Data Mining

✓ Topics

- **Introduction:**
    - Definition of Data Mining
    - Application of Data Mining
- **Clustering:**
    - Definition and Examples
    - Clustering Algorithms
    - K Means Method
        - Optimal Number of cluster
            - Elbow method
            - Average Silhouette Method
            - Gap Statistic Method
        - Assigning points to clusters
        - Updating centroids
        - Repeat until convergence
    - ANN Math Problem
    - Fuzzy C-Means Clustering
        - Tutorial Question-49
    - **K Medoids Clustring**
    - **K Prototype Clustering**
        - **Tutorial-48**
- **Tutorial Solve:**

Introduction:

✠ Definition of Data Mining

Data Mining

Data Mining is an interdisciplinary subfield of computer science,is the computational process of discovering patters in large data sets,involving methods at the intersection of artificial intelligence,machine learning,statistics and database system.

Data mining is a process of extracting meaningful information from vast amount of data.With data mining methods,organizations can discover hidden patterns, relationships, and trends in data,which they can used to solve business problem,make predictions and increase their profits on efficiency.

- The overall goal of data mining process is to extract information from a data set and transform it into an understandable structure for further use

- Data mining is often called as finding hidden information in a database. Alternatively it has called exploratory data analysis,data driven discovery and deductive learning.

✠ Application of data mining

Applications:

- Retailer use data mining to analyze large data sets and discover consumers buying patterns such as items that are frequently bought together or seasonal trends.They can use this information to better organize their physical stones and websites,predict sales and promite deals.

- Educational data mining aims to improve learning by analysing a variety of educational data,such as students' interactions with online learning environments on administrative data from schools and universities.This method can help education providers understand what student need and support them better.

Clustering

✠ Definition and exmples

Clustering is the process of grouping a set of data objects into multiple groups based on similarity, such that objects within the same cluster have high similarity, but are very dissimilar to objects in other clusters.

Clustering is a data analysis technique where a set of data point is devided into groups or clusters based on based on their similarities,with the aim of ensuring that object within each clusters are similar to each other but dissimilar in object to other clusters.This process known as cluster analysis,involves partitioning the data into subsets or clusters,each representing a distinct group of similar objects.The result cluster can reveal previously unknown patterns or groups within the data,making clustering a valuable tool for data exploration and discovery.

Cluster analysis finds extensive application accross various domains including business intelligence,image patterns recognition,web search,biology,security or more

✠ Clustering Algorithms

Most of the commonly used clustering algorithms can be classified into two general categories.

- ▲ **Hierarchical clustering**
  - ◇ Agglomerative Hierarchical Clustering
    - ➜ Simple Linkage(Nearest Neighbour)
    - ➜ Complete Linkage Clustering (Farthest Neighbor Method)
    - ➜ Average Linkage Clustering(Minimal Spanning Tree)
    - ➜ Centroid Linkage
    - ➜ Median Linkage
    - ➜ Ward's Linkage
  - ◇ Divisive Clustering Method (Top-Down Approach)
    - ➜ Divisible (Divisive) Analysis

- ➜ Monothetic Analysis

- ▲ Non Hierarchical Clustering
  - ◇ K means
  - ◇ K modes
  - ◇ K prototypes

✠ K means method

Mac Queen suggests the term k means for describing an algorithm that assigns each item to the cluster having the nearest centroid.
k means clustering algorithm consists of following steps:

- ▲ Step-1: **Partitioning the items into initial clusters**
- ▲ Step-2:**Proceed though the list of items,assigning the item to the cluster whose centroid(mean) is nearest.Recalculate the centroid for the cluster receiving the new item and for the cluster losing item**
- ▲ Step-3:**Repeat step-2 until no more reasgnment take place**

➤ Math Problem:

Suppose the data mining task is to cluster point into three clusters.Where the points are $A_1(2,10), A_2(2,5), A_3(8,4), B_1(5,8), B_2(7,5), B_3(6,4), C_1(1,2), C_2(4,9)$

Tasks:

1. Assign each point to the nearest cluster based on Euclidean distance.
2. Compute the new centroids of the clusters.
3. Repeat the assignment step using the new centroids (2nd iteration).
4. Show the calculation steps clearly.

➤ Solution: YouTube video link:Click me: ✈

Data Points			Distance to						Cluster	New Cluster
	(x_1)	(y_1)	(x_2) 2	(y_2) 10	5	8	1	2		
A_1	2	10	0		3.61		8.06		1	
A_2	2	5	5		4.24		3.16		3	
A_3	8	4	8.49		5		7.28		2	
B_1	5	8	3.61		0		7.21		2	
B_2	7	5	7.07		3.61		6.71		2	
B_3	6	4	7.21		4.12		5.39		2	
C_1	1	2	8.06		7.21		0		3	
C_2	4	9	2.24		1.41		7.62		2	

Initial centroid:

$A_1 = (2, 10)$
$B_1 = (5, 8)$
$C_1 = (1, 2)$

$$\begin{aligned} d(p_1, p_2) &= \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \\ d(p_{(2,10)}, p_{(2,10)}) &= \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} = 0 \\ d(p_{(2,5)}, p_{(2,10)}) &= \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} = 5 \\ d(p_{(8,4)}, p_{(2,10)}) &= \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} = 8.49 \\ &\ldots\ldots\ldots\ldots\ldots \\ &\ldots\ldots\ldots\ldots\ldots \\ &\ldots\ldots\ldots\ldots\ldots \end{aligned}$$

New centroid:

$A_1 = (2, 10)$
$B_1 = (6, 6)$ Explanation in YouTube video link:Click me: ✈
$C_1 = (1.5, 3.5)$

So,Current centroid:

$A_1 = (2, 10)$

$B_1 = (6, 6)$ Explanation in YouTube video link:Click me: ✈
$C_1 = (1.5, 3.5)$

Data Points			Distance to						Cluster	New Cluster
			(x_2) 2	10 (y_2)	6	6	1.5	3.5		
	(x_1)	(y_1)								
A_1	2	10	0		5.06		6.52		1	1
A_2	2	5	5		4.12		1.58		3	3
A_3	8	4	8.49		2.83		6.52		2	2
B_1	5	8	3.61		2.24		5.7		2	2
B_2	7	5	7.07		1.41		5.7		2	2
B_3	6	4	7.21		2		4.53		2	2
C_1	1	2	8.06		6.40		1.58		3	3
C_2	4	9	2.24		3.61		6.04		2	1

New centroid:

$A_1 = (3, 9.5)$
$B_1 = (6.5, 5.25)$ Explanation in YouTube video link:Click me: ✈
$C_1 = (1.5, 3.5)$

Data Points			Distance to			Cluster	New Cluster
	(x_1)	(y_1)	(x_2) 3 9.5 (y_2)	6.5 5.25	1.5 3.5		
A_1	2	10	1.12	6.54	6.52	1	
A_2	2	5	4.61	4.51	1.58	3	
A_3	8	4	7.43	1.95	6.52	2	
B_1	5	8	2.5	3.13	5.7	1	
B_2	7	5	6.02	0.5641	5.7	2	
B_3	6	4	6.25	1.35	4.53	2	
C_1	1	2	7.76	6.39	1.58	3	
C_2	4	9	1.12	4.51	6.04	1	

New centroid:

$A_1 = (3.67, 9)$
$B_1 = (7, 4.33)$ Explanation in YouTube video link:Click me: ✈
$C_1 = (1.5, 3.5)$

Data Points			Distance to			Cluster	New Cluster
	(x_1)	(y_1)	(x_2) 3.67 | (y_2) 9	7 | 4.33	1.5 | 3.5		
A_1	2	10	1.94	7.56	6.52	1	1
A_2	2	5	4.33	5.04	1.58	3	3
A_3	8	4	6.62	1.05	6.52	2	2
B_1	5	8	1.67	4.18	5.7	1	1
B_2	7	5	5.21	0.67	5.7	2	2
B_3	6	4	5.52	1.05	4.53	2	2
C_1	1	2	7.49	6.44	1.58	3	3
C_2	4	9	0.33	5.55	6.04	1	1

So,First Cluster:$A_1(2,10)$**,**$B_1(5,8)$**,**$C_2(4,9)$
So,Second Cluster:$A_3(8,4)$,$\mathbf{B}_2(7,5)$,$\mathbf{B}_3(6,4)$
So,Third Cluster:$A_2(2,5), C_1(1,2)$

Python Code:K Means Clustering Using Numpy,sklearn,Matplotlib

```
import pandas as pd
data=pd.read_csv("Student Attitude and Behavior.csv")
data.head(1)
PCA_1=data['college mark']
PCA_2=data['12th Mark']

import numpy as np
x=np.column_stack((PCA_1,PCA_2))
x
#print(x)

from sklearn.cluster import KMeans
k_means=KMeans(n_clusters=2,random_state=0)
k_means

labels=k_means.fit_predict(x)
labels

data['cluster']=labels
data.head(1)

import matplotlib.pyplot as plt
fsize=plt.figure(figsize=(8,6))
fsize
plt.scatter(x[:,0],x[:,1],c=labels, cmap='viridis', s=50)

plt.xlabel('College Mark')
plt.ylabel('12th Mark')
plt.title('K-Means Clustering based on College & 12th Marks')
plt.grid(True)
plt.colorbar(label='Cluster')
plt.show()
```

K means clustering manual(basic) python code part-1

```
import pandas as pd
data=pd.read_csv("Student Attitude and Behavior.csv")
data.head(1)

pca_1=data['college mark']
pca_2=data['12th Mark']
pca_1=list(pca_1)
pca_1=list(pca_1)

initial_first_centroid=[pca_1[10],pca_2[10]]
print('initial_first_centroid:',initial_first_centroid)
initial_second_centroid=[pca_1[21],pca_2[21]]
print('initial_second_centroid:',initial_second_centroid)
initial_third_centroid=[pca_1[78],pca_2[78]]
print('initial_third_centroid',initial_third_centroid)

distance1=[]
distance2=[]
distance3=[]
for i in range(len(pca_1)):
    a=pca_1[i]
    #print(a)
    b=pca_2[i]
    #print(b)
    c=initial_first_centroid[0]
    d=initial_first_centroid[1]
    dist1=math.sqrt((a-c)**2+(b-d)**2)
    dist1=round(dist,2)
    distance1.append(dist1)
    e=initial_second_centroid[0]
    f=initial_second_centroid[1]
    dist2=math.sqrt((a-e)**2+(b-f)**2)
    dist2=round(dist2,2)
    distance2.append(dist2)
    g=initial_third_centroid[0]
    h=initial_third_centroid[1]
    dist3=math.sqrt((a-g)**2+(b-h)**2)
    dist3=round(dist3,2)
    distance3.append(dist3)
print('distance1:',distance1)
print('distance2:',distance2)
print('distance3:',distance3)
```

K means clustering manual(basic) python code part-2

```
cluster=[]
for i in range(len(pca_1)):
    x=min(distance1[i],distance2[i],distance3[i])
    if x in distance1:
        cluster.append(1)
    elif x in distance2:
        cluster.append(2)
    else:
        cluster.append(3)
print(cluster)
print(len(cluster))

index1=[]
index2=[]
index3=[]
for i in range(len(cluster)):
    if cluster[i]==1:
        index1.append(i)
    elif cluster[i]==2:
        index2.append(i)
    else:
        index3.append(i)
print(index1)
print(index2)
print(index3)

a=0
b=0
for i in index1:
    a=a+pca_1[i]
    b=b+pca_2[i]
print(a)
new_first_cen_x=round(a/len(index1),2)
print(new_first_cen_x)

new_first_cen_y=round(b/len(index1),2)
print(new_first_cen_y)

new_first_centroid=[new_first_cen_x,new_first_cen_y]
print('new_first_centroid:',new_first_centroid)
```

K means clustering manual(basic) python code part-3

```
a=0
b=0
for i in index3:
    a=a+pca_1[i]
    b=b+pca_2[i]
print(a)
new_third_cen_x=round(a/len(index3),2)
print(new_third_cen_x)

new_third_cen_y=round(b/len(index3),2)
print(new_third_cen_y)

new_third_centroid=[new_third_cen_x,new_third_cen_y]
print('new_third_centroid:',new_third_centroid)

distance1_new=[]
distance2_new=[]
distance3_new=[]
for i in range(len(pca_1)):
    a=pca_1[i]
    #print(a)
    b=pca_2[i]
    #print(b)
    c=new_first_centroid[0]
    d=new_first_centroid[1]
    dist1=math.sqrt((a-c)**2+(b-d)**2)
    dist1=round(dist1,2)
    distance1_new.append(dist1)

    e=new_second_centroid[0]
    f=new_second_centroid[1]
    dist2=math.sqrt((a-e)**2+(b-f)**2)
    dist2=round(dist2,2)
    distance2_new.append(dist2)

    g=new_third_centroid[0]
    h=new_third_centroid[1]
    dist3=math.sqrt((a-g)**2+(b-h)**2)
    dist3=round(dist3,2)
    distance3_new.append(dist3)
print('distance1_new:',distance1_new)
print('distance2_new:',distance2_new)
print('distance3_new:',distance3_new)
```

K means clustering manual(basic) python code part-4

```
cluster2=[]
for i in range(len(pca_1)):
    x=min(distance1_new[i],distance2_new[i],distance3_new[i])
    if x in distance1_new:
        cluster2.append(1)
    elif x in distance2_new:
        cluster2.append(2)
    else:
        cluster2.append(3)
print(cluster2)
print(len(cluster2))
```

Optimal Number of cluster in k means clustering

The obtimal clustering is somehow subjective and depends on the method used for measuring similarities and the parameters used for partitioning.In this present section,describe one method from different methods for determing the obtimal number of clusters for k means clustering.These methods includes direct methods and statistical testing method.Three popular methods for determing the optimal number of clusters.

1. Elbow Method
2. Average Silhouette Method
3. Gap Statistic Method

The Elbow Method

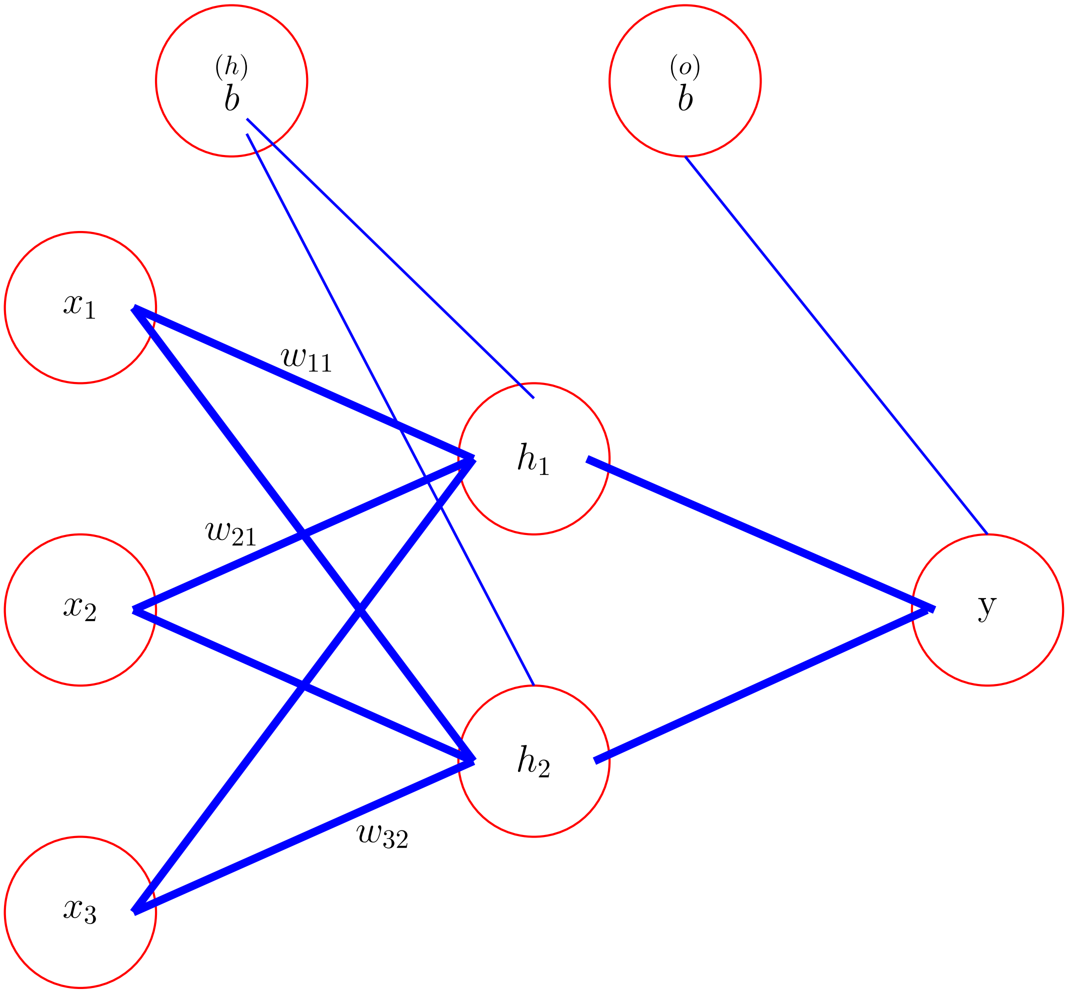
Youtube Video Click me

ANN Math Problem

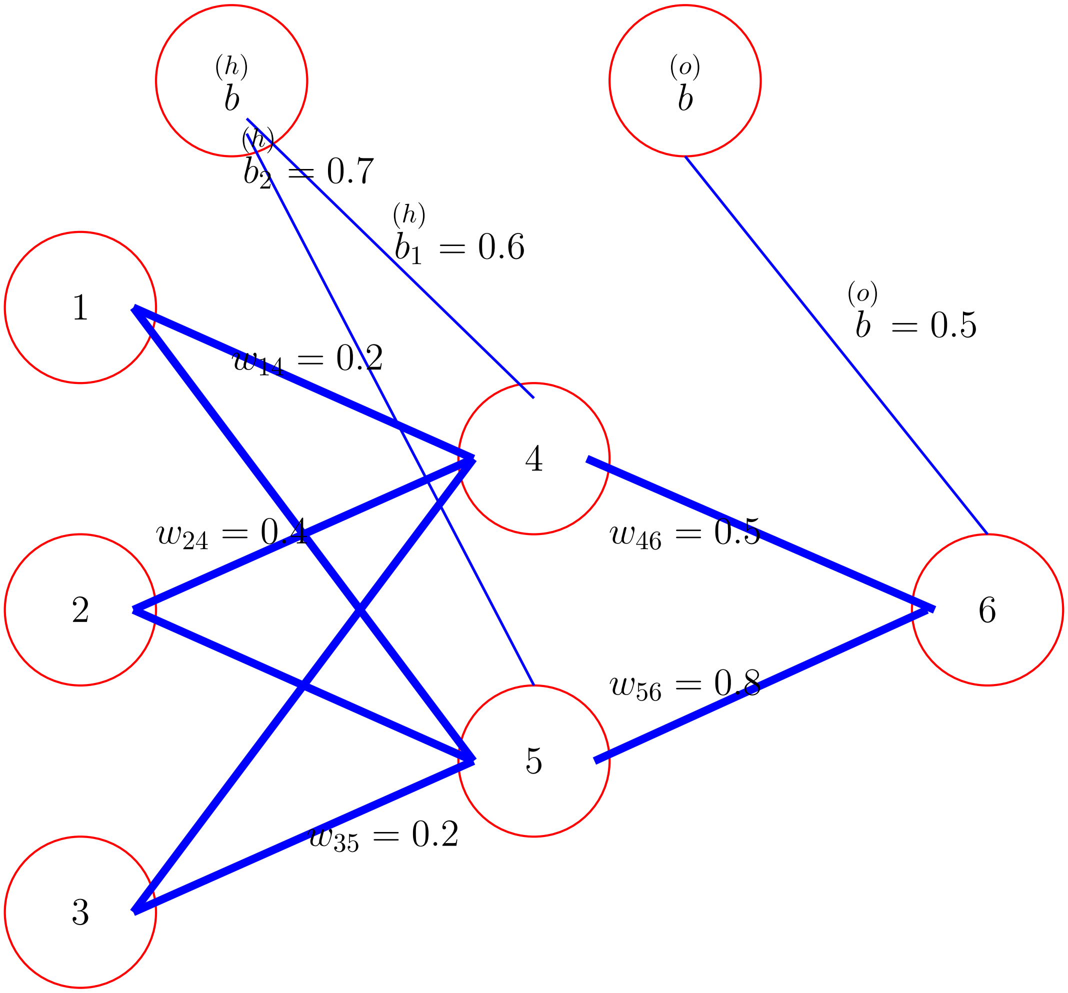
Math Problem:

The inputs of ANN are $x_1 = 0.5, x_2 = 0.8, x_3 = 0.7$**;The weights are** $w_{11}^{(h)} = 0.2, w_{12}^{(h)} = 0.7, w_{21}^{(h)} = 0.4, w_{22}^{(h)} = 0.6, w_{31}^{(h)} = 0.7, w_{32}^{(h)} =$

$0.2, \overset{(o)}{w}_1 = 0.5, \overset{(o)}{w}_2 = 0.8$ **; and the bais are** $\overset{(h)}{b}_1 = 0.6, \overset{(h)}{b}_2 = 0.7, \overset{(o)}{b} = 0.4$**,learning rate 0.9,Target** $T_j = 1$**.Obtain the updated weight after the first iteration.**



➤ Solution:



$x_1 = O_1 = 0.5$ $\quad O_4 = h_1$ $\quad w_{11} = w_{14} = 0.2$

$x_2 = O_2 = 0.8$ $\quad O_5 = h_2$ $\quad w_{21} = w_{24} = 0.4$

$x_3 = O_3 = 0.7$ $\quad O_6 = y$ $\quad w_{31} = w_{34} = 0.7$

$\overset{(h)}{b}_1 = 0.6$ $\quad w_{12} = w_{15} = 0.7$

$\overset{(h)}{b}_2 = 0.7$ $\quad w_{22} = w_{25} = 0.6$ $\quad w_1 = w_{46} = 0.5$

