Q 1. Perform the following operations using Python on a data set : read data from different formats(like csv, xls),indexing and selecting data, sort data, describe attributes of data, checking data types of each column. (Use Titanic Dataset).

```
import pandas as pd
import numpy as np;

df=pd.read_csv("titanic.csv")
df.info()

df_1=df['PassengerId'].mean()
print("Mean of PassengerID:",df_1)
indexing=df[4:7]
print("Indexing:",indexing)
df_2=df.head()
print(df_2)
df_3=df.describe()
print(df_3)
df_4=df.sort_values("PassengerId")
print("Sort_values:",df_4)
```

Q 2. Perform the following operations using Python on the Telecom_Churn dataset. Compute and display summary statistics for each feature available in the dataset using separate commands for each statistic. (e.g. minimum value, maximum value, mean, range, standard deviation, variance and percentiles).

```
import pandas as pd
import numpy as np
```

```
df=pd.read_csv("titanic.csv")


df_5=df['Fare'].min()
print("Minimum of fare:",df_5)
df_6=df['SibSp'].max()
print("Maximum of SibSp:",df_6)
df_7=(df['Fare'].max()-df['Fare'].min())
print("Range of FARE:",df_7)
df_8=df['Fare'].std()
print("S.D. of FARE:",df_8)
df_9=df['Fare'].var()
print("Variance of Fare:",df_9)
df_10=np.percentile(df['Fare'],100)
print("Percentile of Fare:",df_10)
```

Q 3. Perform the following operations using Python on the data set House_Price Prediction dataset. Compute standard deviation, variance and percentiles using separate commands, for each feature. Create a histogram for each feature in the dataset to illustrate the feature distributions.

```
import pandas as pd
import numpy as np
from copy import deepcopy
from matplotlib import pyplot as plt
df=pd.read_csv("titanic.csv")


plt.hist(df['Fare'],bins=30)
plt.ylabel('Fare price')
plt.show()
```

Q 4. Write a program to do: A dataset collected in a cosmetics shop showing details of customers and whether or not they responded to a special offer to buy a new lip-stick is shown in table below. (Implement step by step using commands - Dont use library) Use this dataset to build a decision tree, with Buys as the target variable, to help in buying lipsticks in the future. Find the root node of the decision tree.

Q 9. Write a program to do the following: You have given a collection of 8 points. P1=[0.1,0.6] P2=[0.15,0.71] P3=[0.08,0.9] P4=[0.16, 0.85] P5=[0.2,0.3] P6=[0.25,0.5] P7=[0.24,0.1] P8=[0.3,0.2]. Perform the k-mean clustering with initial centroids as m1=P1 =Cluster#1=C1 and m2=P8=cluster#2=C2. Answer the following 1] Which cluster does P6 belong to? 2] What is the population of a cluster around m2? 3] What is the updated value of m1 and m2?

```
import numpy as np
import pandas as pd
from copy import deepcopy
from matplotlib import pyplot as plt

#Given Dataset
dataset = {
'Points':['P1','P2','P3','P4','P5','P6','P7','P8',],
'x_coordinate':[0.1,0.15,0.08,0.16,0.2,0.25,0.24,0.3],
'y_coordinate':[0.6,0.71,0.9,0.85,0.3,0.5,0.1,0.2]
}
```

```
#dataframe
df = pd.DataFrame(dataset,columns=['Points','x_coordinate','y_coordinate'])
print(df)
# Getting the values and plotting it
f1 = df['x_coordinate'].values
f2 = df['y_coordinate'].values
X = np.array(list(zip(f1, f2)))
plt.scatter(f1, f2, c='black', s=7)

# Euclidean Distance Caculator
def dist(a, b, ax=1):
    return np.linalg.norm(a - b, axis=ax)
# Number of clusters
k = 2
# Two initia Centroids are given
# m1 = P1
# m2 = P8
Centroid_m1 = list(X[0])
Centroid_m2 = list(X[7])
Centroids = np.array([Centroid_m1,Centroid_m2])
print(Centroids)

# Plotting along with the Centroids
plt.scatter(f1, f2, c='#050505', s=7)
plt.scatter(Centroid_m1[0],Centroid_m1[1] ,marker='*', s=200, c='g')
plt.scatter(Centroid_m2[0],Centroid_m2[1] ,marker='*', s=200, c='g')

# To store the value of centroids when it updates
C_old = np.zeros(Centroids.shape)
# Cluster Lables(0, 1, 2)
clusters = np.zeros(len(X))
```

```
# Error func. - Distance between new centroids and old centroids
error = dist(Centroids, C_old, None)
# Loop will run till the error becomes zero
while error != 0:
   # Assigning each value to its closest cluster
   for i in range(len(X)):
      distances = dist(X[i], Centroids)
      cluster = np.argmin(distances)
      clusters[i] = cluster
   # Storing the old centroid values
   C_old = deepcopy(Centroids)
   print(C_old)
   # Finding the new centroids by taking the average value
   for i in range(k):
      points = [X[j] for j in range(len(X)) if clusters[j] == i]
      Centroids[i] = np.mean(points, axis=0)
      print(Centroids[i])
   error = dist(Centroids, C_old, None)
colors = ['r', 'g']
fig, ax = plt.subplots()
for i in range(k):
      points = np.array([X[j] for j in range(len(X)) if clusters[j] == i])
      ax.scatter(points[:, 0], points[:, 1], s=7, c=colors[i])
ax.scatter(Centroids[:, 0], Centroids[:, 1], marker='*', s=200, c='#050505')
plt.show()
```

