y,y).

$$cov(x,y) = \frac{\sum (x_i - \vec{x})(y_i - \vec{y})}{n}$$

x_i → data value of x
$\vec{x}$ → mean value of x
y_i → data value of y
$\vec{y}$ → mean value of y
n → No. of data values.

$$\text{cov}(x_1,x_1) = \frac{1}{N-1}\sum_{K=1}^{3}(x_{1K}-\bar{x}_1)(x_{1K}-\bar{x}_1)$$

$$= \frac{1}{3}\left((4-8)^2+(8-8)^2+(13-8)^2+(7-8)^2\right)$$

$$= 14.$$

$$\text{cov}(x_1,x_2) = \frac{1}{N-1}\sum_{K=1}^{N}(x_{1K}-\bar{x}_1)(x_{2K}-\bar{x}_2)$$

$$= \frac{1}{3}\left((4-8)(11-8.5)+(8-8)(4-8.5)+(13-8)(5-8.5)+\right.$$
$$\left.+(7-8)(14-8.5)\right.$$

$$\text{cov}(x_1,x_2) = -11.$$

$$\text{cov}(x_2,x_1) = \text{cov}(x_1,x_2)$$

$$\text{cov}(x_2,x_1) = -11.$$

$$\text{cov}(x_2,x_2) = \frac{1}{N-1}\sum_{K=1}^{N}(x_{2i}-\bar{x}_2)(x_{2K}-\bar{x}_2)$$

$$= \frac{1}{3}\left((11-8.5)^2+(4-8.5)^2+(5-8.5)^2+(14-8.5)^2\right)$$

$$\text{cov}(x_2,x_2) = 23.$$

$$S = \begin{bmatrix} \text{cov}(x_1,x_1) & \text{cov}(x_1,x_2) \\ \text{cov}(x_2,x_1) & \text{cov}(x_2,x_2) \end{bmatrix}$$

$$S = \begin{bmatrix} 14 & -11 \\ -11 & 23 \end{bmatrix}.$$

Step-3:- a) Eigenvalues of the covariance matrix.

The characteristic equation of the covariance matrix is :-

$$0 = \det(S - \lambda \mathbb{I})$$

(det: determinate of; 0: zero; S: Covariance Matrix; λ: Lambda; $\mathbb{I}$: Identity Matrix)

$$\mathbb{I} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}_{2\times 2}$$

(no. of feature = 2)

$$0 = \begin{vmatrix} 14-\lambda & -11 \\ -11 & 23-\lambda \end{vmatrix}$$

$$= (14-\lambda)(23-\lambda) - (-11)(-11)$$

$$0 = \lambda^2 - 37\lambda + 201 \longrightarrow \lambda \text{ can be calculated by quadratic eqn.}$$

(a ↓, b ↓, c ↓)

Find roots:-

$$\because x = \frac{-b \pm \sqrt{b^2-4ac}}{2a}$$

$a = 1$

$b = -37$

$c = 201.$

$$\lambda = \frac{1}{2}\left(37 \pm \sqrt{565}\right)$$

λ_1 $\quad$ λ_2

$$= 30.3849,\ 6.6151 \longrightarrow 2 \text{ roots.}$$

$$\lambda_1 > \lambda_2 .$$

$\lambda_1 = 30.3849$ ⟶ Always takes highest eigen value.