$\overset{(o)}{b} = 0.4$ $\quad w_{32} = w_{35} = 0.2$

learning rate,$l = 0.9$ $\quad w_2 = w_{56} = 0.8$

$T_j = 1$

$$I_4 = O_1 w_{14} + O_2 w_{24} + O_3 w_{34} + \overset{(h)}{b}_1$$

$$I_4 = 0.5 \times 0.2 + 0.8 \times 0.4 + 0.7 \times 0.7 + 0.6$$

$$I_4 = 1.51$$

$$O_4 = \frac{1}{1 + \mathrm{e}^{-I_4}}$$

$$O_4 = \frac{1}{1 + \mathrm{e}^{-1.51}}$$

$$O_4 = 0.819$$

$$I_5 = O_1 w_{15} + O_2 w_{25} + O_3 w_{35} + \overset{(h)}{b}_1$$

$$I_5 = 0.7 \times 0.5 + 0.6 \times 0.8 + 0.2 \times 0.7 + 0.7$$

$$I_5 = 1.67$$

$$O_5 = \frac{1}{1 + \mathrm{e}^{-I_5}}$$

$$O_5 = \frac{1}{1 + \mathrm{e}^{-1.67}}$$

$$O_5 = 0.842$$

$$I_6 = O_4 w_{46} + O_5 w_{56} + \overset{(o)}{b}$$

$$I_6 = 0.819 \times 0.5 + 0.842 \times 0.8 + 0.6$$

$$I_6 = 1.683 \qquad \text{Or,}1.483$$

$$O_6 = \frac{1}{1 + \mathrm{e}^{-I_6}}$$

$$O_6 = \frac{1}{1 + \mathrm{e}^{-1.683}}$$

$$O_6 = 0.843 \qquad \text{Or,}0.815$$

$$\begin{aligned}
E_{rr_6} &= O_6(1-O_6)(T_j-O_6)\\
&= 0.843\times(1-0.843)\times(1-0.843)\\
&= 0.0207
\end{aligned}$$

$$\begin{aligned}
w_{46_{new}} &= w_{46_{old}} + l\times E_6\times O_4\\
&= (0.5+0.9\times0.021\times0.819)\\
&=
\end{aligned}$$

$$\begin{aligned}
E_4 &= O_4\times(1-O_4)\times E_6\times w_{46_{new}}\\
E_4 &= 0.819\times(1-0.819)\times\\
&=
\end{aligned}$$

$$\begin{aligned}
w_{56_{new}} &= w_{56_{old}} + lE_6O_5\\
w_{56_{new}} &= 0.8+0.9\times0.021\times0.841\\
w_{56_{new}} &= 0.816
\end{aligned}$$

$$\begin{aligned}
E_5 &= O_5(1-O_5)E_6w_{56_{new}}\\
E_5 &= 0.841\times(1-0.841)\times0.021\times0.816\\
E_5 &= 0.003
\end{aligned}$$

$$\begin{aligned}
w_{14_{new}} &= w_{14_{old}} + lE_4O_1\\
w_{14_{new}} &= 0.2+0.9\times1.6\times10^{-3}\times0.5\\
w_{14_{new}} &= 0.2001
\end{aligned}$$

$$\begin{aligned}
w_{24_{new}} &= w_{24_{old}} + lE_4O_2\\
w_{24_{new}} &= 0.4+0.9\times1.6\times10^{-3}\times0.8\\
w_{24_{new}} &= 0.401
\end{aligned}$$

$$
\begin{aligned}
w34_{new} &= w34_{old} + lE_4O_3 \\
w34_{new} &= 0.7 + 0.9 \times 1.6 \times 10^{-3} \times 0.7 \\
w34_{new} &= 0.701
\end{aligned}
$$

$$
\begin{aligned}
w15_{new} &= w14_{old} + lE_5O_1 \\
w15_{new} &= 0.7 + 0.9 \times 2.29 \times 10^{-3} \times 0.5 \\
w15_{new} &= 0.701
\end{aligned}
$$

$$
\begin{aligned}
w25_{new} &= w25_{old} + lE_5O_2 \\
w25_{new} &= 0.6 + 0.9 \times 2.29 \times 10^{-3} \times 0.8 \\
w25_{new} &= 0.601
\end{aligned}
$$

$$
\begin{aligned}
w35_{new} &= w35_{old} + lE_5O_3 \\
w35_{new} &= 0.2 + 0.9 \times 2.29 \times 10^{-3} \times 0.7 \\
w35_{new} &= 0.201
\end{aligned}
$$

✠ Fuzzy C-Means Clustering

YouTube Video: Click me: ✈

Clustering is a distance-based unsupervised machine learning algorithm where data points that are close to each other are grouped in a given number of clusters/groups
There are basically two types of clustering algorithm.

1. **Hard Clustering**
   - ◇ **K-Means**
   - ◇ **K-Medoids (PAM)**
   - ◇ **Hierarchical Clustering (Agglomerative or Divisive) etc.**
2. **Soft Clustering**
   - ◇ **Fuzzy C-Means (FCM)**

- Gaussian Mixture Models (GMM)
- Expectation-Maximization (EM) etc.

♦The Steps of Fuzzy C-Means Clustering

- **Step-01:** Given the data points based on the number of cluster required initialize the membership table with the random values.
  Suppose the given data points are{(1,3),(2,5),(4,8),(7,9)}

Cluster	(1,3)	(2,5)	(4,8)	(7,9)
1	0.8	0.7	0.2	0.1
2	0.2	0.3	0.8	0.9

- **Step-2:** Find Out The Centroid.The formula of finding out V is:

$$V_{ij} = \frac{\sum_{k=1}^{4} \gamma_{ik}^{m} \cdot x_k}{\sum_{k=1}^{n} \gamma_{ik}^{m}}$$

Where,

γ = Fuzzy Membership Value

m = Fuzziness Parameter Generally Taken as two(2)

x_k = is the data point

- **Step-3:** Find Out the distance of each point from the centroid
- **Step-4:** Updating Membership values

$$\gamma_{ki} = \left(\sum_{j=1}^{n} \left(\frac{d_{ki}^2}{d_{kj}^2} \right)^{\frac{1}{m-1}} \right)^{-1}$$

- **Step-5:** Repeat The steps (2-4) until the constant values are obtained for the membership values or the difference is less than the tolerance value

➤ **Tutorial Question-49:** Given that,A(1,2),B(1,3),C(4,3),D(5,4),E(8,6) and F(8,7) be the six points and the initial clusters are $C_1 = (1.5, 2.5), C_2 = (4.5, 4.5)$ and $C_3 = (8.5, 6.5)$.Apply the fuzzy C-Means Clustering Algorithm with the maximum number of iterations 4 to find the cluster solution and find the fuzzy partitioning coefficient.

➤ **Solution:**

Cluster	**(1,2)**	**1,3**	**(4,3)**	**5,4**	**(8,6)**	**(8,7)**
1						
2						
3						

Table 1: Caption

Centroids are,

$$\boldsymbol{C_1 = (1.5, 2.5)}$$
$$\boldsymbol{C_2 = (4.5, 4.5)}$$
$$\boldsymbol{C_3 = (8.5, 6.5)}$$

$$d_{11} = \sqrt{(1-1.5)^2 + (2-2.5)^2} = \sqrt{0.25+0.25} = \sqrt{0.5} \approx 0.7071$$
$$d_{21} = \sqrt{(1-4.5)^2 + (2-4.5)^2} = \sqrt{12.25+6.25} = \sqrt{18.5} \approx 4.3012$$
$$d_{31} = \sqrt{(1-8.5)^2 + (2-6.5)^2} = \sqrt{56.25+20.25} = \sqrt{76.5} \approx 8.7464$$
$$d_{12} = \sqrt{(1-1.5)^2 + (3-2.5)^2} = \sqrt{0.25+0.25} = \sqrt{0.5} \approx 0.7071$$
$$d_{22} = \sqrt{(1-4.5)^2 + (3-4.5)^2} = \sqrt{12.25+2.25} = \sqrt{14.5} \approx 3.8079$$
$$d_{32} = \sqrt{(1-8.5)^2 + (3-6.5)^2} = \sqrt{56.25+12.25} = \sqrt{68.5} \approx 8.2765$$
$$d_{13} = \sqrt{(4-1.5)^2 + (3-2.5)^2} = \sqrt{6.25+0.25} = \sqrt{6.5} \approx 2.5495$$
$$d_{23} = \sqrt{(4-4.5)^2 + (3-4.5)^2} = \sqrt{0.25+2.25} = \sqrt{2.5} \approx 1.5811$$
$$d_{33} = \sqrt{(4-8.5)^2 + (3-6.5)^2} = \sqrt{20.25+12.25} = \sqrt{32.5} \approx 5.7009$$
$$d_{14} = \sqrt{(5-1.5)^2 + (4-2.5)^2} = \sqrt{12.25+2.25} = \sqrt{14.5} \approx 3.8079$$
$$d_{24} = \sqrt{(5-4.5)^2 + (4-4.5)^2} = \sqrt{0.25+0.25} = \sqrt{0.5} \approx 0.7071$$
$$d_{34} = \sqrt{(5-8.5)^2 + (4-6.5)^2} = \sqrt{12.25+6.25} = \sqrt{18.5} \approx 4.3012$$
$$d_{15} = \sqrt{(8-1.5)^2 + (6-2.5)^2} = \sqrt{42.25+12.25} = \sqrt{54.5} \approx 7.3824$$
$$d_{25} = \sqrt{(8-4.5)^2 + (6-4.5)^2} = \sqrt{12.25+2.25} = \sqrt{14.5} \approx 3.8079$$
$$d_{35} = \sqrt{(8-8.5)^2 + (6-6.5)^2} = \sqrt{0.25+0.25} = \sqrt{0.5} \approx 0.7071$$
$$d_{16} = \sqrt{(8-1.5)^2 + (7-2.5)^2} = \sqrt{42.25+20.25} = \sqrt{62.5} \approx 7.9057$$
$$d_{26} = \sqrt{(8-4.5)^2 + (7-4.5)^2} = \sqrt{12.25+6.25} = \sqrt{18.5} \approx 4.3012$$
$$d_{36} = \sqrt{(8-8.5)^2 + (7-6.5)^2} = \sqrt{0.25+0.25} = \sqrt{0.5} \approx 0.7071$$

$\mathbf{min}(d_{11}, d_{21}, d_{31})\mathbf{=0.7071}(d_{11})$
$\mathbf{min}(d_{12}, d_{22}, d_{32})\mathbf{=0.7071}(d_{12})$
$\mathbf{min}(d_{13}, d_{23}, d_{33})\mathbf{=1.5811}(d_{23})$
$\mathbf{min}(d_{14}, d_{24}, d_{34})\mathbf{=0.7071}(d_{24})$
$\mathbf{min}(d_{15}, d_{25}, d_{35})\mathbf{=0.7071}(d_{35})$
$\mathbf{min}(d_{16}, d_{26}, d_{36})\mathbf{=0.7071}(d_{36})$

Cluster	**(1,2)**	**1,3**	**(4,3)**	**5,4**	**(8,6)**	**(8,7)**
1						
2						
3						
	1	1	2	2	3	3

Table 2: Caption

$$\gamma_{11} = \left(\frac{d_{11}^2}{d_{11}^2} + \frac{d_{11}^2}{d_{21}^2} + \frac{d_{11}^2}{d_{31}^2}\right)^{-1} = \left(\frac{0.5}{0.5} + \frac{0.5}{18.5} + \frac{0.5}{76.5}\right)^{-1} = (1 + 0.0270 + 0.0065)^{-1} = \boxed{0.9675}$$

$$\gamma_{21} = \left(\left(\frac{d_{21}^2}{d_{11}^2} + \frac{d_{21}^2}{d_{21}^2} + \frac{d_{21}^2}{d_{31}^2}\right)\right)^{-1} = \left(\frac{18.5}{0.5} + \frac{18.5}{18.5} + \frac{18.5}{76.5}\right)^{-1} = (37 + 1 + 0.2418)^{-1} = \boxed{0.0261}$$

$$\gamma_{31} = \left(\left(\frac{d_{31}^2}{d_{11}^2} + \frac{d_{31}^2}{d_{21}^2} + \frac{d_{31}^2}{d_{31}^2}\right)\right)^{-1} = \left(\frac{76.5}{0.5} + \frac{76.5}{18.5} + \frac{76.5}{76.5}\right)^{-1} = (153 + 4.1351 + 1)^{-1} = \boxed{0.0063}$$

$$\gamma_{12} = \left(\left(\frac{d_{12}^2}{d_{12}^2} + \frac{d_{12}^2}{d_{22}^2} + \frac{d_{12}^2}{d_{32}^2}\right)\right)^{-1} = \left(\frac{0.5}{0.5} + \frac{0.5}{14.5} + \frac{0.5}{68.5}\right)^{-1} = (1 + 0.0345 + 0.0073)^{-1} = \boxed{0.9599}$$

$$\gamma_{22} = \left(\left(\frac{d_{22}^2}{d_{12}^2} + \frac{d_{22}^2}{d_{22}^2} + \frac{d_{22}^2}{d_{32}^2}\right)\right)^{-1} = \left(\frac{14.5}{0.5} + \frac{14.5}{14.5} + \frac{14.5}{68.5}\right)^{-1} = (29 + 1 + 0.2117)^{-1} = \boxed{0.0331}$$

$$\gamma_{32} = \left(\left(\frac{d_{32}^2}{d_{12}^2} + \frac{d_{32}^2}{d_{22}^2} + \frac{d_{32}^2}{d_{32}^2}\right)\right)^{-1} = \left(\frac{68.5}{0.5} + \frac{68.5}{14.5} + \frac{68.5}{68.5}\right)^{-1} = (137 + 4.7241 + 1)^{-1} = \boxed{0.0070}$$

$$\gamma_{13} = \left(\left(\frac{d_{13}^2}{d_{13}^2} + \frac{d_{13}^2}{d_{23}^2} + \frac{d_{13}^2}{d_{33}^2}\right)\right)^{-1} = \left(\frac{6.5}{6.5} + \frac{6.5}{2.5} + \frac{6.5}{32.5}\right)^{-1} = (1 + 2.6 + 0.2)^{-1} = \boxed{0.2632}$$

$$\gamma_{23} = \left(\left(\frac{d_{23}^2}{d_{13}^2} + \frac{d_{23}^2}{d_{23}^2} + \frac{d_{23}^2}{d_{33}^2}\right)\right)^{-1} = \left(\frac{2.5}{6.5} + \frac{2.5}{2.5} + \frac{2.5}{32.5}\right)^{-1} = (0.3846 + 1 + 0.0769)^{-1} = \boxed{0.6842}$$

$$\gamma_{33} = \left(\left(\frac{d_{33}^2}{d_{13}^2} + \frac{d_{33}^2}{d_{23}^2} + \frac{d_{33}^2}{d_{33}^2}\right)\right)^{-1} = \left(\frac{32.5}{6.5} + \frac{32.5}{2.5} + \frac{32.5}{32.5}\right)^{-1} = (5 + 13 + 1)^{-1} = \boxed{0.0526}$$

$$\gamma_{14} = \left(\left(\frac{d_{14}^2}{d_{14}^2} + \frac{d_{14}^2}{d_{24}^2} + \frac{d_{14}^2}{d_{34}^2}\right)\right)^{-1} = \left(\frac{14.5}{14.5} + \frac{14.5}{0.5} + \frac{14.5}{18.5}\right)^{-1} = (1 + 29 + 0.7838)^{-1} = \boxed{0.0325}$$

$$\gamma_{24} = \left(\left(\frac{d_{24}^2}{d_{14}^2} + \frac{d_{24}^2}{d_{24}^2} + \frac{d_{24}^2}{d_{34}^2}\right)\right)^{-1} = \left(\frac{0.5}{14.5} + \frac{0.5}{0.5} + \frac{0.5}{18.5}\right)^{-1} = (0.0345 + 1 + 0.0270)^{-1} = \boxed{0.9421}$$

$$\gamma_{34} = \left(\left(\frac{d_{34}^2}{d_{14}^2} + \frac{d_{34}^2}{d_{24}^2} + \frac{d_{34}^2}{d_{34}^2}\right)\right)^{-1} = \left(\frac{18.5}{14.5} + \frac{18.5}{0.5} + \frac{18.5}{18.5}\right)^{-1} = (1.276 + 37 + 1)^{-1} = \boxed{0.0255}$$

$$\gamma_{15} = \left(\left(\frac{d_{15}^2}{d_{15}^2} + \frac{d_{15}^2}{d_{25}^2} + \frac{d_{15}^2}{d_{35}^2}\right)\right)^{-1} = \left(\frac{54.5}{54.5} + \frac{54.5}{14.5} + \frac{54.5}{0.5}\right)^{-1} = (1 + 3.7586 + 109)^{-1} = \boxed{0.0088}$$

$$\gamma_{25} = \left(\left(\frac{d_{25}^2}{d_{15}^2} + \frac{d_{25}^2}{d_{25}^2} + \frac{d_{25}^2}{d_{35}^2}\right)\right)^{-1} = \left(\frac{14.5}{54.5} + \frac{14.5}{14.5} + \frac{14.5}{0.5}\right)^{-1} = (0.2661 + 1 + 29)^{-1} = \boxed{0.0330}$$

$$\gamma_{35} = \left(\left(\frac{d_{35}^2}{d_{15}^2} + \frac{d_{35}^2}{d_{25}^2} + \frac{d_{35}^2}{d_{35}^2}\right)\right)^{-1} = \left(\frac{0.5}{54.5} + \frac{0.5}{14.5} + \frac{0.5}{0.5}\right)^{-1} = (0.0092 + 0.0345 + 1)^{-1} = \boxed{0.9582}$$

$$\gamma_{16} = \left(\left(\frac{d_{16}^2}{d_{16}^2} + \frac{d_{16}^2}{d_{26}^2} + \frac{d_{16}^2}{d_{36}^2}\right)\right)^{-1} = \left(\frac{62.5}{62.5} + \frac{62.5}{18.5} + \frac{62.5}{0.5}\right)^{-1} = (1 + 3.3784 + 125)^{-1} = \boxed{0.0077}$$

$$\gamma_{26} = \left(\left(\frac{d_{26}^2}{d_{16}^2} + \frac{d_{26}^2}{d_{26}^2} + \frac{d_{26}^2}{d_{36}^2}\right)\right)^{-1} = \left(\frac{18.5}{62.5} + \frac{18.5}{18.5} + \frac{18.5}{0.5}\right)^{-1} = (0.296 + 1 + 37)^{-1} = \boxed{0.0261}$$

$$\gamma_{36} = \left(\left(\frac{d_{36}^2}{d_{16}^2} + \frac{d_{36}^2}{d_{26}^2} + \frac{d_{36}^2}{d_{36}^2}\right)\right)^{-1} = \left(\frac{0.5}{62.5} + \frac{0.5}{18.5} + \frac{0.5}{0.5}\right)^{-1} = (0.0080 + 0.0270 + 1)^{-1} = \boxed{0.9662}$$

Cluster	**(1,2)**	**1,3**	**(4,3)**	**5,4**	**(8,6)**	**(8,7)**
1	0.9675	0.9599	0.263	0.0325	0.0088	0.0077
2	0.0261	0.0331	0.684	0.942	0.033	0.0261
3	0.0063	0.007	0.0526	0.0255	0.958	0.966
	1	1	2	2	3	3

Table 3: Caption

Calculation of New Centroid:

$$V_{ij} = \frac{\sum_{k=1}^{4} \gamma_{ik}^{m} \cdot x_k}{\sum_{k=1}^{n} \gamma_{ik}^{m}}$$

$$V_{11} = \frac{0.967^2 \cdot 1 + 0.9599^2 \cdot 1 + 0.263^2 \cdot 4 + 0.0325^2 \cdot 5 + 0.0088^2 \cdot 8 + 0.0077^2 \cdot 8}{0.967^2 + 0.9599^2 + 0.263^2 \cdot 4 + 0.0325^2 \cdot 5 + 0.0088^2 + 0.0077^2} = 1.094$$

$$V_{12} = \frac{0.967^2 \cdot 2 + 0.9599^2 \cdot 3 + 0.263^2 \cdot 3 + 0.0325^2 \cdot 4 + 0.0088^2 \cdot 6 + 0.0077^2 \cdot 7}{0.967^2 + 0.9599^2 + 0.263^2 \cdot 4 + 0.0325^2 \cdot 5 + 0.0088^2 + 0.0077^2} = 2.434$$

$$V_{21} = \frac{0.0261^2 \cdot 1 + 0.331^2 \cdot 1 + 0.684^2 \cdot 4 + 0.942^2 \cdot 5 + 0.033^2 \cdot 8 + 0.0261^2 \cdot 8}{0.0261^2 + 0.0331^2 + 0.684^2 \cdot 4 + 0.942^2 \cdot 5 + 0.033^2 + 0.0261^2} = 4.887$$

$$V_{22} = \frac{0.0261^2 \cdot 2 + 0.331^2 \cdot 3 + 0.684^2 \cdot 3 + 0.942^2 \cdot 4 + 0.033^2 \cdot 6 + 0.0261^2 \cdot 7}{0.0261^2 + 0.0331^2 + 0.684^2 \cdot 4 + 0.942^2 \cdot 5 + 0.033^2 + 0.0261^2} = 3.266$$

$$V_{31} = \frac{0.0063^2 \cdot 1 + 0.007^2 \cdot 1 + 0.0526^2 \cdot 4 + 0.0255^2 \cdot 5 + 0.958^2 \cdot 8 + 0.966^2 \cdot 8}{0.0063^2 + 0.007^2 + 0.0526^2 \cdot 4 + 0.0255^2 \cdot 5 + 0.958^2 + 0.966^2} = 7.994$$

$$V_{32} = \frac{0.0063^2 \cdot 2 + 0.007^2 \cdot 3 + 0.0526^2 \cdot 3 + 0.0255^2 \cdot 4 + 0.958^2 \cdot 6 + 0.966^2 \cdot 7}{0.0063^2 + 0.007^2 + 0.0526^2 \cdot 4 + 0.0255^2 \cdot 5 + 0.958^2 + 0.966^2} = 6.834$$

So,new centroids are,

$\boldsymbol{C_1(1.049, 2.434)}$
$\boldsymbol{C_2(4.887, 3.266)}$
$\boldsymbol{C_3(7.994, 6.834)}$

✠ K Medoids Clustering

➤ **Example:**

☞YouTube Video- ✈

i	x	y
$\boldsymbol{X_1}$	2	6
$\boldsymbol{X_2}$	3	4
$\boldsymbol{X_3}$	3	8
$\boldsymbol{X_4}$	4	7
$\boldsymbol{X_5}$	6	2
$\boldsymbol{X_6}$	6	4
$\boldsymbol{X_7}$	7	3
$\boldsymbol{X_8}$	7	4
$\boldsymbol{X_9}$	8	5
$\boldsymbol{X_{10}}$	7	6

Table 4: Caption

Apply K-Medoid Clustering Algorithm to form two clusters.