Q 10. Write a program to do the following: You have given a collection of 8 points. P1=[2, 10] P2=[2, 5] P3=[8, 4] P4=[5, 8] P5=[7,5] P6=[6, 4] P7=[1, 2]

P8=[4, 9]. Perform the k-mean clustering with initial centroids as m1=P1 =Cluster#1=C1 and m2=P4=cluster#2=C2, m3=P7 =Cluster#3=C3. Answer the following 1] Which cluster does P6 belong to? 2] What is the population of a cluster around m3? 3] What is the updated value of m1, m2, m3?

```
import numpy as np
import pandas as pd
from copy import deepcopy
from matplotlib import pyplot as plt

#Given Dataset
dataset = {
'Points':['P1','P2','P3','P4','P5','P6','P7','P8',],
'x_coordinate':[2,2,8,5,7,6,1,4],
'y_coordinate':[10,5,4,8,5,4,2,9]
}


#dataframe
df = pd.DataFrame(dataset,columns=['Points','x_coordinate','y_coordinate'])
print(df)
# Getting the values and plotting it
f1 = df['x_coordinate'].values
f2 = df['y_coordinate'].values
X = np.array(list(zip(f1, f2)))
plt.scatter(f1, f2, c='black', s=7)

# Euclidean Distance Caculator
def dist(a, b, ax=1):
   return np.linalg.norm(a - b, axis=ax)
# Number of clusters
k = 2
# Two initia Centroids are given
# m1 = P1
# m2 = P8
Centroid_m1 = list(X[0])
Centroid_m2 = list(X[7])
Centroids = np.array([Centroid_m1,Centroid_m2])
print(Centroids)

# Plotting along with the Centroids
plt.scatter(f1, f2, c='#050505', s=7)
plt.scatter(Centroid_m1[0],Centroid_m1[1] ,marker='*', s=200, c='g')
plt.scatter(Centroid_m2[0],Centroid_m2[1] ,marker='*', s=200, c='g')

# To store the value of centroids when it updates
C_old = np.zeros(Centroids.shape)
# Cluster Lables(0, 1, 2)
clusters = np.zeros(len(X))
# Error func. - Distance between new centroids and old centroids
error = dist(Centroids, C_old, None)
# Loop will run till the error becomes zero
while error != 0:
    # Assigning each value to its closest cluster
    for i in range(len(X)):
        distances = dist(X[i], Centroids)
        cluster = np.argmin(distances)
        clusters[i] = cluster
    # Storing the old centroid values
    C_old = deepcopy(Centroids)
    print(C_old)
    # Finding the new centroids by taking the average value
    for i in range(k):
        points = [X[j] for j in range(len(X)) if clusters[j] == i]
        Centroids[i] = np.mean(points, axis=0)
        print(Centroids[i])
    error = dist(Centroids, C_old, None)
colors = ['r', 'g']
fig, ax = plt.subplots()
for i in range(k):
        points = np.array([X[j] for j in range(len(X)) if clusters[j] == i])
        ax.scatter(points[:, 0], points[:, 1], s=7, c=colors[i])
ax.scatter(Centroids[:, 0], Centroids[:, 1], marker='*', s=200, c='#050505')
plt.show()
```