$\lambda_2 = 6.6159.$

b) Eigent vectors:-

$$U = \begin{Bmatrix} u_1 \\ u_2 \end{Bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} = (S - \lambda I)\, U$$

S: covariance matrix; I: Identity Matrix; λ: Lambda; U: Eigenvector

$$= \begin{bmatrix} 14-\lambda & -11 \\ -11 & 23-\lambda \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

$$\begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} (14-\lambda)u_2 - 11u_2 \\ -11u_1 + (23-\lambda)u_2 \end{bmatrix}$$

$(14-\lambda)u_1 - 11u_2 = 0 \rightarrow (1).$

$-11u_1 + (23-\lambda)u_2 = 0 \rightarrow (2).$

} Either take eqn (1) & (2) to gent u_1, u_2 values.

$$(14-\lambda)u_1 - 11u_2 = 0.$$

$$(14-\lambda)u_1 = 11u_2$$

$$\frac{u_1}{11} = \frac{u_2}{14-\lambda} = t \quad (\text{let assume two values equal to } t)$$

$$u_1 = 11t,\ u_2 = (14-\lambda)t .$$

$$u = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} 11 \\ 14-\lambda \end{bmatrix}.$$

$$u_1 = \begin{bmatrix} 11 \\ 14-\lambda \end{bmatrix}.$$

To find a unit eigenvectors, we compute the length of V_1 which is given by,

$$\|u_1\| = \sqrt{11^2 + (14-\lambda)^2} \longrightarrow \text{High eigen value}$$

$$= \sqrt{11^2 + (14-30.8349)^2}$$

$$= 19.7348$$

$$e_1 = \begin{bmatrix} 11/\|u_1\| \\ (14-\lambda)/\|u_1\| \end{bmatrix}$$

$$= \begin{bmatrix} 11/19.7388 \\ (14-30.8349)/19.7483 \end{bmatrix}.$$

$$e_1 = \begin{bmatrix} 0.5574 \\ -0.8303 \end{bmatrix}.$$

If we follow all the steps the again we get eigenvector 2

$$e_2 = \begin{bmatrix} 0.8303 \\ 0.5574 \end{bmatrix}.$$

Step-4:- Computation of first principal components:

$$e_1^T \begin{bmatrix} x_1 K - \bar{x_1} \\ x_2 K - \bar{x_2} \end{bmatrix}$$

$$e_1^T \begin{bmatrix} x_{1K} - \bar{x_1} \\ x_{2K} - \bar{x_2} \end{bmatrix} = [0.5574 \quad -0.8303]$$

$$= 0.5573(x_{11} - \bar{x_1}) - 0.8303(x_2 - \bar{x_2})$$

$$= 0.5574(4-8) - 0.8303\ (11-85)$$

$$= -4.03535.$$

Mean

$\bar{x_1} = 8$

$\bar{x_2} = 8.5$

Repeat the steps, then all principal components:-

Feature	Ex_1	Ex_2	Ex_3	Ex_4.
x_1	4	8	13	7
x_2	11	4	5	14
first prin-cipal component	-4.3052	3.7631	5.6928	-5.1238

} we reduce data into 1D data.

Implementation of PCA using scikit-learn:-

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from numpy.linalg import eig
```

(linalg → Linearalgebra; eig → Eigen.)

```
Marks = np.array([[4,11],[8,4],[13,5],[7,14]])
print(marks)
```

⇒ o/p:- [[4 11]
[8 4]
[13 5]
[7 14]]

```
mean by column = np.mean(marks.T, axis=1)
print(mean by column)
```

⇒ o/p:- [8 8.5]

```
Marks.T
```

⇒ o/p:- array([[4, 8, 13, 7],
[11, 4, 5, 14]]).

```
cov_mat = np.cov(marks.T)
cov_mat
```

⇒ o/p:- array([[-14, 1], [1, 23]])

```
Eval, Evec = eig(cov, mat)
```

print (eval) ⇒ o/p:- [6.61515358 30.38486432]

print ("The Eigenvector", Evec) ⇒ o/p:- the Eigenvector

{ [-0.83025082 0.55778887]
[-0.55778887 -0.83025082]. }

From sklearn. decomposition import PCA

Pca = PCA (n_components = 2)

pca. fit_transform (Marks).

PCDFF Pd. Data Frame (data = Pca. fit_transform (marks),

column = ['PC1', 'PC2'])

PCDf.

o/p:-	PC1	PC2
	-4.305187	1.92528
	3.736129	2.508255
	5.692828	-2.200389.
	-5.123769	-2.235394.

Random PCA:-

Randomized PCA is very useful when you want to drastically reduce dimensionality and the dataset fit in memory. In such cases, it works faster than the regular principal components analysis.

⇒ PCA is mostly used for very large datasets with many variables in order to make them clearer and easier to interpret. This can lead to a very high computing power and long waiting times.

⇒ Randomized PCA can be used to reduce the calculation time. For this simply set the parameter svd_solver to 'randomized'.

Implementation:-