◆**Step by step Solution:**

☞ Step-1:**Select two medoids**

$\boldsymbol{C_1(3,4)}$
$\boldsymbol{C_2(7,4)}$

☞ Step-2:**Calculate Manhattam distance between medoids and these particular data points**

Manhattam Distance=$|x_1 - x_2| + |y_1 - y_2|$

Mdist[(2,6),(3,4)]=$|2-3| + |6-4| = 3$
Mdist[(3,4),(3,4)]=$|3-3| + |4-4| = 0$
Mdist[(3,8),(3,4)]=$|3-3| + |8-4| = 4$

i	x	y	Dist from $\boldsymbol{C_1}$	Dist from $\boldsymbol{C_2}$	**Cluster**
X_1	2	6	3	7	$\boldsymbol{C_1}$
X_2	3	4	0	4	C_1
X_3	3	8	4	8	C_1
X_4	4	7	4	6	C_1
X_5	6	2	5	3	C_2
X_6	6	4	3	1	C_2
X_7	7	3	5	1	C_2
X_8	7	4	4	0	C_2
X_9	8	5	6	2	C_2
X_{10}	7	6	6	2	C_2

Table 5: Caption

☞ Step-3: **Separate the Points into Clusters**

$\boldsymbol{C_1 : \{(2,6),(3,4),(3,8),(4,7)\}}$

Medoids=(3,4)

$\boldsymbol{C_2 : \{(6,2),(6,4),(7,3),(7,4),(8,5),(7,6)\}}$

Medoids=(7,4)

☞ Step-04: **Calculate Cost of all points from Medoid of that Clusters**

Cost from (3,4) to (2,6) $= |(3-2)| + |(4-6)| = 3$

Cost from (3,4) to (3,8) $= |(3-3)| + |(4-8)| = 4$

Cost from (3,4) to (4,7) $= |(3-4)| + |(4-7)| = 4$

Cost from (7,4) to (6,2) $= |(7-6)| + |(4-2)| = 3$

Cost from (7,4) to (6,4) $= |(7-6)| + |(4-4)| = 1$

Cost from (7,4) to (7,3) $= |(7-7)| + |(4-3)| = 1$

Cost from (7,4) to (8,5) $= |(7-8)| + |(4-5)| = 2$

Cost from (7,4) to (6,2) $= |(7-7)| + |(4-6)| = 2$

Total Cost $=$ **20**

☞ **Step-5:Randomly select One Non Medoid Point and Recalculate the Cost**

New Medoids,

$\boldsymbol{C_1 = (3,4)}$ **and** $\boldsymbol{O = (7,3)}$

Mantahattam Dist= $\boldsymbol{|x_1 - x_2| + |y_1 - y_2|}$

$$\mathbf{Mdist[(2,6),(7,3)] = |2-7|+|6-3| = 8}$$
$$\mathbf{Mdist[(3,4),(7,3)] = |3-7|+|4-3| = 5}$$
$$\mathbf{Mdist[(3,8),(7,3)] = |3-7|+|8-3| = 9}$$
$$\mathbf{Mdist[(4,7),(7,3)] = 7}$$
$$\mathbf{Mdist[(6,2),(7,3)] = 2}$$
$$\mathbf{Mdist[(6,4),(7,3)] = 2}$$
$$\mathbf{Mdist[(7,3),(7,3)] = 0}$$
$$\mathbf{Mdist[(7,4),(7,3)] = 1}$$
$$\mathbf{Mdist[(8,5),(7,3)] = 3}$$
$$\mathbf{Mdist[(7,6),(7,3)] = 3}$$

i	x	y	Dist from $\boldsymbol{C_1}$	Dist from $\boldsymbol{O}$	**Cluster**
X_1	2	6	3	8	$\boldsymbol{C_1}$
X_2	3	4	0	5	C_1
X_3	3	8	4	9	C_1
X_4	4	7	4	7	C_1
X_5	6	2	5	2	O
X_6	6	4	3	2	O
X_7	7	3	5	0	O
X_8	7	4	4	1	O
X_9	8	5	6	3	O
X_{10}	7	6	6	3	O

Table 6: Caption

New Clusters are,

$$\boldsymbol{C_1 : \{(2,6),(3,4),(3,8),(4,7)\}}$$

$$\textbf{Centroid=}\boldsymbol{C_1 : (3,4)}$$

$$\boldsymbol{O : \{(6,2),(6,4),(7,3),(7,4),(8,5),(7,6)\}}$$

Centroid=O:(7,3)

Cost from (2,6) to (3,4)=$|2-3|+|6-4|=3$
Cost from (3,8) to (3,4)=4
Cost from (4,7) to (3,4)=4
Cost from (6,2) to (7,3)=2
Cost from (6,4) to (7,3)=2
Cost from (7,4) to (7,3)=1
Cost from (8,5) to (7,3)=3
Cost from (7,6) to (7,3)=3

Current Total Cost =$3+4+4+2+2+1+3+3=22$

☞ **Step-6:** Check,Whether C_2 will be replaced by O or Not

S=Current Total Cost-Previous Total Cost
$=22-20=2>0$;So Replacing C_2 by O is not good Idea.

Final Medoids are,$C_1(3,4)$;$O(7,3)$
Clusters are,
$C_1:\{(2,6),(3,4),(3,8),(4,7)\}$
$C_2:\{(6,2),(6,4),(7,3),(7,4),(8,5),(7,6)\}$

Solution Done

Most Basic Python Code Part 1

```python
import pandas as pd
data = pd.read_csv('kmedoid.csv')
data.head()

points={}
#print(points)

for index,row in data.iterrows():
    name=row['i']
    #print(name)
    x=row['x']
    #print(x)
    y=row['y']
    #print(y)
    points[name]=((x,y))

print(points)

c1=(3,4)
c2=(7,4)
print(c1)
print(c2)

column_d=[]
for j in range(10):
    key=f'X{j+1}'
    a=points[key][0]
    b=points[key][1]
    c1=(3,4)
    mdist=abs(c1[0]-a)+abs(c1[1]-b)
    column_d.append(mdist)
    #print(mdist)
print(column_d)

data['dist_from_c1']=column_d
data
```

Most Basic Python Code Part-2

```python
column_e=[]
for k in range(10):
    keyb=f'X{k+1}'
    d=points[keyb][0]
    e=points[keyb][1]
    c2=(7,4)
    mdista=abs(c2[0]-d)+abs(c2[1]-e)
    #print(mdista)
    column_e.append(mdista)
print(column_e)

data['dist_from_c2']=column_e
data

cluster=[]
for m in range(10):
    column_a=data['dist_from_c1']
    column_b=data['dist_from_c2']
    if column_a[m]<column_b[m]:
        cluster.append('C1')
    else:
        cluster.append('C2')
print(cluster)

data['cluster']=cluster
data
```

Most Basic Python Code Part-3

```python
poi_b=[]
poi_a = []
for index, row in data.iterrows():
    #print(index)
    found_C1 = False

    for cell in row:
        #print(cell)
        if 'C1' in str(cell):
            found_C1 = True
            break
    if found_C1:
        x = row['x']
        y = row['y']
        poi_a.append((x, y))
    else:
        x=row['x']
        y=row['y']
        poi_b.append((x,y))
print(poi_a)
print(poi_b)

print(poi_a[0][1])

cost_a=[]
for i in range(len(poi_a)):
    a=poi_a[i][0]
    #print(a)
    b=poi_a[i][1]
    #print(b)
    c1=(3,4)
    cost_from=abs(c1[0]-a)+abs(c1[1]-b)
    cost_a.append(cost_from)
print(cost_a)

t_cost_a=sum(cost_a)
print(t_cost_a)
```

Most Basic Python Code Part-4

```
cost_b=[]
for i in range(len(poi_b)):
    a=poi_b[i][0]
    b=poi_b[i][1]
    c2=(7,4)
    cost_from=abs(c2[0]-a)+abs(c2[1]-b)
    cost_b.append(cost_from)
print(cost_b)

t_cost_b=sum(cost_b)

print(t_cost_b)

total_cost=t_cost_a+t_cost_b
print(total_cost)

C1=(3,4)
O=(7,3)

column_c=[]
for k in range(10):
    keyc=f'X{k+1}'
    d=points[keyc][0]
    e=points[keyc][1]
    O=(7,3)
    mdista=abs(O[0]-d)+abs(O[1]-e)
    #print(mdista)
    column_c.append(mdista)
print(column_c)

data['dist_from_O']=column_c
data

cluster=[]
for m in range(10):
    column_a=data['dist_from_c1']
    column_b=data['dist_from_O']
    if column_a[m]<column_b[m]:
        cluster.append('C1')
    else:
        cluster.append('O')
print(cluster)


data['cluster_2']=cluster
data
```

Most basic Python Code Part -5

```python
C1=(3,4)
O=(7,3)

cost_c=[]
for i in range(len(poi_a)):
    a=poi_a[i][0]
    #print(a)
    b=poi_a[i][1]
    #print(b)
    c1=(3,4)
    cost_from=abs(c1[0]-a)+abs(c1[1]-b)
    cost_c.append(cost_from)
print(cost_c)
t_cost_c=sum(cost_c)


cost_d=[]
for i in range(len(poi_b)):
    a=poi_b[i][0]
    b=poi_b[i][1]
    O=(7,3)
    cost_from=abs(O[0]-a)+abs(O[1]-b)
    cost_d.append(cost_from)
print(cost_d)

t_cost_d=sum(cost_d)

print(t_cost_d)

current_total_cost=t_cost_c+t_cost_d
print(current_total_cost)

S=current_total_cost-total_cost

print(S)
if S > 0:
    print("S=2 > 0")
    print("Final Medoids are: C1(3,4) ; O(7,3)")
    print("Clusters are:")
    print("  C1: {(2,6), (3,4), (3,8), (4,7)}")
    print("  C2: {(6,2), (6,4), (7,3), (7,4), (8,5), (7,6)}")
elif S < 0:
    print("S=2 < 0")
else:
    print("S= = 0")
```

➤ **Example-02:**

☞ YouTube Video: ✈

Tutorial Solve

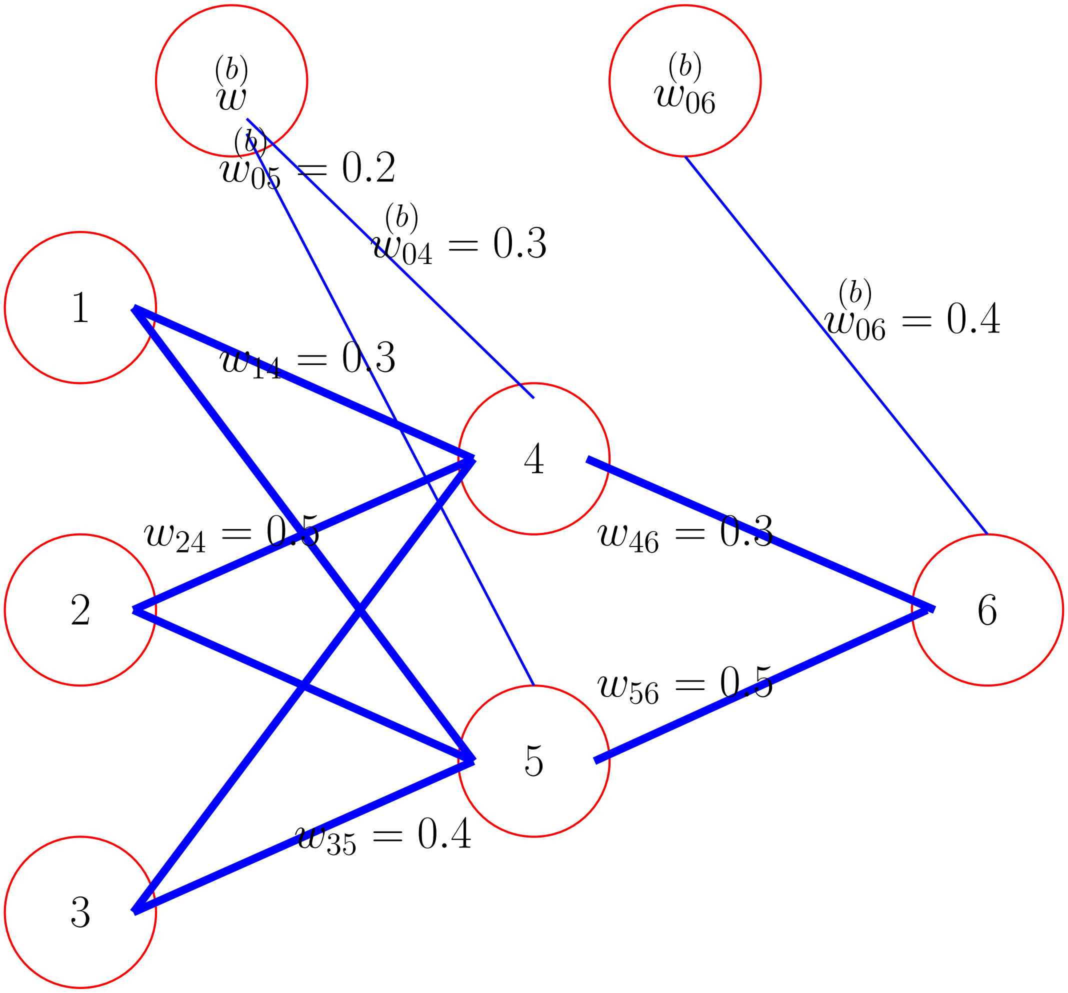
✠ First Tutorial

➤ Question-1: **Find the mean $\boldsymbol{\mu}$,for the data that follows the normal distribution where the known data are $\{1.5, 2, 8, 10\}$ with two missing items.Here,$n = 7$ and $k = 5$.Suppose the initial guess value $\mu_0 = [X_0 \text{ mod } 3]$,where X_0 =your class roll.Consider two decimal classes for detailed calculation.**

➤ Solution:

➤ **Question-2:** **The inputs of ANN be $x_1 = 0.5, x_2 = 0.9, x_3 = 0.3$, at node 1 ,node 2,node 3, respectively; the weights between input nodes to nodes(node 4 and node 5) of the hidden layers are $\overset{(h)}{w}_{14} = 0.3, \overset{(h)}{w}_{15} = 0.2, \overset{(h)}{w}_{24} = 0.5, \overset{(h)}{w}_{25} = 0.5, \overset{(h)}{w}_{34} = 0.7, \overset{(h)}{w}_{35} = 0.4$; and the weight between nodes of the hidden layer to the node(node 6) of the output layers are $\overset{(o)}{w}_{46} = 0.3, \overset{(o)}{w}_{56} = 0.5$.Also,the biases are $\overset{(b)}{w}_{04} = 0.3, \overset{(b)}{w}_{05} = 0.2$ at hidden layer; and $\overset{(b)}{w}_{06} = 0.4$ at output layer.The learning rate and target are 0.9 and 1.0 respectively.Find the updated weights after the first iteration.Consider three decimal places for detailed calculation.**

➤ **Solution:**



$x_1 = O_1 = 0.5$	$O_4 =?$	$w_{14} = 0.3$
$x_2 = O_2 = 0.9$	$O_5 =?$	$w_{15} = 0.2$
$x_3 = O_3 = 0.3$	$O_6 =?$	$w_{24} = 0.5$
$\overset{(b)}{w}_{04} = 0.3$	$I_4 =?$	$w_{25} = 0.5$
$\overset{(b)}{w}_{05} = 0.2$	$I_5 =?$	$w_{34} = 0.7$
$\overset{(b)}{w}_{06} = 0.4$	$E_6 =?$	$w_{46} = 0.3$
learning rate,$l = 0.9$		$w_{56} = 0.5$
$T_j = 1$		
		$w_{35} = 0.4$

$$I_4 = O_1 w_{14} + O_2 w_{24} + O_3 w_{34} + w_{04}^{(b)}$$
$$I_4 =$$
$$I_4 =$$

$$O_4 = \frac{1}{1 + \mathrm{e}^{-I_4}}$$
$$O_4 = \frac{1}{1 + \mathrm{e}^{-}}$$
$$O_4 =$$

$$I_5 = O_1 w_{15} + O_2 w_{25} + O_3 w_{35} + w_{05}^{(b)}$$
$$I_5 =$$
$$I_5 =$$

$$O_5 = \frac{1}{1 + \mathrm{e}^{-I_5}}$$
$$O_5 = \frac{1}{1 + \mathrm{e}^{-}}$$
$$O_5 =$$

$$I_6 = O_4 w_{46} + O_5 w_{56} + w_{06}^{(b)}$$
$$I_6 = \times 0.5 + 0.842 \times 0.8 + 0.6$$
$$I_6 = \qquad \text{Or,}$$

$$O_6 = \frac{1}{1 + \mathrm{e}^{-I_6}}$$
$$O_6 = \frac{1}{1 + \mathrm{e}^{-}}$$
$$O_6 = \qquad \text{Or,}$$

$$\begin{aligned}
E_{rr_6} &= O_6(1-O_6)(T_j-O_6)\\
&=\\
&=
\end{aligned}$$

$$\begin{aligned}
w_{46_{new}} &= w_{46_{old}} + l \times E_6 \times O_4\\
&=\\
&=
\end{aligned}$$

$$\begin{aligned}
E_4 &= O_4 \times (1-O_4) \times E_6 \times w_{46_{new}}\\
E_4 &=\\
&=
\end{aligned}$$

$$\begin{aligned}
w_{56_{new}} &= w_{56_{old}} + lE_6O_5\\
w_{56_{new}} &=\\
w_{56_{new}} &=
\end{aligned}$$

$$\begin{aligned}
E_5 &= O_5(1-O_5)E_6w_{56_{new}}\\
E_5 &=\\
E_5 &=
\end{aligned}$$

$$\begin{aligned}
w_{14_{new}} &= w_{14_{old}} + lE_4O_1\\
w_{14_{new}} &=\\
w_{14_{new}} &=
\end{aligned}$$

$$\begin{aligned}
w_{24_{new}} &= w_{24_{old}} + lE_4O_2\\
w_{24_{new}} &=\\
w_{24_{new}} &=
\end{aligned}$$

$$
\begin{aligned}
w_{34_{new}} &= w_{34_{old}} + lE_4O_3 \\
w_{34_{new}} &= \\
w_{34_{new}} &=
\end{aligned}
$$

$$
\begin{aligned}
w_{15_{new}} &= w_{14_{old}} + lE_5O_1 \\
w_{15_{new}} &= \\
w_{15_{new}} &=
\end{aligned}
$$

$$
\begin{aligned}
w_{25_{new}} &= w_{25_{old}} + lE_5O_2 \\
w_{25_{new}} &= \\
w_{25_{new}} &=
\end{aligned}
$$

$$
\begin{aligned}
w_{35_{new}} &= w_{35_{old}} + lE_5O_3 \\
w_{35_{new}} &= \\
w_{35_{new}} &=
\end{aligned}
$$

➤ **Question-3:** **Apply self organizing Map(MOP) to cluster the A,B,C, and D data points for an iteration.Assume that the initial learning rate is 0.8 and the number of clusters to be formed is 2.Also obtain the learning rate after the first iteration.Consider three decimal places for detailed calculation.**

Table 7: Data Point for SOM

i	**A**	**B**	**C**	**D**
1	1	0	0	0
2	0	1	0	1
3	1	1	0	1
4	1	1	1	0

Consider the initial weight matrix:

$$\begin{bmatrix} \mathbf{0.2} & \mathbf{0.7} \\ \mathbf{0.3} & \mathbf{0.6} \\ \mathbf{0.4} & \mathbf{0.4} \\ \mathbf{0.5} & \mathbf{0.9} \end{bmatrix}$$

➤ **Solution:**

➤ **Question-4:** **Consider the distance matrix from a data set and the cluster solution.Compute the Dunn Index for $k = 2$ and $k = 3$ to find the optimal number of clusters.Comment on your results.**

$$\begin{array}{c|ccccc|}
 & \mathbf{A} & \mathbf{B} & \mathbf{C} & \mathbf{D} & \mathbf{E} \\
\mathbf{A} & 0 & & & & \\
\mathbf{B} & 60 & 0 & & & \\
\mathbf{C} & 69 & 34 & 0 & & \\
\mathbf{D} & 194 & 153 & 144 & 0 & \\
\mathbf{E} & 24 & 56 & 6 & 188 & 0
\end{array}$$

Table 8: $\boldsymbol{k = 3}$

k = 3		
Cluster	**No of Items**	**Items**
Cluster-1	2	{A,E}
Cluster-2	2	{B,C}
Cluster-3	1	{D}

Table 9: $\boldsymbol{k = 4}$

k = 4		
Cluster	**No of Items**	**Items**
Cluster-1	2	{A,E}
Cluster-2	2	{B}
Cluster-3	1	{C}
Cluster-4	1	{D}