Q 11. Use Iris flower dataset and perform following :

1. List down the features and their types (e.g., numeric, nominal) available in the dataset. 2. Create a histogram for each feature in the dataset to illustrate the feature distributions.

```
#Import Libraries
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns

#Read CSV File
dat = pd.read_csv("Iris.csv")
#Return First Five Rows
print(dat.head())
print(dat.columns)
print(dat.SepalLengthCm)
dat_1=dat.sort_values('SepalLengthCm',ascending= True)
print(dat_1)
dat_2=dat.isnull().sum()
print(dat_2)
```

```
replace_median = dat['SepalLengthCm'].median()
print(replace_median)
print(dat['SepalLengthCm'].fillna(replace_median, inplace=True))

dat_3=dat.corr()
print("Correlation:",dat_3)

dat_4=dat.groupby('SepalLengthCm').describe()
print(dat_4)
dat.hist(bins=50, figsize=(15,10))
plt.show()

plt.hist(dat['SepalLengthCm'],bins=30)
plt.ylabel('No of items')
plt.show()

sns.boxplot(y = dat['SepalLengthCm'])
plt.show()
sns.boxplot(x = dat['Species'], y = dat['SepalWidthCm'])
plt.show()
```

Q 13. Use the covid_vaccine_statewise.csv dataset and perform the following analytics.