```
from sklearn.decomposition. Randomized PCA
rnd_pca = PCA(n_components=200, svd_solver='randomized',
                                random_state=42)
X_reduced = rnd_pca.fit_transform(X_train).
```

Note:- By default svd_solver ~~is~~ actually set to "two" scikit-learn automatically uses the randomized PCA algorithm.

"If you set the svd_solver hyperparameter to "randomized", the scikit-learn uses a stochastic algorithm called randomized PCA that quickly finds an approximation of the n components.

Kernel PCA:-

⇒ Used for non-linearly seperable class problems.

⇒ Used a kernel function to project the dataset into a higher dimensional space, where it is linearly separable.

⇒ Kernal PCA is an extension of PCA using techniques of "Kernel Methods".

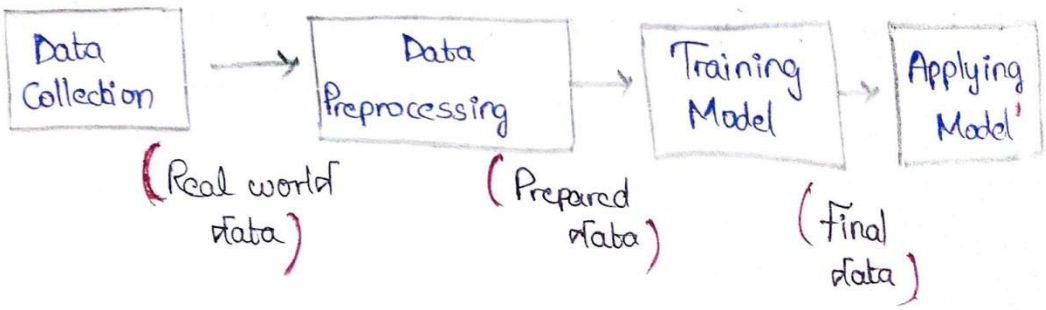
Implementation:-

```
from sklearn.decomposition import KernelPCA
KPCA = KernelPCA(kernel = 'rbf', gamma=15)
x_pca = kpca.fit_transform(x).
```

Using clustering for Preprocessing:-

⇒ Clustering can be efficient approach to dimensionality reduction, in particular preprocessing step before a unsupervised learning algorithm.

⇒ Real world data usually is noisy incomplete could be stored in different places & different formats. In common ML Pipeline, Data Preprocessing stage is between Data collection stage and Training/Tunning Model.



```
# Lets tackle digits dataset which is simple MNIST
like dataset containing 1797 8x8 digits 0 to 9.

from sklearn.clusters import KMeans.

from sklearn.datasets import load_digits.

# Initially we load digits dataset.

X_digits, y_digits = load_digits (return_X_y = True)

# now split digits dataset training set & test set:

from sklearn.model_selection import train_test_split

X_train, y_train, y_test, X_test = train_test_split ( X_digts, y_digits),

#Let's fit a Logistic Regression Model for prediction of
# digits.

from sklearn.linear_model import Logistic Regression.

lg_reg = Logistic Regression (random_state = 42)

log_reg = fit (X_train, y_train)

# Let's evaluate its accuracy on test set;

print (log_reg.score (X_test, y_test)).

# 0.96666667.
```

* Now we can do better by using Kmeans as preproce-
-ssing step.
* For that we create a pipeline that will first cluster the training set into 50 clusters and replace image with the distances to 50 clusters, then apply logistic regression model.