➤ **Solution:**

✠ Decision Tree

➤ **Example-01:Make a decision tree for the following dataset.**

✚ Source: Click On-✈

Day	Weather	Temperature	Humidity	Wind	Play Football
Day_1	Sunny	Hot	High	Weak	No
Day_2	Sunny	Hot	High	Strong	No
Day_3	Cloudy	Hot	High	Weak	Yes
Day_4	Rain	Mild	High	Weak	Yes
Day_5	Rain	Cool	Normal	Weak	Yes
Day_6	Rain	Cool	Normal	Strong	No
Day_7	Cloudy	Cool	Normal	Strong	Yes
Day_8	Sunny	Mild	High	Weak	No
Day_9	Sunny	Cool	Normal	Weak	Yes
Day_10	Rain	Mild	Normal	Weak	Yes
Day_11	Sunny	Mild	Normal	Strong	Yes
Day_12	Cloudy	Mild	High	Strong	Yes
Day_13	Cloudy	Hot	Normal	Weak	Yes
Day_14	Rain	Mild	High	Strong	No

Table 10: Weather data and football decision

Solution

Total number of Sunny Day=5
Total number of Cloudy Day=4
Total number of Rainy Day=5
Total =14
Total Positive("Yes") =9
Total Negative("No") =5
Number of Positive("yes") for Sunny=2

Number of Negative("No") for Sunny=3
Number of Positive("yes") for Rain=3
Number of Negative("No") for Rain=2
Number of Positive("yes") for Cloudy=4
Number of Negative("No") for Cloudy=0

◆ **Calculate IG(Information Gain) of Weather**

☞ Step-1:**Entropy of entire Dataset**

$$\textbf{Total Entropy(S)} = -(p_+)\log_2^{p_+} - (p_-)\log_2^{p_-}$$

Where,
$\boldsymbol{p_+}$ **=Positive Class Proportion(Ex:"Yes")**
$\boldsymbol{p_-}$ **=Negative Class Proportion(Ex:"No")**

Here,
$\boldsymbol{p_+ = \frac{9}{14}}$
$\boldsymbol{p_- = \frac{5}{14}}$

$$S\{+9,-5\} = \frac{-9}{14}\log_2^{\frac{9}{14}} - \frac{5}{14}\log_2^{\frac{5}{14}} = 0.94$$

☞ Step-2:**Entropy of All Attribute**

Weather Part

✓ **Entropy of sunny:**

Number of Positive("yes") for Sunny=2
So, $\boldsymbol{p_+ = \frac{2}{5}}$
Number of Negative("No") for Sunny=3
So, $\boldsymbol{p_- = \frac{3}{5}}$

$$\textbf{Entropy(Sunny)}\{+2,-3\} = -\frac{2}{5}\log_2^{\frac{2}{5}} - \frac{3}{5}\log_2^{\frac{3}{5}} = 0.97$$

✓ **Entropy of Cloudy:**

Number of Positive("yes") for Cloudy=4
So, $p_+ = \frac{4}{4}$
Number of Negative("No") for Cloudy=0
So, $p_- = \frac{0}{4}$

$$\textbf{Entropy(Cloudy)}\{+4, -0\} = -\frac{4}{4}\log_2^{\frac{4}{4}} - \frac{0}{4}\log_2^{\frac{0}{4}} = 0$$

✓ **Entropy of Rain:**

Number of Positive("yes") for Rain=3
So, $p_+ = \frac{3}{5}$
Number of Negative("No") for Rain=2
So, $p_- = \frac{2}{5}$

$$\textbf{Entropy(Rain)}\{+3, -2\} = -\frac{3}{5}\log_2^{\frac{3}{5}} - \frac{2}{5}\log_2^{\frac{2}{5}} = 0.97$$

✓ **Information Gain:**

$$\begin{aligned}\textbf{Information Gain(Weather)} &= Entropy_{(Total)} - \frac{5}{14}Entropy(S\\ &- \frac{4}{14}Entropy(Cloudy) - \frac{5}{14}Ent\\ &= 0.246\end{aligned}$$

Temparature Part

✓ **Entropy of Hot:**

Number of Positive("yes") for Hot=2
So, $p_+ = \frac{2}{4}$
Number of Negative("No") for Hot=2
So, $p_- = \frac{2}{4}$

$$\textbf{Entropy(Hot)}\{+2, -2\} = -\frac{2}{4}\log_2^{\frac{2}{4}} - \frac{2}{4}\log_2^{\frac{2}{4}} = 1.0$$

✓ **Entropy of Mild:**

Number of Positive("yes") for Mild=4
So, $p_+ = \frac{4}{6}$
Number of Negative("No") for Mild=2
So, $p_- = \frac{2}{6}$

$$\textbf{Entropy(Mild)}\{+4,-2\} = -\frac{4}{6}\log_2^{\frac{4}{6}} - \frac{2}{6}\log_2^{\frac{2}{6}} = 0.918$$

✓ **Entropy of Cold:**

Number of Positive("yes") for Cold=3
So, $p_+ = \frac{3}{4}$
Number of Negative("No") for Cold=1
So, $p_- = \frac{1}{4}$

$$\textbf{Entropy(Cold)}\{+3,-1\} = -\frac{3}{4}\log_2^{\frac{3}{4}} - \frac{1}{4}\log_2^{\frac{1}{4}} = 0.81$$

✓ **Information Gain(Temparature):**

$$\begin{aligned}\textbf{Information Gain(Temp.)} &= Entropy_{(Total)} - \frac{4}{14}Entropy(Ho\\ &- \frac{6}{14}Entropy(Mild) - \frac{4}{14}Entropy\\ &= 0.029\end{aligned}$$

Humadity Part

✓ **Entropy of High:**

Number of Positive("yes") for High=3
So, $p_+ = \frac{3}{7}$
Number of Negative("No") for High=4
So, $p_- = \frac{4}{7}$

$$\textbf{Entropy(High)}\{+3,-4\} = -\frac{3}{7}\log_2^{\frac{3}{7}} - \frac{4}{7}\log_2^{\frac{4}{7}} = 0.98$$

✓ **Entropy of Normal:**

Number of Positive("yes") for Normal=6

So, $\boldsymbol{p_+ = \frac{6}{7}}$

Number of Negative("No") for Normal=1

So, $\boldsymbol{p_- = \frac{1}{7}}$

$$\textbf{Entropy(Normal)}\{+6,-1\} = -\frac{6}{7}\log_2^{\frac{6}{7}} - \frac{1}{7}\log_2^{\frac{1}{7}} = 0.59$$

✓ **Information Gain(Humadity):**

$$\textbf{IG(Hum.)} = Entropy_{(Total)} - \frac{7}{14}Entropy(High) - \frac{7}{14}Entropy($$
$$= 0.15$$

Wind Part

✓ **Entropy of Strong:**

Number of Positive("yes") for Strong=3

So, $\boldsymbol{p_+ = \frac{3}{6}}$

Number of Negative("No") for Strong=3

So, $\boldsymbol{p_- = \frac{3}{6}}$

$$\textbf{Entropy(Strong)}\{+3,-3\} = -\frac{3}{6}\log_2^{\frac{3}{6}} - \frac{3}{6}\log_2^{\frac{3}{6}} = 1.0$$

✓ **Entropy of Weak:**

Number of Positive("yes") for Weak=6

So, $\boldsymbol{p_+ = \frac{6}{8}}$

Number of Negative("No") for Weak=2

So, $\boldsymbol{p_- = \frac{2}{8}}$

$$\textbf{Entropy(Weak)}\{+6,-2\} = -\frac{6}{8}\log_2^{\frac{6}{8}} - \frac{2}{8}\log_2^{\frac{2}{8}} = 0.81$$

✓ **Information Gain(Wind):**

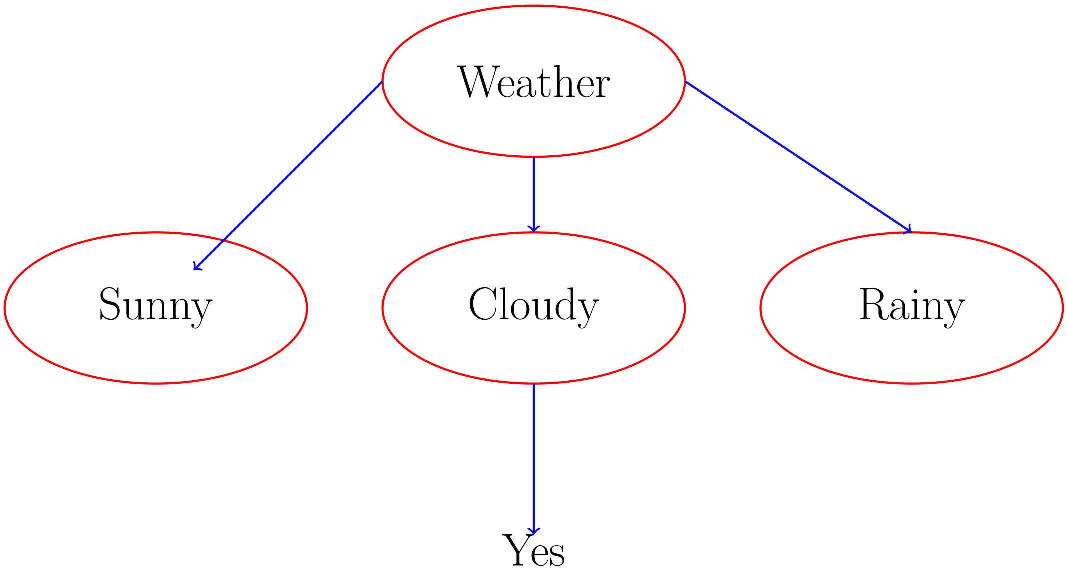
$$\textbf{IG(Wind)} = Entropy_{(Total)} - \frac{6}{14} Entropy(Strong) - \frac{8}{14} Entrop$$
$$= 0.0478$$

Now We Have,

IG(S,weather)=0.246
IG(S,temparature)=0.029
IG(S,humadity)=0.15
IG(S,wind)=0.0478

$\boldsymbol{IG_{max}}$**(max of four)=0.246(weather)**

So,Weather will be root Node



Number of positive("Yes") for Cloudy=4
Number of Negative("No") for Cloudy=0
So,we are not taking this(Cloudy) for further calcualtion.We will take other two(Sunny and Rain)

Table 11: Sunny

Day	Weather	Temparature	Humadity	Wind	Play Football
Day 1	Sunny	Hot	High	Weak	No
Day 2	Sunny	Hot	High	Strong	No
Day 8	Sunny	Mild	High	Weak	No
Day 9	Sunny	Cool	Normal	Weak	Yes
Day 11	Sunny	Mild	Normal	Strong	Yes

◆ Entropy and IG calculation fot the new Table 8."Sunny"

☆ Step-1:Total Entropy for Sunny

$$\textbf{Total Entropy(S)} = -(p_+)\log_2^{p+} - (p_-)\log_2^{p-}$$

Where,

$\boldsymbol{p_+}$ **=Positive Class Proportion(Ex:"Yes")**

$\boldsymbol{p_-}$ **=Negative Class Proportion(Ex:"No")**

Here,

$$\boldsymbol{p_+ = \frac{2}{5}}$$
$$\boldsymbol{p_- = \frac{3}{5}}$$

$$\boldsymbol{Entropy_{(Sunny)}\{+2,-3\} = -\frac{2}{5}\log_2^{\frac{2}{5}} - \frac{3}{5}\log_2^{\frac{3}{5}} = 0.97}$$

☆ Step-2: Entropy and IG for All Attribute

Temparature

✓ **Entropy of Hot:**

$$\boldsymbol{Entropy_{(Hot)}\{+0,-2\} = -\frac{0}{2}\log_2^{\frac{0}{2}} - \frac{2}{2}\log_2^{\frac{2}{2}} = 0}$$

✓ **Entropy of Mild:**

$$\boldsymbol{Entropy_{(Mild)}\{+1,-1\} = -\frac{1}{2}\log_2^{\frac{1}{2}} - \frac{1}{2}\log_2^{\frac{1}{2}} = 1}$$

✓ Entropy of Cool:

$$Entropy_{(Cool)}\{+1,-0\} = -\frac{1}{1}\log_2^{\frac{1}{1}} - \frac{0}{1}\log_2^{\frac{0}{1}} = 0$$

✓ Information Gain:

$$IG = Entropy_{(Sunny)} - \frac{2}{5}Ent_{(Hot)} - \frac{2}{5}Ent_{(Mild)} - \frac{1}{5}Ent_{(}$$

$$IG_{(temp)} = 0.57$$

Humadity

✓ Entropy of High:

$$Entropy_{(High)}\{+0,-3\} = -\frac{0}{3}\log_2^{\frac{0}{3}} - \frac{3}{3}\log_2^{\frac{3}{3}} = 0$$

✓ Entropy of Normal:

$$Entropy_{(Normal)}\{+2,-0\} = -\frac{2}{2}\log_2^{\frac{2}{2}} - \frac{0}{2}\log_2^{\frac{0}{2}} = 0$$

✓ Information Gain:

$$IG = Entropy_{(Sunny)} - \frac{3}{5}Ent_{(High)} - \frac{2}{5}Ent_{(Normal)}$$

$$IG_{(humadity)} = 0.97$$

Wind

✓ Entropy of Strong:

$$Entropy_{(Strong)}\{+1,-1\} = -\frac{1}{2}\log_2^{\frac{1}{2}} - \frac{1}{2}\log_2^{\frac{1}{1}} = 1$$

✓ Entropy of Weak:

$$Entropy_{(Weak)}\{+1,-2\} = -\frac{1}{3}\log_2^{\frac{1}{3}} - \frac{2}{3}\log_2^{\frac{2}{3}} = 0.918$$

✓ **Information Gain:**

$$IG = Entropy_{(Sunny)} - \frac{2}{5}Ent_{(Strong)} - \frac{3}{5}Ent_{(Weak)}$$

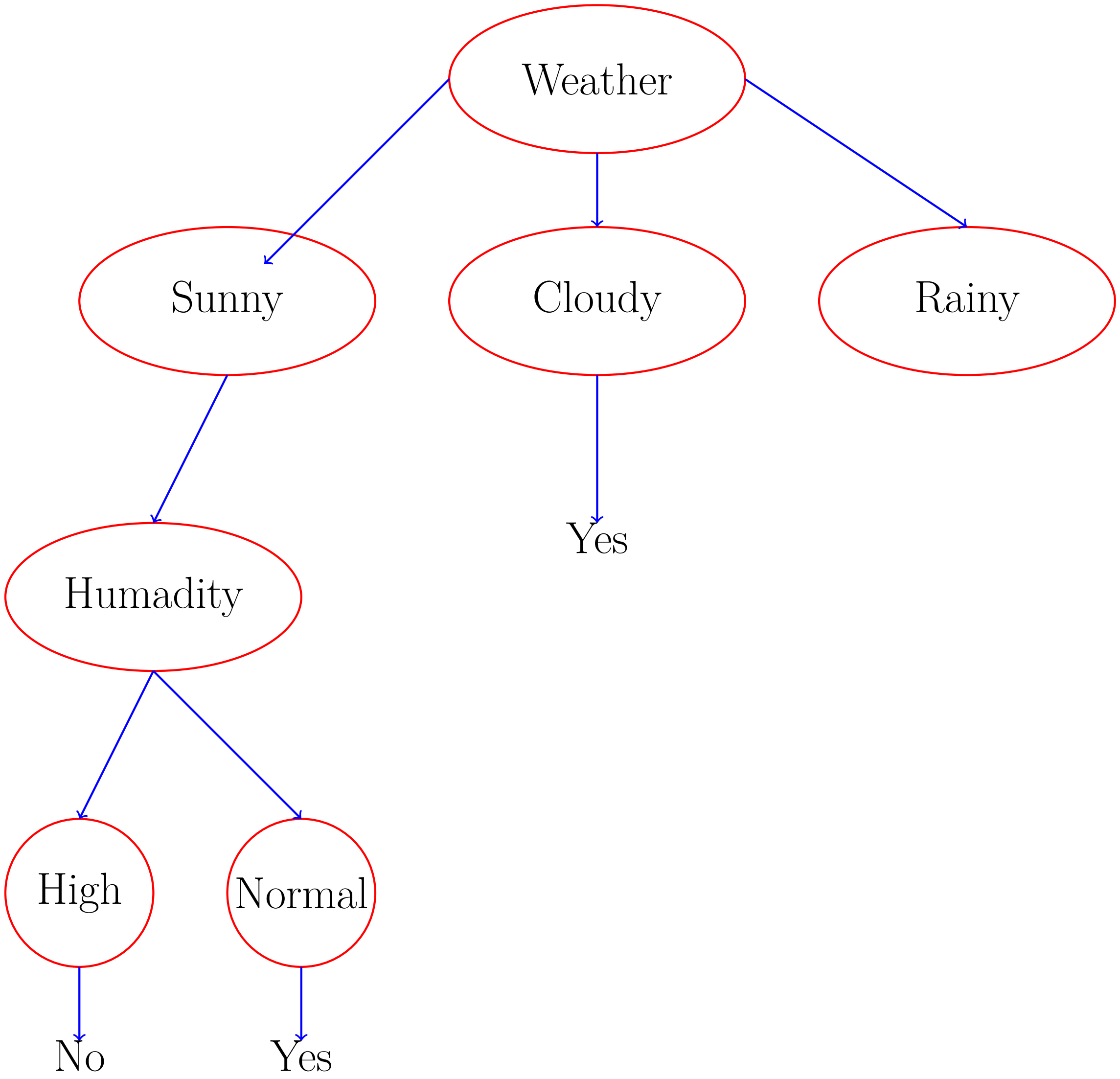
$$IG_{(Wind)} = 0.019$$

We Get,

$IG(S_{(sunny)},Temparature) = 0.57$
$IG(S_{(sunny)},Humidity) = 0.97$
$IG(S_{(sunny)},Wind) = 0.019$

IG_{max}**(max of three)=0.246(Humadity)**

So,Humadity will be Taken after sunny



◆ **Calculate Entropy and IG(Information Gain) of Rain**

☞ **Step-1:Total Entropy of Rain**

$$\textbf{Total Entropy(S)} = -(p_+)\log_2^{p+} - (p_-)\log_2^{p-}$$

Where,

p_+ =Positive Class Proportion(Ex:"Yes") **p_- =Negative Class Proportion(Ex:"No")**

Here,

$$p_+ = \frac{3}{5}$$

$$\boldsymbol{p_{-} = \frac{2}{5}}$$

$$S\{+3,-2\} = -\frac{3}{5}\log_2^{\frac{3}{5}} - \frac{2}{5}\log_2^{\frac{2}{5}} = 0.97$$

☞ **Step-2:Entropy of All Attributes Under Rain**

Temparature Part

✓ **Entropy of Hot:**

Number of Positive("yes") for Hot=0

So, $\boldsymbol{p_{+} = \frac{0}{0}}$

Number of Negative("No") for Hot=0

So, $\boldsymbol{p_{-} = \frac{0}{0}}$

Entropy(Hot)$\{+0,-0\} = -\frac{0}{0}\log_2^{\frac{0}{0}} - \frac{0}{0}\log_2^{\frac{0}{0}} = 0.0$

✓ **Entropy of Mild:**

Number of Positive("yes") for Mild=2

So, $\boldsymbol{p_{+} = \frac{2}{3}}$

Number of Negative("No") for Mild=1

So, $\boldsymbol{p_{-} = \frac{1}{3}}$

Entropy(Mild)$\{+2,-1\} = -\frac{2}{3}\log_2^{\frac{2}{3}} - \frac{1}{3}\log_2^{\frac{1}{3}} = 0.918$

✓ **Entropy of Cool:**

Number of Positive("yes") for Cool=1

So, $\boldsymbol{p_{+} = \frac{1}{2}}$

Number of Negative("No") for

Cool=1

So, $p_- = \frac{1}{2}$

Entropy(Cool)$\{+1, -1\} = -\dfrac{1}{2}\log_2^{\frac{1}{2}} - \dfrac{1}{2}\log_2^{\frac{1}{2}} = 1.0$

✓ **Information Gain of Temparature:**

$$\textbf{IG(Temp)} = Tot.Entropy_{(Rain)} - \frac{0}{5}Entropy(Hot)$$
$$- \frac{3}{5}Entropy(Mild) - \frac{2}{5}Entropy(Cool)$$
$$= 0.019$$

Humadity Part

✓ **Entropy of High:**

Number of Positive("yes") for High=1

So, $p_+ = \frac{1}{2}$

Number of Negative("No") for High=1

So, $p_- = \frac{1}{2}$

Entropy(High)$\{+1, -1\} = -\dfrac{1}{2}\log_2^{\frac{1}{2}} - \dfrac{1}{2}\log_2^{\frac{1}{2}} =$

✓ **Entropy of Normal:**

Number of Positive("yes") for Normal=2

So, $p_+ = \frac{2}{3}$

Number of Negative("No") for Normal=1

So, $p_- = \frac{1}{3}$

Entropy(Normal)$\{+2, -1\} = -\dfrac{2}{3}\log_2^{\frac{2}{3}} - \dfrac{1}{3}\log_2^{\frac{1}{3}} =$

✓ **Information Gain(Humadity):**

$$\begin{aligned}\textbf{IG(hum.)} &= TotalEntropy_{(Rain)} - \frac{2}{5}Entropy(High)\\ &- \frac{3}{5}Entropy(Normal)\\ &=\end{aligned}$$

Wind Part

✓ **Entropy of Strong:**

Number of Positive("yes") for Strong=2

So, $\boldsymbol{p_+ = \frac{0}{2}}$

Number of Negative("No") for Strong=0

So, $\boldsymbol{p_- = \frac{2}{2}}$

$$\textbf{Entropy(Strong)}\{+0,-2\} = -\frac{0}{2}\log_2^{\frac{0}{2}} - \frac{2}{2}\log_2^{\frac{2}{2}} =$$

✓ **Entropy of Weak:**

Number of Positive("yes") for Weak=3

So, $\boldsymbol{p_+ = \frac{3}{3}}$

Number of Negative("No") for Weak=0

So, $\boldsymbol{p_- = \frac{0}{3}}$

$$\textbf{Entropy(Weak)}\{+3,-0\} = -\frac{3}{3}\log_2^{\frac{3}{3}} - \frac{0}{3}\log_2^{\frac{0}{3}} =$$