a. Describe the dataset

b. Number of persons state wise vaccinated for first dose in India

c. Number of persons state wise vaccinated for second dose in India

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

```
dataset = pd.read_csv("covid_vaccine_statewise.csv")
print(dataset.head(10))



dataset.tail(10)
dataset.columns
dataset.describe()

dataset["State"].unique()

dataset.shape
dict = {}
for state in dataset["State"].unique():
  dict[state] = 0
for i in range(dataset.shape[0]):
   dict[dataset["State"].iloc[i]] += dataset["First Dose Administered"].iloc[i]

dataset.dtypes
dataset.isnull().sum()

dataset.fillna(value=0, inplace=True)
dataset.head()



dataset.isnull().sum()
dataset['Total Doses Administered'] = dataset['Total Doses Administered'].astype('int64')
dataset['Sessions'] = dataset['Sessions'].astype('int64')
dataset['First Dose Administered'] = dataset['First Dose Administered'].astype('int64')
dataset['Second Dose Administered'] = dataset['Second Dose Administered'].astype('int64')
```

```
dataset['Male (Doses Administered)'] = dataset['Male (Doses Administered)'].astype('int64')
dataset['Female (Doses Administered)'] = dataset['Female (Doses Administered)'].astype('int64')
dataset['Transgender (Doses Administered)'] = dataset['Transgender (Doses Administered)'].astype('int64')
dataset[' Covaxin (Doses Administered)'] = dataset[' Covaxin (Doses Administered)'].astype('int64')
dataset['CoviShield (Doses Administered)'] = dataset['CoviShield (Doses Administered)'].astype('int64')
dataset['Sputnik V (Doses Administered)'] = dataset['Sputnik V (Doses Administered)'].astype('int64')
dataset['AEFI'] = dataset['AEFI'].astype('int64')
dataset['18-44 Years (Doses Administered)'] = dataset['18-44 Years (Doses Administered)'].astype('int64')
dataset['45-60 Years (Doses Administered)'] = dataset['45-60 Years (Doses Administered)'].astype('int64')
dataset['60+ Years (Doses Administered)'] = dataset['60+ Years (Doses Administered)'].astype('int64')
dataset['18-44 Years(Individuals Vaccinated)'] = dataset['18-44 Years(Individuals Vaccinated)'].astype('int64')
dataset['45-60 Years(Individuals Vaccinated)'] = dataset['45-60 Years(Individuals Vaccinated)'].astype('int64')
dataset['60+ Years(Individuals Vaccinated)'] = dataset['60+ Years(Individuals Vaccinated)'].astype('int64')
dataset['Male(Individuals Vaccinated)'] = dataset['Male(Individuals Vaccinated)'].astype('int64')
dataset['Female(Individuals Vaccinated)'] = dataset['Female(Individuals Vaccinated)'].astype('int64')
dataset['Transgender(Individuals Vaccinated)'] = dataset['Transgender(Individuals Vaccinated)'].astype('int64')
dataset['Total Individuals Vaccinated'] = dataset['Total Individuals Vaccinated'].astype('int64')


dataset.dtypes


date = dataset['Updated On'].str.split('/', expand=True)
date
dataset['day'] = date[0]
dataset['month'] = date[1]
```

```
dataset['year'] = date[2]

dataset['day'] = pd.to_numeric(dataset['day'])
dataset['month'] = pd.to_numeric(dataset['month'])
dataset['year'] = pd.to_numeric(dataset['year'])

dataset['Updated On'] = pd.to_datetime(dataset['Updated On'])
dataset.rename(columns = {'Updated On' : 'Vaccine_date'}, inplace = True)

dataset.head()

dataset.describe()

dataset['State'].unique()

new_dataset = dataset

statewise_vaccination = new_dataset.groupby('State')['Total Individuals Vaccinated'].sum().to_frame('Total Individuals Vaccinated')
statewise_vaccination



statewise_vaccination.plot(kind='bar', figsize=(17,10))
plt.ylabel('Number of people vaccinated')
plt.show()

statewise_vaccination_dose1 = new_dataset[['State','First Dose Administered']].groupby('State').sum()
statewise_vaccination_dose1

statewise_vaccination_dose2 = new_dataset[['State','Second Dose Administered']].groupby('State').sum()
```

```
statewise_vaccination_dose2
```

Q 14. Use the covid_vaccine_statewise.csv dataset and perform the following analytics.

A. Describe the dataset.

B. Number of Males vaccinated

C.. Number of females vaccinated

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt



dataset = pd.read_csv("covid_vaccine_statewise.csv")
print(dataset.head(10))



dataset.tail(10)
dataset.columns
dataset.describe()

dataset["State"].unique()

dataset.shape
dict = {}
for state in dataset["State"].unique():
 dict[state] = 0
for i in range(dataset.shape[0]):
  dict[dataset["State"].iloc[i]] += dataset["First Dose Administered"].iloc[i]
```

```
dataset.dtypes

dataset.isnull().sum()


dataset.fillna(value=0, inplace=True)

dataset.head()




dataset.isnull().sum()

dataset['Total Doses Administered'] = dataset['Total Doses Administered'].astype('int64')

dataset['Sessions'] = dataset['Sessions'].astype('int64')

dataset['First Dose Administered'] = dataset['First Dose Administered'].astype('int64')

dataset['Second Dose Administered'] = dataset['Second Dose Administered'].astype('int64')

dataset['Male (Doses Administered)'] = dataset['Male (Doses Administered)'].astype('int64')

dataset['Female (Doses Administered)'] = dataset['Female (Doses Administered)'].astype('int64')

dataset['Transgender (Doses Administered)'] = dataset['Transgender (Doses
Administered)'].astype('int64')