```
from sklearn.pipeline import Pipeline
pipeline = pipeline ( [ ("kmeans", Kmeans (n_clusters = 50)),
                        ("log_reg", Logistic Regression ()), ])
pipeline.fit (x_train, y_train)
print (pipline.score (x_test, y_test)) # evaluate pipeline classification.
```

* Upto now no. of clusters K completely arbitarily, we can surely do better than by using Grid Search CV which results in the best classification performance during class validation. It is to find the optimal no. of clusters.

```
from sklearn.model_selection import GridSearch CV
Poolroom_grid = dict(kmean_n_clusters = range (2,100))
grid_cf = Grid Search CV (pipeline, param_grid, CV = 3, verboose=2)
grid_cf . fit (x_train, y_train)
print ( grid_cf.best_params.. ) # {'kmeans_n_clusters : 90}
print (grid_cf.score (x_test, y_test))
```

Output:-

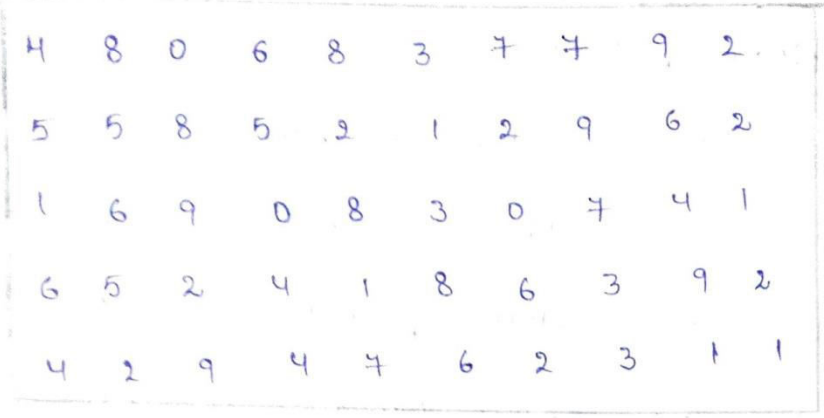
```
# 0.98445 [# with K = 90 clusters accuracy 98.44%.]
```

Using Clustering for Semi Supervised Learning:-

⇒ Semi Supervised Learning is a type of machine learning that uses both labels and unlabeled data to train a model. Clustering is a technique that groups similar data points together based on same measures of similarity or distance.

* Lets train a logistic regression model on a sample of 50 labeled instances from the digit dataset.

Dataset shows 50 representative images:-



Program:-

```
import numpy as np

from sklearn.datasets import load_digits

x_digits, y_digits = load_digits ( return_x_y = True)

from sklearn.model_selection import train_test_spit

x_train, x_test, y_train, y_test = train_test_split (x_digits, y_digits).

from sklearn.clusters import Kmeans

# fit the Logistic Regression

from sklearn.linear_model import Logistic Regression

log_reg = Logistic Regression ( random_state = 42)

log_reg.fit (x_train, y_train)

# Evaluate its accuracy on the test set

log_reg_score (x_test, y_test)

# Output:- 0.9666666.
```

Now look at each image & manually label it from above take dataset.

```
K = 50          # fifty representative digit images (one per cluster)
kmeans = KMeans(n_clusters = k)
x_digits_dist = kmeans.fit_transform(x_train)
representative_digit_idx = np.argmin(x_digits_dist, axis = 0)
x_representative_digits = x_train[representative_digit_idx]
y_representative_digits = np.array([4,8,0,.....,3,1,7])
```

images taken in array from above dataset.

```
log_reg = LogisticRegression()
log_reg.fit(x_representative_digits, y_representative_digits)
log_reg.score(x_test, y_test)
```

Output:- 0.92444....

* We get accuracy 92.4%, although we are still only training the 50 models.