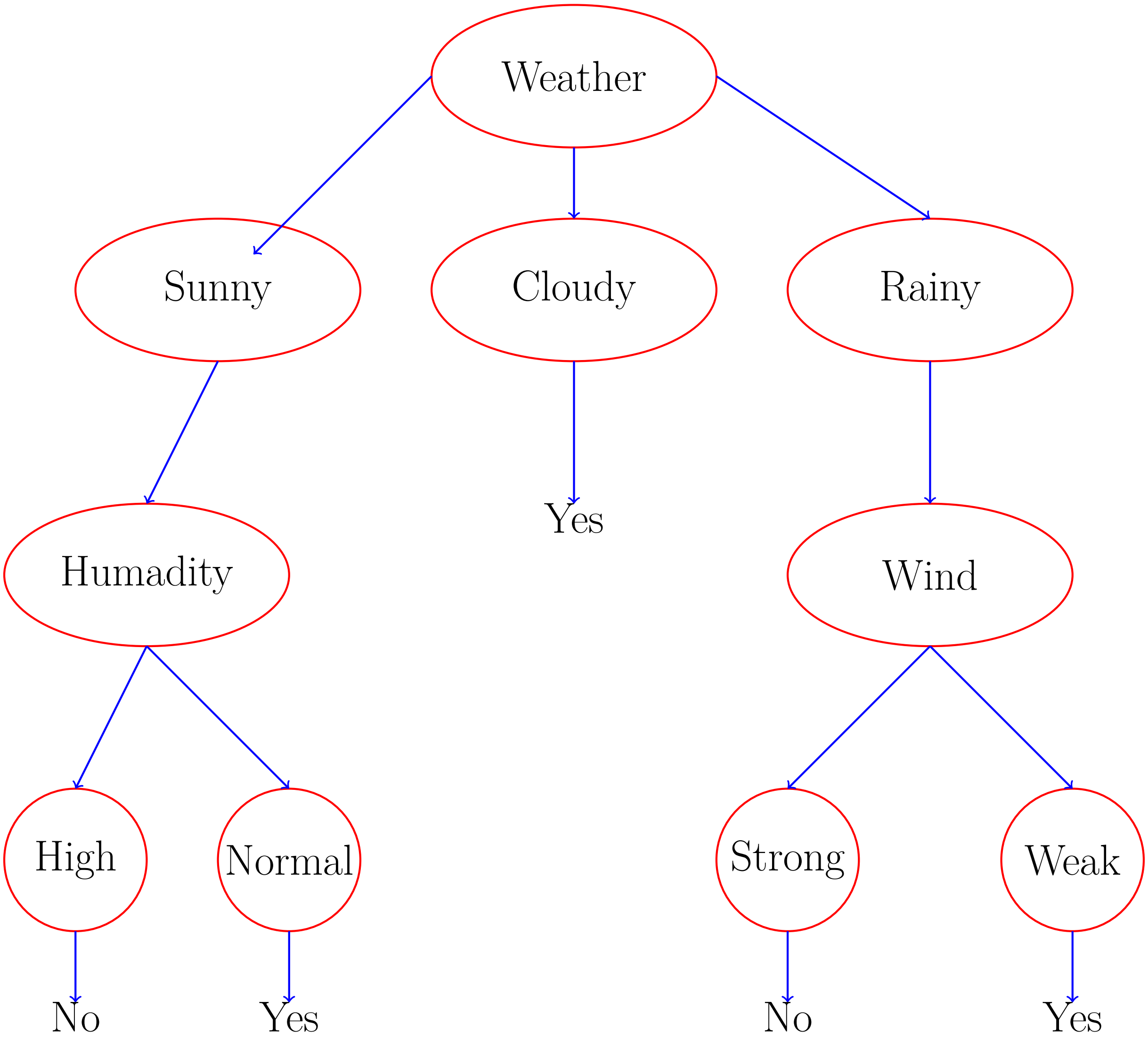
✓ **Information Gain(Wind):**

$$\textbf{IG(Wind)} = Tot.Entropy_{(Rain)} - \frac{2}{5}Entropy(Str) - \frac{3}{5}Entr$$
$$=$$

Now We Have,

IG($\boldsymbol{S_{(rain)}}$,Temp)=0.019
IG($\boldsymbol{S_{(rain)}}$,Humadity=0.019
IG($\boldsymbol{S_{(rain)}}$,Wind=0.97

$\boldsymbol{IG_{max}}$(max of three)=0.97(Wind)
So,Wind will be add in decision tree



Solution Done

➤ Example-02:Build a decesion tree for the following dataset using Classification and Regression Trees(CART) algorithm.

✚ Source: Click On-✈

CGPA	Inter Active	Practical Knowledge	Com Skill	Job Offer
≥ 9	Yes	Very Good	Good	Yes
≥ 8	No	Good	Moderate	Yes
≥ 9	No	Average	Poor	No
< 8	No	Average	Good	No
≥ 8	Yes	Good	Moderate	Yes
≥ 9	Yes	Good	Moderate	Yes
< 8	Yes	Good	Poor	No
≥ 9	No	Very Good	Good	Yes
≥ 8	Yes	Good	Good	Yes
≥ 8	Yes	Average	Good	Yes

Table 12: Caption

Solution:

◆**Step by step Solution:**

☞ Step-1: **Calculate the GiNi index for the dataset.**

$$\mathbf{Gini\_Index(T)} = 1 - \sum_{i=1}^{m} p_i^2$$

Here,

m=number of classes

p_i=Proportion of instances in class i in the dataset

From This Table

Proportion of Yes=$\frac{7}{10}$

Proportion of no=$\frac{3}{10}$

$$\textbf{Gini\_Index(T)=1} - \left(\left(\frac{\mathbf{7}}{\mathbf{10}}\right)^{\mathbf{2}} + \left(\frac{\mathbf{3}}{\mathbf{10}}\right)\right)^{\mathbf{2}} = \mathbf{0.42}$$

☞ Step-2:**Compute Gini_Index for the each of the attribute and each of the subset in the attribute**

✓ **Gini_Index for CGPA:**

CGPA	Job Offer=Yes	Job Offer=No
≥ 9	3	1
≥ 8	4	0
< 8	0	2

Table 13: Categories of CGPA

All Possible subset with the elements,$\geq 9, \geq 8, < 8$ **are,**

$\{\}, \{\geq 9\}, \{\geq 8\}, \{< 8\}, \{\geq 9, \geq 8\}, \{\geq 9, < 8\}, \{\geq 8, < 8\}, \{\geq 9, \geq 8, < 8\}$

$$\boxed{\textbf{Gini\_Index(T,A)}=\frac{|S1|}{|T|}Gini(S_1)+\frac{|S_2|}{|T|}Gini(S_2)}$$

$$\textbf{G\_I(T,CGPA(}\in\{\geq 9,\geq 8\}\textbf{))=1}-\left\{\left(\frac{3+4}{8}\right)^2+\left(\frac{1+0}{8}\right)^2\right\}=0.2194$$

$$\textbf{G\_I(T,CGPA}\in\{<8\}\textbf{=1}-\left\{\left(\frac{0}{2}\right)^2+\left(\frac{2}{2}\right)^2\right\}=0$$

$$\textbf{G\_I(T,CGPA}\in\{(\geq 9,\geq 8,<8)\}\textbf{)}=\left(\frac{8}{10}\right)\times 0.2194+\left(\frac{2}{10}\right)\times 0=0.17552$$

$$\textbf{G\_I(T,CGPA(}\in\{\geq 9,<8\}\textbf{))=1}-\left\{\left(\frac{3+0}{6}\right)^2+\left(\frac{1+2}{6}\right)^2\right\}=0.50$$

$$\textbf{G\_I(T,CGPA(}\in\{\geq 8\}\textbf{))=1}-\left\{\left(\frac{4}{4}\right)^2+\left(\frac{0}{4}\right)^2\right\}=0.0$$

$$\textbf{G\_I(T,CGPA}\in\{(\geq 9,<8,\geq 8)\}\textbf{)}=\left(\frac{6}{10}\right)\times 0.50+\left(\frac{4}{10}\right)\times 0=0.3$$

$$\textbf{G\_I(T,CGPA(}\in\{\geq 8,<8\}\textbf{))=1}-\left\{\left(\frac{4+0}{6}\right)^2+\left(\frac{2+0}{6}\right)^2\right\}=0.445$$

$$\textbf{G\_I(T,CGPA(}\in\{\geq 9\}\textbf{))=1}-\left\{\left(\frac{3}{4}\right)^2+\left(\frac{1}{4}\right)^2\right\}=0.375$$

$$\textbf{G\_I(T,CGPA}\in\{(\geq 8,<8,\geq 9)\}\textbf{)}=\left(\frac{6}{10}\right)\times 0.445+\left(\frac{4}{10}\right)\times 0.375=0.417$$

Subsets		**Gini_Index**
$\geq 9, \geq 8$	< 8	0.1755
$\geq 9, < 8$	≥ 8	0.3
$\geq 8, < 8$	≥ 9	0.417

Table 14: Gini_Index of CGPA for all possibilities

☞ **Step-3: Choose the best splitting subset which has minimum Gini_Index for an attribute**

Subsets		Gini_Index
$\geq 9, \geq 8$	< 8	0.1755
$\geq 9, < 8$	≥ 8	0.3
$\geq 8, < 8$	≥ 9	0.417

Table 15: Gini_Index of CGPA for all possibilities

The Subset CGPA=$\{(\geq 9, \geq 9), < 8\}$ has the lowest Gini_Index value as 0.1755 is choosen as the best splitting subset.

☞ **Step-4: Compute ΔGini or best splitting subset of that attribute.**

$$\begin{aligned}\Delta\text{Gini(CGPA=)Gini(T)-Gini(T,CGPA)} \\ \mathbf{0.42 - 0.1755 = 0.2445}\end{aligned}$$

Incomplete-Try Again-Go to YouTube Video

✠ KNN

➤ Example-1:Predicting Movie Genre

IMDb Rating	Duration	Genre
8.0(Mission Impossible)	160	Action
6.2(Gadar 2)	170	Action
7.2(Rocky & Rani)	168	Comedy
8.2(OMG 2)	155	Comedy

Table 16: Caption

Now predict the Genre of "Barbie" movie with IMDb rating 7.4 and duration 114.

✚ Source: **Click On-✈**

Solution:

◆ **Step by Step Solution:**

☞ Step-1:**Calculate the Euclidean distance between the new movie(7.4,114) and each movie in the dataset**

$$\begin{aligned}
\textbf{Distance to (8.0,160)} &= \sqrt{((8-7.4)^2+(160-114)^2)} \\
&= 46.00 \\
\textbf{Distance to (6.2,170)} &= \sqrt{((6.2-7.4)^2+(170-114)^2)} \\
&= 56.01 \\
\textbf{Distance to (7.2,168)} &= \sqrt{((7.2-7.4)^2+(168-114)^2)} \\
&= 54.00 \\
\textbf{Distance to (8.2,155)} &= \sqrt{((8.2-7.4)^2+(155-114)^2)} \\
&= 41.00
\end{aligned}$$

☞ **Step-2:Select K Nearest Neighbours**
Minimum Distance of (7.4,114) from these four point=41.00,(Distance between (7.4,114) and (8.2,155))

Let,K=3,Then,Minimum three distances are=41.00, 46.00,54.00
For 41.00,Genre is,"Comedy"
For 54.00,Genre is,"Comedy"
For 46.00,Genre is,"Action"

☞ **Step-3:Majority Voting(Classification)**
Class={"Action,"Comedy","Comedy"}
Majority={"Comedy","Comedy"}

So,Genre of "Barbie" movie with IMDb rating 7.4 and duration 114.is "Comedy"(Answer)

Core Basic Python Code:Part-1

```
import pandas as pd
#create a dictionary by the information of the table
dictionary={"IMDB Rating":[8.0,6.2,7.2,8.2],
            'Duration':[160,170,168,155],
           'Genre':['Action','Action','Comedy','Comedy']}
print(dictionary)

#Create DataFrame
data=pd.DataFrame(dictionary)
data

#Create a list consisting four points(IMDB Rating,Duration)
points=[]
for index,row in data.iterrows():
    x=row['IMDB Rating']
    #print(x)
    y=row['Duration']
    #print(y)
    points.append((x,y))
print(points)


"""Calculate Euclidean Distance from (7.4,114) to
individual point existing in the list named points"""
import math
target_point=(7,4,114)
p=(7.4,114)
distance=[]
for i in range(len(points)):
    a=points[i][0]
    #print(a)
    b=points[i][1]
    #print(b)
    Euclidean_dist=math.sqrt((p[0]-a)**2 + (p[1]-b)**2)
    #print(Euclidean_dist)
    distance.append(Euclidean_dist)
print(distance)
```

Core Basic Python Code:Part-2

```
#Add distance as a column to The dataset named data
data['dist']=distance
data

#Make four pair with the element of Genre and Distance
pair=[]
for indexx,roww in data.iterrows():
    w=roww['dist']
    #print(w)
    y=roww['Genre']
    #print(y)
    pair.append((w,y))
print(pair)


#Three minimum distance value in a list
three_minimum = sorted(distance)[:3]
print(three_minimum)



#Separate corresponding Genre with respect to three minimum distance value
genre=[]
for dist_value in three_minimum:
    for i in range(len(pair)):
        if pair[i][0]==dist_value:
            genre.append(pair[i][1])
print(genre)


#Find total number of individual Genre
counts={}  #Dictionary to count occurences
for item in genre:
    if item in counts:
        counts[item]+=1
    else:
        counts[item]=1
print(counts)
```

Core Basic Python Code:Part-3

```python
#Make list of number of Genre
number_of_genre=[]
for x in counts:
    z=counts[x]
    #print(z)
    number_of_genre.append(z)
print(number_of_genre)




maximum=max(number_of_genre)
print(maximum)


number_of_genre=[]
for x in counts:
    z=counts[x]
    #print(z)
    number_of_genre.append(z)
print(number_of_genre)

maximum=max(number_of_genre)
print(maximum)

for key, val in counts.items():
    if val == maximum:
        print("Majority Genre:", key)
        print(f'Genre of new movie is "{key}"')
```

➤ **Example-2:Given the following training instances(see table),each having two attributes(x_1 and x_2).Compute the class label for test instances $t_1 = (3, 7)$ using three nearest neighbours(k=3).** ✚ Source: **Click On-✈**

Training Instance	x_1	x_2	Output
$\boldsymbol{I_1}$	7	7	0
$\boldsymbol{I_2}$	7	4	0
$\boldsymbol{I_3}$	3	4	1
$\boldsymbol{I_4}$	1	4	1

Table 17: Caption

Solution:

◆**Step by step Solution:**

☞ **Step-1:Calculate the distances from the particular new(test) instances,$t_1 = (3, 7)$ to the existing data points**

Training Instance	x_1	x_2	Output	Distance	Neighbour Rank
I_1	7	7	0	$\sqrt{(7-3)^2+(7-7)^2}=4$	3
I_2	7	4	0	$\sqrt{(7-3)^2+(4-7)^2}=5$	4
I_3	3	4	1	$\sqrt{(3-3)^2+(4-7)^2}=3$	1
I_4	1	4	1	$\sqrt{(1-3)^2+(4-7)^2}=3.6$	2

Table 18: Caption

Incomplete

➤ **Example-03: Apply KNN Algorithm to classify the new instance with information Sepal Length:5.2,sepal width:3.1 based on the following training data for K=5.**

✚ Source: Click On-✈

Sepal Length	Sepal Width	Species
5.3	3.7	Setosa
5.1	3.8	Setosa
7.2	3.0	Virginica
5.4	3.4	Setosa
5.1	3.3	Setosa
5.4	3.9	Setosa
7.4	2.8	Virginica
6.1	2.8	Verscicolor
7.3	2.9	Virginica
6.0	2.7	Verscicolor
5.8	2.8	Virginica
6.3	2.3	Verscicolor
5.1	2.5	Verscicolor
6.3	2.5	Verscicolor
5.5	2.4	Verscicolor

Table 19: Caption

Solution:

◆Step by step Solution:

☞ Step-1: Calculate Euclidean Distance

New instance,

(Sepal length,Sepal Width)=(5.2,3.1)

Sepal Length	Sepal Width	Species	Distance
5.3	3.7	Setosa	$\sqrt{(5.3-5.2)^2+(3.7-3.1)^2}=0.608$
5.1	3.8	Setosa	$\sqrt{(5.1-5.2)^2+(3.8-3.1)^2}=0.707$
7.2	3.0	Virginica	$\sqrt{(7.2-5.2)^2+(3.0-3.1)^2}=2.002$
5.4	3.4	Setosa	$\sqrt{(5.4-5.2)^2+(3.4-3.1)^2}=0.36$
5.1	3.3	Setosa	$\sqrt{(5.1-5.2)^2+(3.3-3.1)^2}=0.22$
5.4	3.9	Setosa	$\sqrt{(5.4-5.2)^2+(3.9-3.1)^2}=0.82$
7.4	2.8	Virginica	$\sqrt{(7.4-5.2)^2+(2.8-3.1)^2}=2.22$
6.1	2.8	Verscicolor	$\sqrt{(6.1-5.2)^2+(2.8-3.1)^2}=0.94$
7.3	2.9	Virginica	$\sqrt{(7.3-5.2)^2+(2.9-3.1)^2}=2.1$
6.0	2.7	Verscicolor	$\sqrt{(6.0-5.2)^2+(2.7-3.1)^2}=0.89$
5.8	2.8	Virginica	$\sqrt{(5.8-5.2)^2+(2.8-3.1)^2}=0.67$
6.3	2.3	Verscicolor	$\sqrt{(6.3-5.2)^2+(2.3-3.1)^2}=1.36$
5.1	2.5	Verscicolor	$\sqrt{(5.1-5.2)^2+(2.5-3.1)^2}=0.6$
6.3	2.5	Verscicolor	$\sqrt{(6.3-5.2)^2+(2.5-3.1)^2}=1.25$
5.5	2.4	Verscicolor	$\sqrt{(5.5-5.2)^2+(2.4-3.1)^2}=0.75$

Table 20: Caption

☞ Step-2 Find Rank

Sepal Length	Sepal Width	Species	Distance	Rank
5.3	3.7	Setosa	$= 0.608$	3
5.1	3.8	Setosa	$= 0.707$	6
7.2	3.0	Virginica	$= 2.002$	13
5.4	3.4	Setosa	$= 0.36$	2
5.1	3.3	Setosa	$= 0.22$	1
5.4	3.9	Setosa	$= 0.82$	8
7.4	2.8	Virginica	$= 2.22$	15
6.1	2.8	Verscicolor	$= 0.94$	10
7.3	2.9	Virginica	$= 2.1$	14
6.0	2.7	Verscicolor	$= 0.89$	9
5.8	2.8	Virginica	$= 0.67$	5
6.3	2.3	Verscicolor	$= 1.36$	12
5.1	2.5	Verscicolor	$= 0.6$	4
6.3	2.5	Verscicolor	$= 1.25$	11
5.5	2.4	Verscicolor	$= 0.75$	7

Table 21: Caption

☞ Step-3: Find K Nearest Neighbour

K=5:So,Five nearest neighbours are,

K=1,Species=Setosa
K=2,Species=Setosa
K=3,Species=Setosa
K=4,Species=Verscicolor
K=5,Species=Virginica

☞ Step-4: Majority Voting

In 5NN,Number of Setosa=3

In 5NN,Number of Verscicolor=1
In 5NN,Number of Virginica=1

So, the species of new instance with information Sepal Length:5.2,sepal width:3.1 is "Setosa"

Solution Done

✠ Confusion Matrix

➤ Example-1:

$$\begin{array}{cc} \begin{array}{c}\textbf{Dog}\\ \textbf{Cat}\\ \textbf{Rabbit}\end{array} & \begin{bmatrix} 23 & 12 & 7 \\ 11 & 29 & 13 \\ 4 & 10 & 24 \end{bmatrix} \end{array}$$

Actual	**Predicted**		
	Dog	**Cat**	**Rabbit**
Dog	23	12	7
Cat	11	29	13
Rabbit	4	10	24

Table 22: Multiclass Confusion Matrix

Calculate the value of Accuracy Rate,Sensitivity and Specificity.

✚ Source: **Click On-✈**

Solution:

	TP	TN	FP	FN
Dog	23	$29 + 13 + 10 + 24$ $=76$	$11 + 4 = 15$	$12 + 7 = 19$
Cat	29	$23 + 7 + 4 + 24$ $=58$	$12 + 10 = 22$	$11 + 13 = 24$
Rabbit	24	$23 + 12 + 11 + 29$ $=75$	$7 + 13 = 20$	$4 + 10 = 14$

Table 23: Per-Class Binary Confusion Metrics Table

$$
\begin{aligned}
\textbf{Accuracy for Dog} &= \frac{\boldsymbol{TP + TN}}{\boldsymbol{TP + TN + FP + FN}} \\
&= \frac{\mathbf{23 + 76}}{\mathbf{23 + 76 + 15 + 19}} \\
&= \frac{\mathbf{99}}{\mathbf{133}} = \mathbf{0.744 = 74.4\%}
\end{aligned}
$$

$$
\begin{aligned}
\textbf{Sensitivity for Cat} &= \frac{\boldsymbol{TP}}{\boldsymbol{TP + FN}} \\
&= \frac{\mathbf{29}}{\mathbf{29 + 24}} = \mathbf{0.54 = 54\%}
\end{aligned}
$$

$$
\begin{aligned}
\textbf{Specificity for Rabbit} &= \frac{\boldsymbol{TN}}{\boldsymbol{TN + FP}} \\
&= \frac{\mathbf{75}}{\mathbf{75 + 20}} = \mathbf{0.78 = 78\%}
\end{aligned}
$$

☞Confusion Matrix Example-2:

✚ **Source:YouTube Video-Click on-✈**

Consider the following 3-class confusion matrix.Calculate precision and recall per class.Also calculate weighted average precision and recall for classifier.