dataset[' Covaxin (Doses Administered)'] = dataset[' Covaxin (Doses Administered)'].astype('int64')

dataset['CoviShield (Doses Administered)'] = dataset['CoviShield (Doses
Administered)'].astype('int64')

dataset['Sputnik V (Doses Administered)'] = dataset['Sputnik V (Doses Administered)'].astype('int64')

dataset['AEFI'] = dataset['AEFI'].astype('int64')

dataset['18-44 Years (Doses Administered)'] = dataset['18-44 Years (Doses
Administered)'].astype('int64')

dataset['45-60 Years (Doses Administered)'] = dataset['45-60 Years (Doses
Administered)'].astype('int64')

dataset['60+ Years (Doses Administered)'] = dataset['60+ Years (Doses
Administered)'].astype('int64')

dataset['18-44 Years(Individuals Vaccinated)'] = dataset['18-44 Years(Individuals
Vaccinated)'].astype('int64')

dataset['45-60 Years(Individuals Vaccinated)'] = dataset['45-60 Years(Individuals
Vaccinated)'].astype('int64')

dataset['60+ Years(Individuals Vaccinated)'] = dataset['60+ Years(Individuals
Vaccinated)'].astype('int64')

dataset['Male(Individuals Vaccinated)'] = dataset['Male(Individuals Vaccinated)'].astype('int64')
```

```
dataset['Female(Individuals Vaccinated)'] = dataset['Female(Individuals Vaccinated)'].astype('int64')

dataset['Transgender(Individuals Vaccinated)'] = dataset['Transgender(Individuals Vaccinated)'].astype('int64')

dataset['Total Individuals Vaccinated'] = dataset['Total Individuals Vaccinated'].astype('int64')



dataset.dtypes


number_of_males_vaccinated = dataset['Male(Individuals Vaccinated)'].sum()

print('Total Number of Males Vaccinated : ',number_of_males_vaccinated )



number_of_females_vaccinated = dataset['Female(Individuals Vaccinated)'].sum()

print('Total Number of Females Vaccinated : ',number_of_females_vaccinated )


plt.pie([number_of_males_vaccinated, number_of_females_vaccinated], labels = ['Males','Females'],

    colors = ['blue','pink'], autopct = '%0.0f%%', startangle = 90, radius = 2)

plt.legend(loc='best',shadow=True,fontsize=11)

plt.show()
```

Q 15. Use the dataset 'titanic'. The dataset contains 891 rows and contains information about the passengers who boarded the unfortunate Titanic ship. Use the Seaborn library to see if we can find any patterns in the data.

```
#importing pandas library

import pandas as pd


#loading data

titanic = pd.read_csv('titanic.csv')

# View first five rows of the dataset

print(titanic.head())

print(titanic.isnull().sum())
```

```
import seaborn as sns
import matplotlib.pyplot as plt

# Countplot
sns.catplot(x ="Sex", hue ="Survived",
kind ="count", data = titanic)
plt.show()

# Group the dataset by Pclass and Survived and then unstack them
group = titanic.groupby(['Pclass', 'Survived'])
pclass_survived = group.size().unstack()

# Heatmap - Color encoded 2D representation of data.
sns.heatmap(pclass_survived, annot = True, fmt ="d")
plt.show()

# Violinplot Displays distribution of data
# across all levels of a category.
sns.violinplot(x ="Sex", y ="Age", hue ="Survived",
data = titanic, split = True)
plt.show()
```

Q 16. Use the inbuilt dataset 'titanic'. The dataset contains 891 rows and contains information about the passengers who boarded the unfortunate Titanic ship. Write a code to check how the price of the ticket (column name: 'fare') for each passenger is distributed by plotting a histogram.