	Predicted		
Actual	15	2	3
	7	15	8
	2	3	45

Table 24: **Confusion Matrix**

Solution:

	Predicted				
Actual		A	B	C	**Total**
	A	15	2	3	20
	B	7	15	8	30
	C	2	3	45	50
	Total	24	20	56	100

Table 25: Caption

$$\textbf{Precision} = \frac{\textbf{Correctly Predicted}}{\textbf{Total Predicted}}$$

$$\textbf{Precision of Class A} = \frac{15}{24} = 0.625$$

$$\textbf{Precision of Class B} = \frac{15}{20} = 0.75$$

$$\textbf{Precision of Class C} = \frac{45}{56} = 0.80$$

$$\textbf{Recall} = \frac{\textbf{Correctly Classified}}{\textbf{Actual}}$$

$$\textbf{Class A Recall} = \frac{\mathbf{15}}{\mathbf{20}} = \mathbf{0.75}$$
$$\textbf{Class B Recall} = \frac{\mathbf{15}}{\mathbf{30}} = \mathbf{0.50}$$
$$\textbf{Class C Recall} = \frac{\mathbf{45}}{\mathbf{50}} = \mathbf{0.90}$$

$$\textbf{Accuracy} = \frac{\textbf{Total Correctly Classified}}{\textbf{Actual}}$$

The diagonal elements of the matrix represent correctly classified observations

$$\textbf{Accuracy of Classifier} = \frac{\mathbf{15 + 15 + 45}}{\mathbf{100}} = \mathbf{0.75}$$

Actual Class A instances=$\frac{20}{100} = 0.2$
Actual Class B instances=$\frac{30}{100} = 0.3$
Actual Class C instances=$\frac{50}{100} = 0.5$

Weighted average precision=

Actual Class A instances × Precision of Class A+
Actual Class B instances × Precision of Class B+
Actual Class C instances × Precision of Class C

Weighted average precision=

$$= 0.2 \times 0.625 + 0.3 \times 0.75 + 0.50 \times 0.80 = 0.75$$

Weighted average Recall=

Actual Class A instances × Recall of Class A+
Actual Class B instances × Recall of Class B+
Actual Class C instances × Recall of Class C

Weighted average Recall=

$$= 0.2 \times 0.75 + 0.3 \times 0.5 + 0.50 \times 0.90 = 0.75$$

Solution Done

➤ **Tutorial Question-48:** **The following result(Table 1) was found for test data after applying k Nearest Neighbour,(KNN) algorithm to the atmospheric data from a region of Bangladesh to classify the Rainfall(RAN) [No Rain and Trace(NRT),Light Rain(LTR),Moderate and High Rain(MHR)] based on Temparature(TEM),Dew Point Temparature(DPT),Wind Speed(WIS),Humidity (HUM),and Sea Level Pressure(SLP) for the optimal value of k=9 and seventy percent observations were used as training data and the rest of data as test data.**

		Predicted		
	Category	LRT	MHR	NRT
Actual	LTR	65	6	11
	MHR	12	53	5
	NRT	8	0	73

Table 26: Confusion Matrix for the Test Data

① **What is the actual number of observations?**

② **Find the prediction accuracy rate,error rate for the test data.**

③ **Obtain the value of sensitivity,specificity,and F_1-score for each category LTR,MHR,NRT.**

Soultion:

❶ Finding Actual Number of Observation:

		Predicted			
	Category	LRT	MHR	NRT	**Total**
Actual	LTR	65	6	11	82
	MHR	12	53	5	70
	NRT	8	0	73	81
	Total	85	59	89	233

Table 27: Confusion Matrix for the Test Data

So,Actual Number of observation is 233

❷ Finding Prediction Accuracy Rate and Error Rate:

	TP	TN	FP	FN
LTR	65	$53 + 5 + 0 + 73$ $=132$	$12 + 8 = 20$	$6 + 11 = 17$
MHR	53	$65 + 11 + 8 + 73$ $=147$	$6 + 0 = 6$	$12 + 5 = 17$
NRT	73	$65 + 6 + 12 + 53$ $=136$	$5 + 11 = 16$	$8 + 0 = 8$

Table 28: **Per-Class Binary Confusion Metrics Table**

$$\textbf{Accuracy for LTR} = \frac{\boldsymbol{TP_{LTR} + TN_{LTR}}}{\boldsymbol{TP_{LTR} + TN_{LTR} + FP_{LTR} + FN_{LTR}}}$$

$$= \frac{\mathbf{65 + 132}}{\mathbf{65 + 132 + 20 + 17}} =$$

$$\textbf{Accuracy for MHR} = \frac{\boldsymbol{TP_{MHR} + TN_{MHR}}}{\boldsymbol{TP_{MHR} + TN_{MHR} + FP_{MHR} + FN_{MHR}}}$$

$$= \frac{\mathbf{53 + 147}}{\mathbf{53 + 147 + 6 + 17}} =$$

$$\textbf{Accuracy for NRT} = \frac{\boldsymbol{TP_{NRT} + TN_{NRT}}}{\boldsymbol{TP_{NRT} + TN_{NRT} + FP_{NRT} + FN_{NRT}}}$$

$$= \frac{\mathbf{73 + 136}}{\mathbf{73 + 136 + 16 + 8}} =$$

- Helpful information:
  1. Error rate is the number of incorrect predictions divided by the total number of predictions.
  2. The sum of all elements in the confusion matrix represents the total number of predictions.
  3. The diagonal elements of the confusion matrix represent the correct predictions for each class.

Here,

Total Number of Prediction=233

Number of incorrect predictions=6+11+12+5+8+0

=42

Error Rate= $\frac{42}{233}$

❸ Finding Sensitivity,Specificity and F_1 Score:

$$\text{Sensitivity of LTR} = \frac{TP_{LTR}}{TP_{LTR} + FN_{LTR}} = \frac{65}{65+17} = 0.7927$$

$$\text{Sensitivity of MHR} = \frac{TP_{MHR}}{TP_{MHR} + FN_{MHR}} = \frac{53}{53+17} = 0.7571$$

$$\text{Sensitivity of NRT} = \frac{TP_{NRT}}{TP_{NRT} + FN_{NRT}} = \frac{73}{73+8} = 0.9012$$

$$\text{Sensitivity of LTR} = \frac{TN_{LTR}}{TN_{LTR} + FP_{LTR}} = \frac{132}{132+20} = 0.8684$$

$$\text{Sensitivity of MHR} = \frac{TN_{MHR}}{TN_{MHR} + FP_{MHR}} = \frac{147}{147+6} = 0.9608$$

$$\text{Sensitivity of NRT} = \frac{TN_{NRT}}{TN_{NRT} + FP_{NRT}} = \frac{136}{136+16} = 0.8947$$

$$\text{Precision}=\frac{\text{Correctly Predicted}}{\text{Total Predicted(Column Total)}}=\frac{TP}{TP+FP}$$

$$\text{Precision of LTR}=\frac{65}{85}=\frac{65}{65+20}=0.7647$$

$$\text{Precision of MHR}=\frac{53}{59}=\frac{53}{53+6}=0.8983$$

$$\text{Precision of NRT}=\frac{73}{89}=\frac{73}{73+16}=0.8202$$

$$\text{Recall}=\text{Sensitivity}=\frac{\text{Correctly Classified}}{\text{Actual (Row Total)}}=\frac{TP}{TP+FN}$$

$$\text{Recall of LTR}=\frac{65}{82}=\frac{65}{65+17}=0.7927$$

$$\text{Recall of MHR}=\frac{53}{70}=\frac{53}{53+17}=0.7571$$

$$\text{Recall of NRT}=\frac{73}{81}=\frac{73}{73+8}=0.9012$$

$$\boldsymbol{F_1}\textbf{-Score}=2\times\frac{\textbf{Precision}\times\textbf{Recall}}{\textbf{Precision+Recall}}$$

$$\boldsymbol{F_1}\textbf{-Score of LTR}=2\times\frac{\mathbf{0.7647\times0.7927}}{\mathbf{0.7647+0.7927}}=\mathbf{0.7784}$$

$$\boldsymbol{F_1}\textbf{-Score of MHR}=2\times\frac{\mathbf{0.8983\times0.7571}}{\mathbf{0.8983+0.7571}}=\mathbf{0.8216}$$

$$\boldsymbol{F_1}\textbf{-Score of NRT}=2\times\frac{\mathbf{0.8202\times0.9012}}{\mathbf{0.8202+0.9012}}=\mathbf{0.8590}$$

Solution Done

✠ Naive Bayes

Definition: Naïve Bayes is a simple learning algorithm that utilizes Bayes' rule together with a strong assumption that the attributes are conditionally independent given the class. While this independence assumption is often violated in practice, naïve Bayes nonetheless often delivers competitive classification accuracy. Coupled with its computational efficiency and many other desirable features, this leads to naïve Bayes being widely applied in practice[1].

➤ **YouTube Video:** ✈

Naive Bayes is a supervised machine learning algorithm which is based on applying Bayes Theorem.

➤ **An example of Bayes Theorem:** **Bag 1 contains 2 Red and 3 Black Balls.Bag 2 contains 3 Red and 4 Black Balls.One Ball is drawn at random from one of these bags and it is red.Find the probability that it is drawn from Bag 1.**

Solution:

2 Red
3 Black

Bag 1

3 Red
4 Black

Bag 2

Probability of Drawn Red Ball from Bag_1 :

$$P(Ball = Red|Bag_1) = \frac{\binom{2}{1}}{\binom{5}{1}} = \frac{2}{5}$$

Probability of Drawn Red Ball from Bag_2 :

$$P(Ball = Red|Bag_2) = \frac{\binom{3}{1}}{\binom{7}{1}} = \frac{3}{7}$$

Probability of $Bag_1, P(Bag_1) = \frac{1}{2}$
Probability of $Bag_2, P(Bag_2) = \frac{1}{2}$

According to Law of Total Probability,

According to Law of Total Probability,

$$P(Ball = Red) = P(Bag_1) \cdot P(Ball = Red|Bag_1) + P(Bag_2) \cdot P(Ball = Red|Bag_2)$$

$$P(Ball = Red) = \frac{1}{2} \cdot \frac{2}{5} + \frac{1}{2} \cdot \frac{3}{7} = \frac{29}{70} = 0.41428$$

$$P(\text{Bag}_1 \mid \text{Ball} = \text{Red}) = \frac{P(\text{Ball} = \text{Red} \mid \text{Bag}_1) \cdot P(\text{Bag}_1)}{P(\text{Ball} = \text{Red})}$$
$$= \frac{\frac{2}{5} \cdot \frac{1}{2}}{\frac{29}{70}} = 0.4828((Answer))$$

Naive Bayes Classifier Formula:

$$P(C_k|x_1,x_2,...,n) = \frac{P(C_k)\prod_{i=1}^{n} P(x_i|C_k)}{P(x_1,x_2...x_n)}$$

➤ Example-1 of Naive Bayes:

Person	Covid(Yes/No)	Flue(Yes/No)	Fever(Yes/No)
1	Yes	No	Yes
2	No	Yes	Yes
3	Yes	Yes	Yes
4	No	No	No
5	Yes	No	Yes
6	No	No	Yes
7	Yes	No	Yes
8	Yes	No	No
9	No	Yes	Yes
10	No	Yes	No

Table 29: Caption

1.What is the probability that a person has fever given that the person has both Covid and Flu?)

2.What is the probability that a person does not have fever given that the person does not have Covid and does not have Flu

➤ YouTube Video: ✈

Solution:

◆ Step by Step Solution:

☞ Step-1: **Calculate Prior Probability**
Number of Fever Positive("Yes")=7
Number of Fever Negative("No")=3
$\boldsymbol{P(Fever = Yes) = \frac{7}{10}}$

$$\boldsymbol{P(Fever = No) = \frac{3}{10}}$$

☞ **Step-2: Calculate Conditional Probability**

$\boldsymbol{P(Covid = Yes|Fever = Yes) = \frac{4}{7}}$
$\boldsymbol{P(Covid = Yes|Fever = No) = \frac{2}{3}}$
$\boldsymbol{P(Flu = Yes|Fever = Yes) = \frac{3}{7}}$
$\boldsymbol{P(Flu = Yes|Fever = No) = \frac{2}{3}}$

Disease		**Fever**	
	Status	**Yes**	**No**
Covid	Yes	$\frac{4}{7}$	$\frac{1}{3}$
	No	$\frac{3}{7}$	$\frac{2}{3}$
Flu	Yes	$\frac{3}{7}$	$\frac{1}{3}$
	No	$\frac{4}{7}$	$\frac{2}{3}$

Table 30: Caption

☞ **Step-3: Posterior Probabilities using Bayes' Theorem**

$$\begin{aligned}
P(Fe = Y|C = Y, Fl = Y) &= P(C = Y|Fe = Y) \\
&\cdot P(Fl = Y|Fe = Y) \cdot P(F = Y) \\
&= \frac{4}{7} \cdot \frac{3}{7} \cdot \frac{7}{10} = 0.1714 \\
P(Fe = N|C = N, Fl = N) &= P(C = N|Fe = N) \\
&\cdot P(Fl = N|Fe = N) \cdot P(F = N) \\
&= \frac{2}{3} \cdot \frac{2}{3} \cdot \frac{3}{10} = \frac{2}{15} = 0.1333
\end{aligned}$$

➤ Example-2 of Naive Bayes:

☞YouTube Video: ✈

Day	Outlook	Temparature	Humadity	Wind	Play Tennis
Day1	Sunny	Hot	High	Weak	No
Day2	Sunny	Hot	High	Strong	No
Day3	Overcast	Hot	High	Weak	Yes
Day4	Rain	Mild	High	Weak	Yes
Day5	Rain	Cool	Normal	Weak	Yes
Day6	Rain	Cool	Normal	Strong	No
Day7	Overcast	Cool	Normal	Strong	Yes
Day8	Sunny	Mild	High	Weak	No
Day9	Sunny	Cool	Normal	Weak	Yes
Day10	Rain	Mild	Normal	Weak	Yes
Day11	Sunny	Mild	Normal	Strong	Yes
Day12	Overcast	Mild	High	Strong	Yes
Day13	Overcast	Hot	Normal	Weak	Yes
Day14	Rain	Mild	High	Strong	No

Table 31: Caption

Now, based on this dataset, classify the following new instance using an appropriate method:

Outlook = Sunny, Temperature = Cool,
Humidity= High, Wind = Strong

Will tennis be played in this condition? Answer, Yes or No,by finding the Probability with proper justification based on the dataset.

Solution:

◆ Step by Step Solution:

- ☞ Step-1:**Calculate the prior probability**

  $$\mathbf{P(Playtennis{=}Yes){=}\frac{9}{14}}$$

  $$\mathbf{P(Playtennis{=}No){=}\frac{5}{14}}$$

- ☞ Step-2:**Calculate Conditional probability of individual Attributes**

Outlook		**Playtennis**	
	Status	**Yes**	**No**
Sunny	Yes	$\frac{2}{9}$	$\frac{2}{5}$
	No	$\frac{3}{9}$	$\frac{3}{5}$
Overcast	Yes	$\frac{4}{9}$	$\frac{4}{5}$
	No	$\frac{0}{9}$	$\frac{0}{5}$
Rain	Yes	$\frac{3}{9}$	$\frac{3}{5}$
	No	$\frac{2}{9}$	$\frac{2}{5}$

Table 32: Caption

Temparature		**Playtennis**	
	Status	Yes	No
Hot	Yes	$\frac{2}{9}$	
	No		$\frac{2}{5}$
Mild	Yes	$\frac{4}{9}$	
	No		$\frac{2}{5}$
Cool	Yes	$\frac{3}{9}$	
	No		$\frac{1}{5}$

Table 33: Caption

Feature			**Play tennis**	
		status	Yes	No
Humadity	High	Yes	$\frac{3}{9}$	
		No		$\frac{4}{5}$
	Normal	Yes	$\frac{6}{9}$	
		No		$\frac{1}{5}$
Wind	Strong	Yes	$\frac{3}{9}$	
		No		$\frac{3}{5}$
	Weak	Yes	$\frac{6}{9}$	
		No		$\frac{2}{5}$

Table 34: Caption

☞ Step-3:

Naive Bayes Classifier formula

$$\boldsymbol{P(C_k|x_1,x_2,...,n) = \frac{P(C_k)\prod_{i=1}^{n}P(x_i|C_k)}{P(x_1,x_2...x_n)}}$$

$$P(Yes|Instance) = P(Ten = Y) \cdot P(Out = Su|Tn = Y)$$
$$\cdot P(Tm = Co|Te = Y) \cdot P(Hm = Hi|Tn = Y)$$
$$\cdot P(Wi = St|Tn = Y) = \frac{9}{14} \cdot \frac{2}{9} \cdot \frac{3}{9} \cdot \frac{3}{9} \cdot \frac{3}{9} = 0.00529$$

$$P(No|Instance) = P(Ten = N) \cdot P(Out = Su|Tn = N)$$
$$\cdot P(Tm = Co|Te = N) \cdot P(Hm = Hi|Tn = N)$$
$$\cdot P(Wi = St|Tn = N) = \frac{5}{14} \cdot \frac{3}{5} \cdot \frac{1}{5} \cdot \frac{4}{5} \cdot \frac{3}{5} = 0.0411$$

$$P(Su) \cdot P(Co) \cdot P(Hi) \cdot P(St) = \frac{5}{14} \cdot \frac{4}{14} \cdot \frac{7}{14} \cdot \frac{6}{14} = 0.02187$$

$$P(tn =Y|Ot = Su, Tm = Co, Hm = Hi, Wi = St)$$

Unsolved

✠ Support Vector Machine

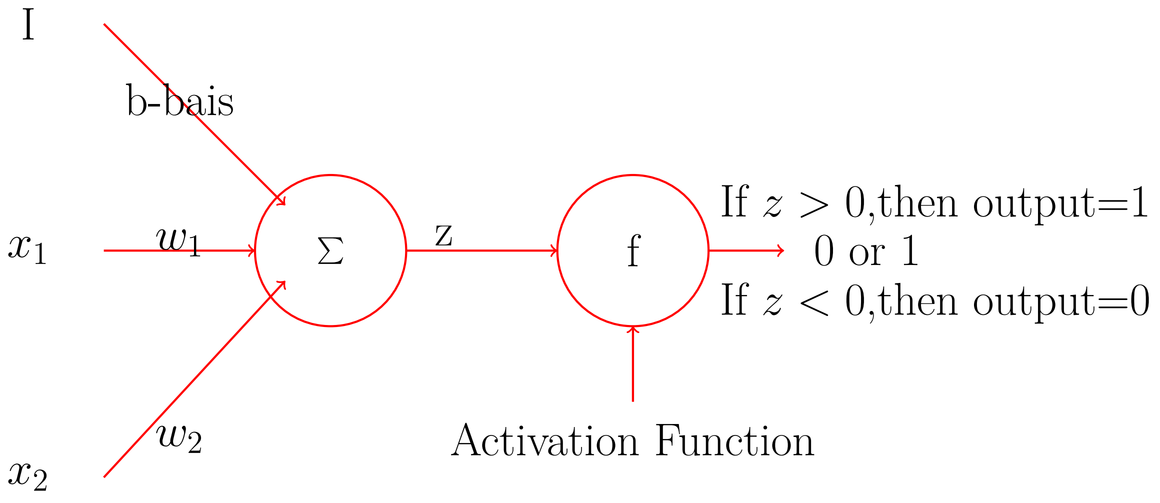
➤ **YouTube Video:** ✈

hello guys.my name is nirdesh and welcome to my youtube channel.So we are continuing the course of deep learning.Today we will start actual deep learning.we are going to start with perceptron.percentron is the fundamental building block of artificial newral network.

what is perception??
geometric interpretation of perception
importance of multilayer perception

What is perception ?
perception is an algorithm for supervised machine learning
design of perception-mathematical model-mathematical function

✚ **Source: Click On-**✈



$$z = w_1x_1 + w_2x_2 + b$$

Activation Function

$Z > 0$ output=1
$Z < 0$ output=0

IQ	CGPA	Place
78	7.8	1
69	51	0

Table 35: Caption

Two Stage,
1.Training
2.prediction
$x_1 = iq$
$x_2 = CGPA$
$w_1 =?, w_2 =?b =?$
From training,
$w_1 = 1, w_2 = 2, b = 3$
Consider a student,$iq = 100, CGPA = 5.1$,predict placement.
$z = 100 \times 1 + 5.1 \times 2 + 1 \times 3 = 113.2$
since,$z > 0$,placement=1

Geometric Intuition

$$y = f(z) = \begin{cases} 1, & \text{if } z \geq 0 \\ 0, & \text{if } z < 0 \end{cases}$$

let,
$w_1 = A$
$w_2 = B$
$b = C$
$x_1 = X$
$x_2 = Y$

$AX + BY + C = 0$
$AX + BY + C = 1$

this is straight line.

binary classifier

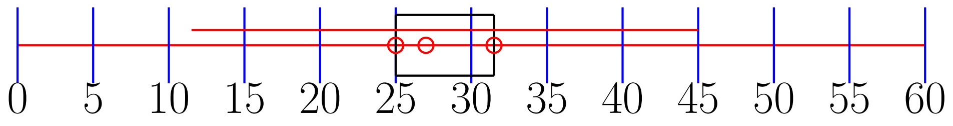
Box Plot Manual Process

Make box plot for the following data:
20,25,25,26,28,29,30,32,33,37,56

✚ Source: Click On-✈

Minimum=20
$Q_1 = 25$
$Q_2 = Median = 29$
$Q_3 = 33$
Maximum=56
IQR=inter quartile range=$Q_3 - Q_1 = 33 - 25 = 8$
Threshold=$1.5 \times IQR = 1.5 \times 8 = 12$
High Quartile=$Q_3 + Threshold = 33 + 12 = 45$
Low Quartile=$Q_1 - Threshold = 25 - 12 = 13$

For Boxplot maintain this serial: Q_1–Median–Q_3
For whisker draw line from Low quartile to High quartile



Python Code-For single box plot in a graph

```python
import matplotlib.pyplot as plt

score=[20,25,25,26,28,29,30,32,33,37,56]



fsize=plt.figure(figsize=(6,3))

ploting=plt.boxplot(score,patch_artist=True,widths=0.1)


for median in ploting['medians']:
    median.set_color('black')
    median.set_linewidth(1)


plt.ylabel('Score')
plt.show()
```

Python Code-For Double box plot in a graph

```python
import matplotlib.pyplot as plt

Male_score=[23,45,89,56,34,21,10,45,67,23,56,54,37,29,17,40,42,63]
Female_score=[108,45,95,5,34,61,10,100,34,23,42,54,56,29,17,59,12,165]


data_box=[Male_score,Female_score]


fsize=plt.figure(figsize=(6,3))
axis=fsize.add_axes([0,0,1,1])
ploting=axis.boxplot(data_box,patch_artist=True,widths=0.5)


for median in ploting['medians']:
    median.set_color('black')
    median.set_linewidth(1)


axis.set_xticklabels(['Male','Female'])


plt.xlabel('Gender')
plt.ylabel('Score')
plt.show()
```

✠ Correlation coefficient

Example: Calculate the correlation coefficient between the two variables x and given below:

✚ **Source: Click On-✈**

x	1	2	3	4	5	6
y	2	4	7	9	12	14

Table 36: Caption

Solution:

X	Y	XY	X^2	Y^2
1	2	2	1	4
2	4	8	4	16
3	7	21	9	49
4	9	36	16	81
5	12	60	25	144
6	14	84	36	196
$\sum x = 21$	$\sum y = 48$	$\sum xy = 211$	$\sum x^2 = 91$	$\sum y^2 = 490$

Table 37: Caption

$$r = \frac{n\sum xy - \sum x \sum y}{\sqrt{\{n\sum x^2 - (\sum x)^2\}\{n\sum y^2 - (\sum y)^2\}}}$$

$$r = \frac{6 \times 211 - 21 \times 48}{\sqrt{\{6 \times 91 - 21^2\}\{6 \times 490 - 48^2\}}}$$

$$r = 0.998$$

Comment: Strong positive correlation

Basic Python Code to find Correlation Coefficent

```
x=[1,2,3,4,5,6]
y=[2,4,7,9,12,14]
n=len(x)
sum_x=sum(x)
sum_y=sum(y)

sumx_into_sum_y=sum_x*sum_y
#print(sumx_into_sum_y)

a=[]
for i in range(len(x)):
    o=x[i]*y[i]
    a.append(o)
#print(a)

sum_x_into_y=sum(a)
#print(sum_x_into_y)

s_square=[]
for i in range(len(x)):
    square=x[i]
    sq=square**2
    s_square.append(sq)
#print(s_square)

sum_x_square=sum(s_square)
#print(sum_x_square)

y_square=[]
for i in range(len(y)):
    sqr=y[i]
    sq=sqr**2
    y_square.append(sq)
#print(y_square)

sum_y_square=sum(y_square)
#print(sum_y_square)

numerator=n*sum_x_into_y-sum_x*sum_y
#print(numerator)

import math
denominator=math.sqrt((n*sum_x_square-sum_x**2)*(n*sum_y_square-sum_y**2))
#print(denominator)

corrwlation_coefficient=round(numerator/denominator,4)
print(corrwlation_coefficient)
```

Python Code for Ordinal Encoding,droping Column,Correlation Matrix and Heatmap

✈What are the differences between Ordinal Encoding and Scaling?

There are two types of scaling.

1.Normalization
2.Standardization

Min Max Scaling formula:

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

Python Code for Ordinal Encoding

```
import pandas as pd
data=pd.read_csv("Student Attitude and Behavior.csv")
data.columns=data.columns.str.strip()
data.head(3)

x=list(data['Stress Level'])
#print(x)

data['Stress Level'].unique()

a=[]
for i in range(len(x)):
    if x[i]=='Awful':
        a.append(1)
    elif x[i]=='Bad':
        a.append(2)
    elif x[i]=='Good':
        a.append(3)
    else:
        a.append(4)
#print(a)

data['Stress Level']=a
data

y=list(data['Financial Status'])
#print(y)

data['Financial Status'].unique()

b=[]
for i in range(len(y)):
    if y[i]=='Awful':
        b.append(1)
    elif y[i]=='Bad':
        b.append(2)
    elif y[i]=='good':
        b.append(3)
    else:
        b.append(4)
#print(b)

data['Financial Status']=b
data
```

Python Code for Droping Columns,Correlation Matrix and Heatmap

```
data=data.drop(['Certification Course','Gender','Department',
                'hobbies','daily studing time',
                'prefer to study in','Do you like your degree?',
                'willingness to pursue a career based on their degree',
                'social medai & video','Travelling Time',
                'part-time job'],axis=1)
data


correlation_matrix=round(data.corr(),3)
correlation_matrix

import matplotlib.pyplot as plt
import seaborn as sns

fsize=plt.figure(figsize=(8,6))
sns.heatmap(correlation_matrix,annot=True,
            cmap='coolwarm',fmt='.2f',linewidths=1,
            square=True)
plt.title('Heatmap')
plt.tight_layout()
plt.show()
```

Python Code for Min Max Scaling

```
import pandas as pd
data=pd.read_csv("Student Attitude and Behavior.csv")
data.columns=data.columns.str.strip()
data.head(3)

x=list(data['Stress Level'])
#print(x)

data['Stress Level'].unique()

a=[]
for i in range(len(x)):
    if x[i]=='Awful':
        a.append(1)
    elif x[i]=='Bad':
        a.append(2)
    elif x[i]=='Good':
        a.append(3)
    else:
        a.append(4)
#print(a)

normalized_value=[]
for i in range(len(a)):
    min_a=min(a)
    max_a=max(a)
    numerator=a[i]-min_a
    denominator=max_a-min_a
    scaled_value=round(numerator/denominator,2)
    normalized_value.append(scaled_value)
#print(normalized_value)

data['Stress Level']=normalized_value
data.head(6)
```

When to use boxplot??

When we are dealing with the comparison with two columns where one column consists of nominal values and other column consists of numeric value,then we make box plot for the numeric values of numeric column for the corresponding nominal cases of nominal column.For example,we have Student Attitude and Behavior.csv dataset from kaggle.We compare the college mark between the two nominal factor;college mark who have certification course (Certification course=yes) and the college mark who don't have certification (Certification course=no)course.We found that who have certification course (Certification course=yes) has greater college mark than who have not any certification course.

Similarly we can Compare the college mark of male and Female by creating box plot.

Also we can compare the college mark of those students who have different hobies.

Finally we can say that to comapre the academic performnce according to nominal factos like Gender(male,female) hobies(reading,gardening,...) and so on we use box plot

When not to use Box Plot??

When we are finding the impact of a feature (which consists of numeric value or ordinal values(not nominal values)) on our target feature or on another feature we can not use box plot-then we need to find correlation coefficient.

When to use Correlation Coefficient??

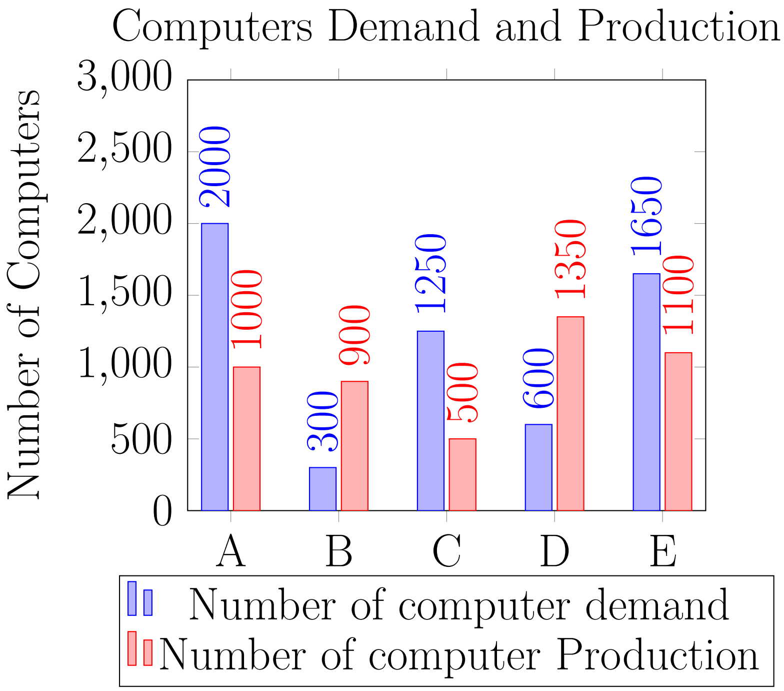
When we are dealing with the association of two numeric columns like the effect of student's height/weight/salary expectation etc on college mark,we find correaltion coefficient.

When not to use Correlation Coefficient??

When we are dealing with the comparison with two columns where one column consists of nominal values and other column consists of numeric value,We don't use correlation coefficient.Then We use box plot for the comparison of each category of the nominal column with our target feature.

✠ Bar Graphs

Example-01: Data in respect of Demand and Production of Computers of five companies.A-E are shown in the figure.Based on the data,answer questions that follow.



1.The ratio of the number of the companies having more production than of companies having more demand than production is,

1. 3:2
2. 1:4
3. 4:1
4. 2:3

Python Code for Removing Duplicate items from a list or column and Counting frequency of the categories in a list or column

```python
import pandas as pd
data=pd.read_csv("Student Attitude and Behavior.csv")
data.columns=data.columns.str.strip()
data.head(3)
x=list(data['hobbies'])

#Removing duplicates elements from a list or column
list_items=[]
for element in x:
    if element==element:
        if element not in list_items:
            list_items.append(element)
print(list_items)


#Counting frequency of specific categories in a list or column
item_count=[]

count_1=0
count_2=0
count_3=0
count_4=0

for element in x:
    if element=='Video Games':
        count_1+=1
    elif element=='Cinema':
        count_2+=1
    elif element=='Reading books':
        count_3+=1
    else:
        count_4+=1

item_count.append(('Video Games',count_1))
item_count.append(('Cinema',count_2))
item_count.append(('Reading books',count_3))
item_count.append(('Sports',count_4))

print(item_count)

dict_item_count=dict(item_count)
print(dict_item_count)
```

Python Code for Bar Graph

```
item_count=[('Video Games', 36), ('Cinema', 78),
            ('Reading books', 36), ('Sports', 85)]
labels=[]
values=[]
for i,j in item_count:
    labels.append(i)
    values.append(j)
print(labels)
print(values)

import matplotlib.pyplot as plt

plt.figure(figsize=(8,6))

bar=plt.bar(labels,values,color=['red','blue',
                                 'yellow','green'])
plt.title("Bar Plot")
plt.xlabel('Hobbies')
plt.ylabel('Frequency')
plt.grid(axis='y',linestyle='--',alpha=0.7)


for b in bar:
    yval=b.get_height()
    #print(yval)
    plt.text(b.get_x()+b.get_width()/2,
             yval + 1, yval, ha='center',
             va='bottom', fontsize=10)


plt.tight_layout()
plt.show()
```

Descending order of List Element in Python

```
x=[1,3,4,5,2,8,9,3,4,8,7,6,5]
descending=[]
for i in range(len(x)):
    maxi=max(x)
    descending.append(maxi)
    x.remove(maxi)
print(descending)
```

Python Code Descending Order of special type of list(dictionary type list) according to values

```python
item_count=[('Video Games', 36), ('Cinema', 78),
            ('Reading books', 36), ('Sports', 85)]
labels=[]
values=[]
for i,j in item_count:
    labels.append(i)
    values.append(j)
print(labels)
print(values)

descending=[]
x=values

for i in range(len(x)):
    maxi=max(x)
    descending.append(maxi)
    x.remove(maxi)
print(descending)



c=[]
for key in descending:
    for i in range(len(item_count)):
        if key==item_count[i][1]:
            if item_count[i] not in c:
                c.append(item_count[i])
                break
print(c)
```

Python Code for Horizontal Bar Chart(Barplot)

```
import pandas as pd
data=pd.read_csv("Student Attitude and Behavior.csv")
data.head(2)

correlation1 = round(data['Height(CM)'].corr(data['college mark']),2)
correlation2 = round(data['Weight(KG)'].corr(data['college mark']),2)
correlation3 = round(data['10th Mark'].corr(data['college mark']),2)
correlation4 = round(data['12th Mark'].corr(data['college mark']),2)
correlation5 = round(data['salary expectation'].
                     corr(data['college mark']),2)

#print(correlation1,correlation2,correlation3,correlation4)

features=['Height(CM)','Weight(KG)','10th Mark',
          '12th Mark','salary expectation']
correlational_value=[correlation1,correlation2,
                     correlation3,correlation4,correlation5]
#print(correlational_value)


#Descending order of List Element in Python
descending=[]
for i in range(len(correlational_value)):
    maxi=max(correlational_value)
    descending.append(maxi)
    correlational_value.remove(maxi)
print(descending)

corr_value=descending


import matplotlib.pyplot as plt
import seaborn as sns


plt.figure(figsize=(10,6))
sns.barplot(x=corr_value,y=features,palette='viridis')
plt.title("Correlation of features with College Mark")
plt.xlabel("Correlation coefficient")
plt.ylabel("Features")
plt.tight_layout()
plt.show()
```

Python Code to Separate Numeric Columns of a Dataset Using Advance Tools

```
import pandas as pd
data=pd.read_csv("Student Attitude and Behavior.csv")
data.head(1)

numeric_data=data.select_dtypes(include=['number'])
numeric_data
```

Python Code to Separate Numeric Columns of a Dataset Using Loop

```
import pandas as pd
data=pd.read_csv('Student Attitude and Behavior.csv')
data.head(1)


matching_column=[]

for i in range(len(list(data['Certification Course']))):
    rows=data.iloc[i]
    #print(rows)
    for col_name,value in rows.items():
        if isinstance(value, (int, float)):
            if col_name not in matching_column:
                matching_column.append(col_name)
print(matching_column)

x=data[matching_column]
x
```

Extracting Columns Based on a target value in a Specific Row using Pandas

```
import pandas as pd
data=pd.read_csv("Student Attitude and Behavior.csv")
data.head(1)

row1=data.iloc[1]
print(row1)

target_value=70.0

matching_column=[]

for col_name,value in row1.items():
    if value==target_value:
        matching_column.append(col_name)
print(matching_column)


x=data[matching_column]
x
```

Extracting Columns Based on a target value which exists on any of Row of entire dataset using Pandas

```
import pandas as pd
data=pd.read_csv('Student Attitude and Behavior.csv')
data.head(1)


target_value=25000

matching_column=[]

for i in range(len(list(data['Certification Course']))):
    rows=data.iloc[i]
    #print(rows)
    for col_name,value in rows.items():
        if value==target_value:
            if col_name not in matching_column:
                matching_column.append(col_name)
print(matching_column)

x=data[matching_column]
x
```

✠ PCA

Principal Component Analysis is a dimentionality reduction technique used in unsupervised learning to transform a dataset with many variables into a smaller set that still contains most of the essential information

◆The steps of the algorithm are,

☞ **Step-1:** Calculate mean of every features

☞ **Step-2:** Calculate the covariance matrix

$$\text{Cov}=\begin{bmatrix} Var(X_1) & Cov(X_2,X_1) \\ Cov(X_1,X_2) & Var(X_2) \end{bmatrix}$$

☞ **Step-3:**Calculate the Eigenvalues and Eigenvectors

☞ **Step-4:** Choose Principal Component

☞ **Step-5**Project data onto principal Component

Example-1:

Size(in canal)-X_1	No.of Rooms-X_2
5	10
3	7
10	15
2	4

✚ **Source: Click On-✈**

◆Step by Step Solution,

☞ **Step-1:** Calculate mean

$$\overline{X_1} = 5$$
$$\overline{X_2} = 9$$

☞ **Step-2:** Covariance Matrix

$$\text{Cov matrix} = \begin{bmatrix} \text{Var}(X_1) & \text{Cov}(X_2, X_1) \\ \text{Cov}(X_1, X_2) & \text{Var}(X_2) \end{bmatrix}$$

$$\text{Var}(X_1) = \frac{(5-5)^2 + (3-5)^2 + (10-5)^2 + (2-5)^2}{4-1} = 12.67$$

$$\text{Var}(X_2) = \frac{(10-9)^2 + (7-9)^2 + (15-9)^2 + (4-9)^2}{4-1} = 22$$

$$\text{Cov}(X_1, X_2) = \frac{(5-5)(10-9) + (3-5)(7-9) + (10-5)(15-9) +}{4-1}$$

$$\text{Cov}(X_2, X_1) = 16.33$$

$$\text{Cov Matrix} = \begin{bmatrix} 12.67 & 16.33 \\ 16.33 & 22 \end{bmatrix}$$

☞ **Step-3:** Calculate Eigenvalues

$$\begin{vmatrix} 12.67-\lambda & 16.33 \\ 16.33 & 22-\lambda \end{vmatrix} = 0$$

$$\lambda_1 = 34.3$$
$$\lambda_2 = -0.4$$

Calculate Eigenvectors:

$$\begin{bmatrix} 12.67-\lambda_1 & 16.33 \\ 16.33 & 22-\lambda_2 \end{bmatrix} \cdot \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} -1.3256 \\ 1 \end{bmatrix}$$

☞ Principal Component

$$PC_1 = \begin{bmatrix} 5 & 10 \end{bmatrix} \begin{bmatrix} -1.3256 \\ 1 \end{bmatrix} = 3.37$$

$$PC_2 = \begin{bmatrix} 3 & 7 \end{bmatrix} \begin{bmatrix} -1.3256 \\ 1 \end{bmatrix} = 3$$

$$PC_3 = \begin{bmatrix} 10 & 15 \end{bmatrix} \begin{bmatrix} -1.3256 \\ 1 \end{bmatrix} = 1.8$$

$$PC_4 = \begin{bmatrix} 2 & 4 \end{bmatrix} \begin{bmatrix} -1.3256 \\ 1 \end{bmatrix} = 1.3$$

Finally The Table Updated with PC(Principal Component) is given below:

Size(in canal)-X_1	No.of Rooms-X_2	PC
5	10	3.37
3	7	3
10	15	1.8
2	4	1.3

cumulative Sum Using Loop:Finding Covariance of a column with itself

```
import pandas as pd
data=pd.read_csv('Student Attitude and Behavior.csv')
data.columns=data.columns.str.strip()
data.head(1)

x=[]
for i in range(len(data['Gender'])):
    rows=data.iloc[i]
    for col_name,value in rows.items():
        if isinstance(value,(int,float)):
            if col_name not in x:
                x.append(col_name)
x

y=data[x]
y
y.dropna()

import numpy as np
mean_h=np.mean(data['Height(CM)'])
mean_w=np.mean(data['Weight(KG)'])
mean_10th=np.mean(data['10th Mark'])
mean_12th=np.mean(data['12th Mark'])
mean_c=np.mean(data['college mark'])


print(mean_h,mean_w,mean_10th,mean_12th,mean_c)

height=list(data['Height(CM)'])
x=height
a=0
for i in range(len(x)):
    a=a+(x[i]-mean_h)**2
print(a)

cov11=a/(len(x)-1)
cov11
```

Covariance between two different coulmns

```
import pandas as pd
data=pd.read_csv('Student Attitude and Behavior.csv')
data.columns=data.columns.str.strip()
data.head(1)

x=[]
for i in range(len(data['Gender'])):
    rows=data.iloc[i]
    for col_name,value in rows.items():
        if isinstance(value,(int,float)):
            if col_name not in x:
                x.append(col_name)
x

y=data[x]
y
y.dropna()

import numpy as np
mean_h=np.mean(data['Height(CM)'])
mean_w=np.mean(data['Weight(KG)'])
mean_10th=np.mean(data['10th Mark'])
mean_12th=np.mean(data['12th Mark'])
mean_c=np.mean(data['college mark'])


print(mean_h,mean_w,mean_10th,mean_12th,mean_c)

#cov_12
x=list(data['Height(CM)'])
y=list(data['Weight(KG)'])

sum_val=0
for i in range(len(x)):
    a=(x[i]-mean_h)
    b=(y[i]-mean_w)
    sum_val=sum_val+a*b
print(sum_val)

cov12=round(sum_val/(len(x)-1),2)
cov12
```

Covariance of a specific column with the other all coulumn of the dataset (including covariance of that specific coulumn with itself) Part-1

```
import pandas as pd
data=pd.read_csv('Student Attitude and Behavior.csv')
data.columns=data.columns.str.strip()
data.head(1)

x=[]
for i in range(len(data['Gender'])):
    rows=data.iloc[i]
    for col_name,value in rows.items():
        if isinstance(value,(int,float)):
            if col_name not in x:
                x.append(col_name)
x

y=data[x]
y
y.dropna()
```

Covariance of a specific column with the other all coulumn of the dataset (including covariance of that specific coulumn with itself) Part-2

```python
import numpy as np
mean_h=np.mean(data['Height(CM)'])
mean_w=np.mean(data['Weight(KG)'])
mean_10th=np.mean(data['10th Mark'])
mean_12th=np.mean(data['12th Mark'])
mean_c=np.mean(data['college mark'])

u=list(data['Height(CM)'])
v=list(data['Weight(KG)'])
w=list(data['10th Mark'])
x=list(data['12th Mark'])
y=list(data['college mark'])
mean=[mean_h,mean_w,mean_10th,mean_12th,mean_c]
features=[u,v,w,x,y]

b = []
for i in range(len(u)):
    r = []
    for j in range(5):
        #wrong: a = (u[i] - mean[j]) * (u[i] - mean[j])
        a = (u[i] - mean_h) * (features[j][i] - mean[j])
        a = round(a, 2)
        #print(a)
        r.append(a)
    b.append(r)
#print(r)
#print(b)

p=[]

for i in range(5):
    s=0
    for j in range(len(u)):
        s=s+b[j][i]
        s=round(s,2)
    p.append(s)
print(p)

cov_first_row=[]
for i in range(len(p)):
    a=p[i]/(len(u)-1)
    a=round(a,2)
    cov_first_row.append(a)

print(cov_first_row)
```