```
# importing pandas library
import pandas as pd

# loading data
```

```
titanic = pd.read_csv('titanic.csv')
# View first five rows of the dataset
print(titanic.head())
print(titanic.isnull().sum())
import seaborn as sns
import matplotlib.pyplot as plt

# Divide Fare into 4 bins
titanic['Fare_Range'] = pd.qcut(titanic['Fare'], 4)

# Barplot - Shows approximate values based
# on the height of bars.
sns.barplot(x ='Fare_Range', y ='Survived',
data = titanic)
plt.show()

# Countplot
sns.catplot(x ='Embarked', hue ='Survived',
kind ='count', col ='Pclass', data = titanic)
plt.show()
```

Q 17. Compute Accuracy, Error rate, Precision, Recall for following confusion matrix (Use formula for each)

```
tp=1
fn=8
fp=1
tn=90

precision=tp/(tp+fp)
print("Precision:",precision)
```

```
accuracy=(tp+tn)/(tp+tn+fp+fn)
print("Accuracy:",accuracy)

recall=tp/(tp+fn)
print("Recall:",recall)

error=(fp+fn)/(tp+tn+fn+fp)
print("Error:",error)
```

Q 18. Use House_Price prediction dataset. Provide summary statistics (mean, median, minimum, maximum, standard deviation) of variables (categorical vs quantitative) such as- For example, if categorical variable is age groups and quantitative variable is income, then provide summary statistics of income grouped by the age groups.

Q 19. Write a Python program to display some basic statistical details like percentile, mean, standard deviation etc (Use python and pandas commands) the species of ‘Iris-setosa’, ‘Iris-versicolor’ and ‘Iris-versicolor’ of iris.csv dataset.

```
import pandas as pd
data = pd.read_csv("iris.csv")
print(data.describe())
print("Value Count of each species:")
print(data['Species'].value_counts())
```

Q 20. Write a program to cluster a set of points using K-means for IRIS dataset. Consider, K=3, clusters. Consider Euclidean distance as the distance measure. Randomly initialize a cluster mean as one of the data points. Iterate at least for 10 iterations. After iterations are over, print the final cluster means for each of the clusters.

```
import pandas as pd
import numpy as np
from sklearn.cluster import KMeans
import seaborn
import matplotlib.pyplot as plt

df= pd.read_csv("Iris.csv")
print(df)
Y = df['Species']
X = df.drop(columns=['Species'])

clustering = KMeans(n_clusters=3,random_state=3,max_iter=10)

clustering.fit(X)

colorArr = np.array(['g','r','b','m'])

plt.scatter(x=X.SepalLengthCm,y=X.SepalWidthCm,c=colorArr[clustering.labels_])

plt.xlabel("SepalLengthCm")
plt.ylabel("SepalWidthCm")

plt.show()

colorArr = np.array(['g','r','b','m'])

plt.scatter(x=X.PetalLengthCm,y=X.PetalWidthCm,c=colorArr[clustering.labels_])
plt.xlabel("PetalLengthCm")
plt.ylabel("PetalWidthCm")

plt.show()
```

Q 21. Write a program to cluster a set of points using K-means for IRIS dataset. Consider, K=4, clusters. Consider Euclidean distance as the distance measure. Randomly initialize a cluster mean as one of the data points. Iterate at least for 10 iterations. After iterations are over, print the final cluster means for each of the clusters.

```
import pandas as pd
import numpy as np
from sklearn.cluster import KMeans
import seaborn
import matplotlib.pyplot as plt

df= pd.read_csv("Iris.csv")
print(df)
Y = df['Species']
X = df.drop(columns=['Species'])

clustering = KMeans(n_clusters=4 ,random_state=3,max_iter=10)

clustering.fit(X)

colorArr = np.array(['g','r','b','m'])

plt.scatter(x=X.SepalLengthCm,y=X.SepalWidthCm,c=colorArr[clustering.labels_])

plt.xlabel("SepalLengthCm")
plt.ylabel("SepalWidthCm")

plt.show()

colorArr = np.array(['g','r','b','m'])

plt.scatter(x=X.PetalLengthCm,y=X.PetalWidthCm,c=colorArr[clustering.labels_])

plt.xlabel("PetalLengthCm")
plt.ylabel("PetalWidthCm")

plt.show()
```

Q 22.

Compute Accuracy, Error rate, Precision, Recall for the following
confusion matrix.