First row of covariance matrix:Covariance of first column with the other all coulumn of the dataset (including covariance of first coulumn with itself) Part-01

```python
import pandas as pd
data=pd.read_csv('Student Attitude and Behavior.csv')
data.columns=data.columns.str.strip()
data.head(1)

x=[]
for i in range(len(data['Gender'])):
    rows=data.iloc[i]
    for col_name,value in rows.items():
        if isinstance(value,(int,float)):
            if col_name not in x:
                x.append(col_name)
x

y=data[x]
y
y.dropna()
```

First row of covariance matrix:Covariance of first column with the other all coulumn of the dataset (including covariance of first coulumn with itself) Part-02

```python
import numpy as np
mean_h=np.mean(data['Height(CM)'])
mean_w=np.mean(data['Weight(KG)'])
mean_10th=np.mean(data['10th Mark'])
mean_12th=np.mean(data['12th Mark'])
mean_c=np.mean(data['college mark'])

u=list(data['Height(CM)'])
v=list(data['Weight(KG)'])
w=list(data['10th Mark'])
x=list(data['12th Mark'])
y=list(data['college mark'])
mean=[mean_h,mean_w,mean_10th,mean_12th,mean_c]
features=[u,v,w,x,y]

b = []
for i in range(len(u)):
    r = []
    for j in range(5):
        #wrong: a = (u[i] - mean[j]) * (u[i] - mean[j])
        a = (u[i] - mean_h) * (features[j][i] - mean[j])
        a = round(a, 2)
        #print(a)
        r.append(a)
    b.append(r)
#print(r)
#print(b)

p=[]
for i in range(5):
    s=0
    for j in range(len(u)):
        s=s+b[j][i]
        s=round(s,2)
    p.append(s)
print(p)

cov_first_row=[]
for i in range(len(p)):
    a=p[i]/(len(u)-1)
    a=round(a,2)
    cov_first_row.append(a)
print(cov_first_row)
```

Second row of covariance matrix:Covariance of Second column with the other all coulumn of the dataset (including covariance of Second coulumn with itself)

```python
import numpy as np
mean_h=np.mean(data['Height(CM)'])
mean_w=np.mean(data['Weight(KG)'])
mean_10th=np.mean(data['10th Mark'])
mean_12th=np.mean(data['12th Mark'])
mean_c=np.mean(data['college mark'])

u=list(data['Height(CM)'])
v=list(data['Weight(KG)'])
w=list(data['10th Mark'])
x=list(data['12th Mark'])
y=list(data['college mark'])
mean=[mean_h,mean_w,mean_10th,mean_12th,mean_c]
features=[u,v,w,x,y]

b = []
for i in range(len(u)):
    r = []
    for j in range(5):
        #wrong: a = (u[i] - mean[j]) * (u[i] - mean[j])
        a = (v[i] - mean_w) * (features[j][i] - mean[j])
        a = round(a, 2)
        #print(a)
        r.append(a)
    b.append(r)
#print(r)
#print(b)

p=[]

for i in range(5):
    s=0
    for j in range(len(u)):
        s=s+b[j][i]
        s=round(s,2)
    p.append(s)
print(p)

cov_second_row=[]
for i in range(len(p)):
    a=p[i]/(len(u)-1)
    a=round(a,2)
    cov_second_row.append(a)

print(cov_second_row)
```

Third row of covariance matrix:Covariance of Third column with the other all coulumn of the dataset (including covariance of Third coulumn with itself)

```
import numpy as np
mean_h=np.mean(data['Height(CM)'])
mean_w=np.mean(data['Weight(KG)'])
mean_10th=np.mean(data['10th Mark'])
mean_12th=np.mean(data['12th Mark'])
mean_c=np.mean(data['college mark'])

u=list(data['Height(CM)'])
v=list(data['Weight(KG)'])
w=list(data['10th Mark'])
x=list(data['12th Mark'])
y=list(data['college mark'])
mean=[mean_h,mean_w,mean_10th,mean_12th,mean_c]
features=[u,v,w,x,y]

b = []
for i in range(len(u)):
    r = []
    for j in range(5):
        #wrong: a = (u[i] - mean[j]) * (u[i] - mean[j])
        a = (w[i] - mean_10th) * (features[j][i] - mean[j])
        a = round(a, 2)
        #print(a)
        r.append(a)
    b.append(r)
#print(r)
#print(b)

p=[]
for i in range(5):
    s=0
    for j in range(len(u)):
        s=s+b[j][i]
        s=round(s,2)
    p.append(s)
print(p)

cov_third_row=[]
for i in range(len(p)):
    a=p[i]/(len(u)-1)
    a=round(a,2)
    cov_third_row.append(a)
print(cov_third_row)
```

Fourth row of covariance matrix:Covariance of Fourth column with the other all coulumn of the dataset (including covariance of Fourth coulumn with itself)

```
import numpy as np
mean_h=np.mean(data['Height(CM)'])
mean_w=np.mean(data['Weight(KG)'])
mean_10th=np.mean(data['10th Mark'])
mean_12th=np.mean(data['12th Mark'])
mean_c=np.mean(data['college mark'])

u=list(data['Height(CM)'])
v=list(data['Weight(KG)'])
w=list(data['10th Mark'])
x=list(data['12th Mark'])
y=list(data['college mark'])
mean=[mean_h,mean_w,mean_10th,mean_12th,mean_c]
features=[u,v,w,x,y]

b = []
for i in range(len(u)):
    r = []
    for j in range(5):
        #wrong: a = (u[i] - mean[j]) * (u[i] - mean[j])
        a = (x[i] - mean_12th) * (features[j][i] - mean[j])
        a = round(a, 2)
        #print(a)
        r.append(a)
    b.append(r)
#print(r)
#print(b)

p=[]
for i in range(5):
    s=0
    for j in range(len(u)):
        s=s+b[j][i]
        s=round(s,2)
    p.append(s)
print(p)

cov_fourth_row=[]
for i in range(len(p)):
    a=p[i]/(len(u)-1)
    a=round(a,2)
    cov_fourth_row.append(a)
print(cov_fourth_row)
```

Fifth row of covariance matrix:Covariance of Fifth column with the other all coulumn of the dataset (including covariance of Fifth coulumn with itself)

```
import numpy as np
mean_h=np.mean(data['Height(CM)'])
mean_w=np.mean(data['Weight(KG)'])
mean_10th=np.mean(data['10th Mark'])
mean_12th=np.mean(data['12th Mark'])
mean_c=np.mean(data['college mark'])

u=list(data['Height(CM)'])
v=list(data['Weight(KG)'])
w=list(data['10th Mark'])
x=list(data['12th Mark'])
y=list(data['college mark'])
mean=[mean_h,mean_w,mean_10th,mean_12th,mean_c]
features=[u,v,w,x,y]

b = []
for i in range(len(u)):
    r = []
    for j in range(5):
        #wrong: a = (u[i] - mean[j]) * (u[i] - mean[j])
        a = (y[i] - mean_c) * (features[j][i] - mean[j])
        a = round(a, 2)
        #print(a)
        r.append(a)
    b.append(r)
#print(r)
#print(b)

p=[]
for i in range(5):
    s=0
    for j in range(len(u)):
        s=s+b[j][i]
        s=round(s,2)
    p.append(s)
print(p)

cov_fifth_row=[]
for i in range(len(p)):
    a=p[i]/(len(u)-1)
    a=round(a,2)
    cov_fifth_row.append(a)
print(cov_fifth_row)
```

Covariance Matrix

```
'''from previous code:
 cov_first_row=[462.72, 88.42, 5.6, -28.35, -6.27]
 cov_second_row=[88.42, 221.89, 12.24, -3.14, -3.49]
 cov_third_row=[5.6, 12.24, 170.24, 68.04, 95.6]
 cov_fourth_row=[-28.35, -3.14, 68.04, 121.4, 73.62]
 cov_fifth_row=[-6.27, -3.49, 95.6, 73.62, 247.35]

 cov=[[462.72, 88.42, 5.6, -28.35, -6.27],
 [88.42, 221.89, 12.24, -3.14, -3.49],
 [5.6, 12.24, 170.24, 68.04, 95.6],
 [-28.35, -3.14, 68.04, 121.4, 73.62],
 [-6.27, -3.49, 95.6, 73.62, 247.35]]
 '''


cov=[cov_first_row,cov_second_row,cov_third_row,
     cov_fourth_row,cov_fifth_row]
feature=['Height(CM)','Weight(KG)','10th Mark',
         '12th Mark','college mark']
lis=[]
for i in range(len(cov)):
    lis.append((feature[i],cov[i]))
print(lis)
dic=dict(lis)
dic

cov_matrix=pd.DataFrame(dic)
cov_matrix

import numpy as np
matrix=np.array(cov)
matrix
```

display a 3×3 matrix in the typical paper-style format

Display n×n matrix in the typical paper-style format

```
import numpy as np

# Example 55 matrix (replace this with your actual data)
matrix = np.array([
    [1.00, 2.00, 3.00, 4.00, 5.00],
    [1.10, 2.20, 3.30, 4.40, 5.50],
    [0.90, 1.80, 2.70, 3.60, 4.50],
    [1.25, 2.50, 3.75, 5.00, 6.25],
    [0.95, 1.90, 2.85, 3.80, 4.75]
])

# Unicode brackets: top, middle, bottom
left_brackets  = [' ', '   ', '   ', '   ', '   ']
right_brackets = [' ', '   ', '   ', '   ', '   ']

# Print matrix with matching brackets
for i, row in enumerate(matrix):
    left = left_brackets[i]
    right = right_brackets[i]
    row_str = "  ".join(f"{val:.2f}" for val in row)
    print(f"{left} {row_str} {right}")
```

Eigenvalue of 2×2 Matrix

```
a=[[2,5],[3,9]]

trace=a[0][0]+a[1][1]
print(trace)

determinant=a[0][0]*a[1][1]-a[0][1]*a[1][0]
print(determinant)

# Solve quadratic: lamda^2 - trace*lamda + determinant = 0

import math
lamda1=(trace+math.sqrt((trace**2)-(4*1*determinant)))/(2*1)
print(lamda1)

lamda2=(trace-math.sqrt((trace**2)-(4*1*determinant)))/(2*1)
print(lamda2)
```

Eigenvalues of 3 ×3 Matrix

```
a=[[1,4,3],[5,7,2],[9,0,5]]
a

t=0
for i in range(len(a)):
    t=t+a[i][i]
trace=t
print(trace)

cofactor1=a[1][1]*a[2][2]-a[1][2]*a[2][1]
print(cofactor1)
cofactor2=(-1)*(a[1][0]*a[2][2]-a[1][2]*a[2][0])
print(cofactor2)
cofactor3=a[1][0]*a[2][1]-a[1][1]*a[2][0]
print(cofactor3)

cofactor4=(-1)*(a[0][1]*a[2][2]-a[0][2]*a[2][1])
print(cofactor4)
cofactor5=(1)*(a[0][0]*a[2][2]-a[0][2]*a[2][0])
print(cofactor5)
cofactor6=(-1)*(a[0][0]*a[2][1]-a[0][1]*a[2][0])
print(cofactor6)

cofactor7=(+1)*(a[0][1]*a[1][2]-a[0][2]*a[1][1])
print(cofactor7)

cofactor8=(-1)*(a[0][0]*a[1][2]-a[0][2]*a[1][0])
print(cofactor8)

cofactor9=(+1)*(a[0][0]*a[1][1]-a[0][1]*a[1][0])
print(cofactor9)

sum_of_diogonal_cofactor=cofactor1+cofactor5+cofactor9
print(sum_of_diogonal_cofactor)

determinant=a[0][0]*cofactor1+a[0][1]*cofactor2+a[0][2]*cofactor3
print(determinant)

#equation for eigenvalue: eigenvalue=lamda
#lamda^3-trace*lamda^2+sum_of_diogonal_cofactor*lamda-determinant=0

import numpy as np
coefficient=[1,-trace,sum_of_diogonal_cofactor,-determinant]
roots_are=np.roots(coefficient)
print(roots_are)

eigenvalues=list(roots_are)
print(eigenvalues)
```

Rezaul Karim Rifat, SDS-JU

Cofactor of first row of 3×3 Matrix

```
def get_minor(matrix,row,col):
    minor=[]
    for i in range(len(matrix)):
        if i!=row:
            row_val=[]
            for j in range(len(matrix)):
                if j!=col:
                    row_val.append(matrix[j][i])
            minor.append(row_val)
    return minor

def determinant_2by2(m):
    det=m[0][0]*m[1][1]-m[0][1]*m[1][0]
    return det


a=[[1,4,3],
   [5,7,2],
   [9,0,5]]
#print(a)

j=0 #Cofactor for first row j=0
for i in range(len(a)):
    minor_val=get_minor(a,i,j)
    cofactor=((-1)**(i+j))*determinant_2by2(minor_val)
    print(f'Cofactor of the element[{0}][{i}]=',cofactor)
```

Cofactor for the first column of 3×3 Matrix

```python
def get_minor(matrix,row,col):
    minor=[]
    for i in range(len(matrix)):
        if i!=row:
            row_val=[]
            for j in range(len(matrix)):
                if j!=col:
                    row_val.append(matrix[i][j])
            minor.append(row_val)
    return minor

def determinant_2by2(m):
    det=m[0][0]*m[1][1]-m[0][1]*m[1][0]
    return det


a=[[1,4,3],
   [5,7,2],
   [9,0,5]]
#print(a)

j=0 #Cofactor for first column j=0
    #Cofactor for Second column j=1
    #Cofactor for Third column j=2
for i in range(len(a)):
    minor_val=get_minor(a,i,j)
    cofactor=((-1)**(i+j))*determinant_2by2(minor_val)
    print(f'Cofactor{i}=',cofactor)
```

Cofactors of all element of 3×3 Matrix

```python
def get_minor(matrix,row,col):
    minor=[]
    for i in range(len(matrix)):
        if i!=row:
            row_val=[]
            for j in range(len(matrix)):
                if j!=col:
                    row_val.append(matrix[i][j])
            minor.append(row_val)
    return minor

def determinant_2by2(m):
    det=m[0][0]*m[1][1]-m[0][1]*m[1][0]
    return det


a=[[1,4,3],
   [5,7,2],
   [9,0,5]]
#print(a)

s=[0,1,2]
for i in range(len(a)):
    for j in s:
        minor_val=get_minor(a,i,j)
        cofactor=((-1)**(i+j))*determinant_2by2(minor_val)
        print(f'Cofactor{i}=',cofactor)
```

Eigenvalue Calculation Using Numpy–np.linalg.eigvals()

```
import pandas as pd

cov_first_row=[462.72, 88.42, 5.6, -28.35, -6.27]
cov_second_row=[88.42, 221.89, 12.24, -3.14, -3.49]
cov_third_row=[5.6, 12.24, 170.24, 68.04, 95.6]
cov_fourth_row=[-28.35, -3.14, 68.04, 121.4, 73.62]
cov_fifth_row=[-6.27, -3.49, 95.6, 73.62, 247.35]

cov=[cov_first_row,cov_second_row,cov_third_row,
     cov_fourth_row,cov_fifth_row]
feature=['Height(CM)','Weight(KG)','10th Mark',
         '12th Mark','college mark']
lis=[]
for i in range(len(cov)):
    lis.append((feature[i],cov[i]))
print(lis)
dic=dict(lis)
print(dic)

cov_matrix=pd.DataFrame(dic)
print(cov_matrix)

n1=list(cov_matrix['Height(CM)'])
print(n1)
n2=list(cov_matrix['Weight(KG)'])
print(n1)
n3=list(cov_matrix['10th Mark'])
print(n1)
n4=list(cov_matrix['12th Mark'])
print(n1)
n5=list(cov_matrix['college mark'])
print(n1)

matrix=[n1,n2,n3,n4,n5]
matrix

import numpy as np
eigenvalues = np.linalg.eigvals(matrix)

# Print the result
print("Eigenvalues:")
print(eigenvalues)
```

Principal Component Analysis:Part-1

```python
import numpy as np
A = np.array([
    [-1020.71, 88.42, 5.6, -28.35, -6.27],
    [88.42, -838.82, 12.24, -3.14, -3.49],
    [5.6, 12.24, -411.01, 68.04, 95.6],
    [-28.35, -3.14, 68.04, -92.5, 73.62],
    [-6.27, -3.49, 95.6, 73.62, -84.16]
])

# Compute eigenvalues and eigenvectors
eigenvalues, eigenvectors = np.linalg.eig(A)

# Display results
print("Eigenvalues:\n", eigenvalues)
print("\nEigenvectors (columns):\n", eigenvectors)

# PCA finding Part
max_eval = max(eigenvalues)
print(max_eval)

x = []
for i in range(len(eigenvalues)):
    if np.isclose(eigenvalues[i], max_eval):
        for j in range(5):
            a = eigenvectors[j][i]
            a = round(a, 4)
            x.append(a)
print(x)

max_egv = max(x, key=lambda k: abs(k))
print(max_egv)

feature = ['Height(CM)', 'Weight(KG)',
           '10th Mark', '12th Mark', 'college mark']
a = []
for i in range(5):
    if abs(x[i]) == abs(max_egv):
        a.append(feature[i])

print('PCA-1 (most contributing feature):', a)
```

Principal Component Analysis:Part-2

```
# Finding second PC

th = list(eigenvalues)
th.remove(max_eval)

second_max_eval = max(th)
print(second_max_eval)

d = []
for i in range(5):
    if np.isclose(eigenvalues[i], second_max_eval):
        for j in range(5):
            s = eigenvectors[j][i]
            s = round(s, 4)
            d.append(s)
print(d)

max_for_pca2 = max(d, key=lambda k: abs(k))
print(max_for_pca2)

b = []
for i in range(5):
    if abs(d[i]) == abs(max_for_pca2):
        b.append(feature[i])

print('PCA-2:', b)
```

✠ K prototype Clustering

The K protype clustering algorithm is a partitional clustering algorithm designed to handle datasets with a mixure of numerical and categorical variables.Its objective is to partition a dataset into k cluster such thatthe within cluster distance is minimized.
The objective function of the k protypes algorithm is defined as:

$$\boldsymbol{E = \sum_{i=1}^{n} \sum_{j=1}^{k} \mu_{ij} d(x_i, \mu_j)}$$

Where,

- x_i are the observations in the sample.
- μ_j are the cluster prototype observations.
- μ_{ij} are the elements of the binary partition matrix satisfying $\Sigma_{j=1}^{k} \mu_{ij} = 1$ for all i.
- $d(x_i \mu_j)$ is the distance function between observation x_i and cluster prototype μ_j
  The distance function $d(x_i \mu_j)$ is defined as a weighted sum of Euclidean Distance for numeric variables and simple matching distance for categorical variables.

  $$d(x_i \mu_j) = \sum_{m=1}^{p} (x_i^m - \mu_j^m)^2 + \gamma \sum_{m=1}^{q} \delta(x_i^m, \mu_j^m)$$

  Where,

  - x_i:data point
  - μ_j:prototype(cluster center)
  - p:is number of numerical feature.
  - q:number of categorical features

- γ: weight or scaling factor to balance numerical and categorical contributions.
- $\delta(a, b)$: dissimilarity function for categorical attributes, typically:

$$\delta(a,b) = \begin{cases} 0; & \text{if } a = b \\ 1; & \text{if } a \neq b \end{cases}$$

◆The steps of the algorithm are,

1. Initialization with random cluster prototype
2. For each observation do,
    - Assign observations to its closest prototype according to d()
    - Update cluster prototype by cluster specific means/modes for all variables.
3. As long as any observations have swaped their cluster assignment in 2 or the maximum number of iterations ha not bee reached repeat from 2

◆ The modified k prototype algorithm consists of the following steps.

1. Initialization with random cluster prototype.
2. Assign all observations to its closest prototype according to d().
3. Update Cluster prototype.
4. As long as any observations have swapped their cluster assignment in 2 or the maximum number of iterations has not been reached: repeat from 2.

   Here,$\lambda = \dfrac{\sigma}{h_{cat}}$

Tutorial-48: Apply K prototype clustering algorithm to find the cluster solution for k=2 for following data on $X_1, X_2, ..., X_5$.Use the ID-7 and ID-11 as initial cluster prototypes.

ID	$\boldsymbol{X_1}$	$\boldsymbol{X_2}$	$\boldsymbol{X_3}$	$\boldsymbol{X_4}$	$\boldsymbol{X_5}$
1	13	14.5	5.7	F	P
2	12.2	10.3	5.8	M	C
3	13.3	13.9	5.7	F	P
4	13.3	15	6	F	P
5	12.7	12	5.8	M	C
6	13.4	15	5.7	F	P
7	12	10.7	5.7	M	C
8	12.3	10	5.8	M	C
9	12	9.5	5.8	M	C
10	12.7	11	5.8	M	C
11	13.7	15.2	5.8	F	P

Table 38: Caption

Solution:

✠ RNN

RNN is the topic of deep learning.It is specifically uses for sequnes. Previously we discussed ANN.It deals with mainly tabular data

Then CNN-Convolutional Neural Network which deals with images

Third kind of neural network is RNN which deals with sequential data

RNN=Recurrent Neural Network

RNN is type of sequential model specificly design to work on sequential data.

What is sequential data??

what is non sequential data??

References

[1] Geoffrey I. Webb. Naïve bayes. In Claude Sammut and Geoffrey I. Webb, editors, *Encyclopedia of Machine Learning*, pages 713–714. Springer, Boston, MA, 2010. Synonyms: Idiot's Bayes, Simple Bayes.

(a)

This is italic text.

rafi

My Famile

$\overline{Myfamily}$

$\overline{x}$

`My name is Rafi`

A correlation matrix in Python is a table displaying the pairwise correlation coefficients between multiple numerical variables in a dataset. It is a fundamental tool in data analysis, particularly for exploratory data analysis and feature selection in machine learning.

$$\boldsymbol{x_{scaled}} = \frac{\boldsymbol{x - x_{min}}}{\boldsymbol{x_{max} - x_{min}}}$$

END