```
tp=90
fn=210
fp=140
tn=9560

precision=tp/(tp+fp)
print("Precision:",precision)

accuracy=(tp+tn)/(tp+tn+fp+fn)
print("Accuracy:",accuracy)

recall=tp/(tp+fn)
print("Recall:",recall)

error=(fp+fn)/(tp+tn+fn+fp)
```

```
print("Error:",error)
```

Q23. With reference to Table , obtain the Frequency table for the attribute age. From the frequency table you have obtained, calculate the information gain of the frequency table while splitting on Age. (Use step by step Python/Pandas commands)

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sn
import math

df = pd.read_csv("age.csv")



freqTable = pd.crosstab(df['Age'],df['Class'])

lM = [freqTable["Yes"][0],freqTable["No"][0]]
lO = [freqTable["Yes"][1],freqTable["No"][1]]
lY1 = [freqTable["Yes"][2],freqTable["No"][2]]
lY2 = [freqTable["Yes"][3],freqTable["No"][3]]
T = [freqTable["Yes"].sum(),freqTable["No"].sum()]



print(freqTable)

freqTable = pd.crosstab(df['Married'],df['Class'])

print(freqTable)

freqTable= pd.crosstab(df['Health '],df['Class'])
```

```
print(freqTable)


print(lM)

print(lO)

print(lY1)

print(lY2)

print(T)


def entropy(l) -> float:

   total = l[0] + l[1]

   if(l[0] == 0 or l[1] == 0):

      return 0

   else:

      return -((l[0]/total)*math.log2(l[0]/total)+(l[1]/total)*math.log2(l[1]/total))


TE = entropy(T)


EM = entropy(lM)

EY1 = entropy(lY1)

EY2 = entropy(lY2)

EO = entropy(lO)

totalElements = T[0] + T[1]


infoGain = TE - (((lM[0] + lM[1])/totalElements)*EM + ((lY1[0] + lY1[1])/totalElements)*EY1 + ((lO[0]
+ lO[1])/totalElements)*EO + ((lY2[0] + lY2[1])/totalElements)*EY2)



print(f"Info gain for age is : {infoGain}")
```

Q 24. Perform the following operations using Python on a suitable data set, counting unique values of data, format of each column, converting variable data type (e.g. from long to short, vice versa), identifying missing values and filling in the missing values.

```
import math
import numpy as np
import pandas as pd

df = pd.read_excel("cancer patient data sets.xlsx")

df
df['Alcohol use'].unique()
df['Age'].unique()
df['Dust Allergy'].unique()
df.describe()
df.dtypes
df['Chest Pain'].astype("float")

df['Frequent Cold'].astype("float")

df['Air Pollution'].astype("float")

df.dtypes

AlcoholUseMedian = df['Chest Pain'].median()

DustAllergyMedian = df['Dust Allergy'].median()

AirPollutionMedian = df['Air Pollution'].median()

df['Air Pollution'] = df['Air Pollution'].fillna(AirPollutionMedian)
```

```
df['Alcohol use'] = df['Alcohol use'].fillna(AlcoholUseMedian)

df['Dust Allergy'] = df['Dust Allergy'].fillna(DustAllergyMedian)

df.head(10)
```

Q 25. Perform Data Cleaning, Data transformation using Python on any data set.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv("Covid Vaccine Statewise.csv")

df
df.head(10)
df.columns
df.describe

# Data Cleaning

df.fillna(value=0, inplace=True)
df.head()
df.isnull().sum()
df.dtypes

# Data Transformation



df['Total Doses Administered'] = df['Total Doses Administered'].astype('int64')
```

```
df['Sessions'] = df['Sessions'].astype('int64')

df.dtypes

# Droping the columns

val = df['18-44 Years(Individuals Vaccinated)'].mean()

df['18-44 Years(Individuals Vaccinated)'].fillna(val)
```