GUJARAT TECHNOLOGICAL UNIVERSITY
Bachelor of Engineering
Subject Code: 3170717
CLOUD COMPUTING
Semester - 7th Semester

Type of course: Professional Elective

Prerequisite: Fundamentals of Distributed Computing

Rationale: This course aims students to understand the hardware, software concepts and architecture of cloud computing. Students realize the importance of Cloud Virtualization, Abstractions and Enabling Technologies.

Teaching and Examination Scheme:

Teaching Scheme			Credits	Examination Marks				Total Marks
				Theory Marks		Practical Marks		
L	T	P	C	ESE (E)	PA	ESE (V)	PA (I)	
3	0	0	3	70	30	0	0	100

Contents:

Sr. No.	Content	Total Hrs
1	**Introductio**n: Cloud Computing, Layers and Types of Clouds, Cloud Infrastructure Management, Challenges and Applications. Virtualization: Virtualization of Computing, Storage and Resources. Cloud Services: Introduction to Cloud Services IaaS, PaaS and SaaS	04
2	**Software as a Service (SaaS):** Evolution of SaaS, Challenges of SaaS Paradigm, SaaS Integration Services, SaaS Integration of Products and Platforms. Infrastructure As a Services (IaaS): Introduction, Background & Related Work, Virtual Machines Provisioning and Manageability, Virtual Machine Migration Services, VM Provisioning and Migration in Action. Platform As a service (PaaS): Integration of Private and Public Cloud, Technologies and Tools for Cloud Computing, Resource Provisioning services	08
3	**Abstraction and Virtualization:** Introduction to Virtualization Technologies, Load Balancing and Virtualization, Understanding Hyper visors, Understanding Machine Imaging, Porting Applications, Virtual Machines Provisioning and Manageability Virtual Machine Migration Services, Virtual Machine Provisioning and Migration in Action, Provisioning in the Cloud Context, Virtualization of CPU, Memory, I/O Devices, Virtual Clusters and Resource management, Virtualization for Data Center Automation	08
4	**Cloud Infrastructure and Cloud Resource Management:** Architectural Design of Compute and Storage Clouds, Layered Cloud Architecture Development, Design Challenges, Inter Cloud Resource Management, Resource Provisioning and Platform Deployment, Global Exchange of Cloud Resources. Administrating the Clouds, Cloud Management Products, Emerging Cloud Management Standards	08
5	**Security:** Security Overview, Cloud Security Challenges and Risks, Software-as-a Service Security, Cloud computing security architecture: Architectural Considerations, General Issues Securing the Cloud, Securing Data, Data Security, Application Security, Virtual Machine Security, Identity and Presence, Identity Management and Access Control, Autonomic Security Establishing Trusted Cloud	07

	computing, Secure Execution Environments and Communications, , Identity Management and Access control Identity management, Access control, Autonomic Security Storage Area Networks, Disaster Recovery in Clouds.	
6	**Cloud Middleware:** OpenStack, Eucaluptus, Windows Azure, CloudSim, EyeOs, Aneka, Google App Engine	05
7	**Cloud Based Case-Studies:** Overview of Cloud services, Designing Solutions for the Cloud, Implement & Integrate Solutions, Emerging Markets and the Cloud, Tools for Building Private Cloud: IaaS using Eucalyptus, PaaS on IaaS - AppScale	05

Suggested Specification table with Marks (Theory):

Distribution of Theory Marks					
R Level	U Level	A Level	N Level	E Level	C Level
25	**30**	**10**	**05**	**-**	**-**

Legends: R: Remembrance; U: Understanding; A: Application, N: Analyze and E: Evaluate C: Create and above Levels (Revised Bloom's Taxonomy)

Note: This specification table shall be treated as a general guideline for students and teachers. The actual distribution of marks in the question paper may vary from above table.

Books:

1. Rajkumar Buyya, James Broberg, Andrzej M Goscinski, Cloud Computing: Principles and Paradigms, Wiley publication.
2. Toby Velte, Anthony Velte, Cloud Computing: A Practical Approach, McGraw-Hill Osborne Media.
3. George Reese, Cloud Application Architectures: Building Applications and Infrastructure in the Cloud, O'Reilly Publication.
4. John Rhoton, Cloud Computing Explained: Implementation Handbook for Enterprises, Recursive Press.

Course Outcomes: Students will be able to

Sr. No.	**CO Statement**	**Marks % Weightage**
1	Compare the strengths and limitations of cloud computing	15
2	Identify the architecture, infrastructure and delivery models of cloud computing	25
3	Apply suitable virtualization concept.	20
4	Choose the appropriate cloud player, Programming models and approach	20
5	Address the core issues of cloud computing such as security, privacy and interoperability	20

List of Open Source Software/learning website:

- technolamp.blogspot.com
- www.intelligentedu.com/
- NITTR Instructional Resources Videos

uzima® WATER FILTERS

JUST ONE AFRICA®

	Surveyed Date (DD/MM/YY)	Place filter sticker here	29/1/2021	
	Partner			
Q1	Who is condudcting this survey? (First name Last name)		Catherine Walanyi	Q1
Q2	Barcode/Filter ID		UZI-120448	Q2
Q3	Is this a new Filter Installation or Data Update on previous installation?		New | Update |	Q3
Q4	Has your UZima Filter has been working well with little or no issues?			Q4
Q5	Please indicate what is wrong with your filter		No Issues	Q5
Q6	Contact's Name (First and Last name)		Rhoda Munee	Q6
Q7	Contact Phone Number		0718149364	Q7
Q8	Location Name (address,village, city, area, etc)		Chief area	Q8
Q9	Country Name		Kenya	Q9
Q10	How many people live in the household?		3	Q10
Q11	How many children UNDER the age of 5 live in the household?		1	Q11
Q12	How many liters of water does the household use in 1 day for Human Use?		60 Litres	Q12
Q13	What is the source of the water that the household uses?		Tap	Q13
Q14	How far is the water source?		—	Q14
Q15	How many HOURS a day does it require to collect the water?		—	Q15
Q16	How much do you currently spend on water in one day?		60/=	Q16
Q17	How do you Store your water?		Silyplast	Q17
Q18	What type of container do you store your water?		Plastic	Q18
Q19	How many people in the household experienced Amoeba, Typhoid, Cholera, Dysentery, or Diarrhea in the LAST 3 MONTHS?		Nil	Q19
Q20	How many children UNDER AGE OF 5 in the household experienced Typhoid, Cholera, Dysentery, or Diarrhea in the LAST 3 MONTHS?		Nil	Q20
Q21	How much money was spent in the Last 3 months due to Amoeba, Typhoid, Cholera, Dysentery, or Diarrhea for Medicines and Doctor Visits?		Nil	Q21
Q22	How many days of school were missed in the last 3 months due to Amoeba, Typhoid, Cholera, Dysentery, or Diarrhea?		Nil	Q22
Q23	How many work days were missed in the Last 3 Months due to Amoeba, Typhoid, Cholera, Dysentery, or Diarrhea?		Nil	Q23

Q24 Optional field notes: Q24

dm-sem6

September 6, 2023

C.K.Pithawala College of Engineering and Technology, Surat Subject: Data Mining(3160714) Practical file Computer Engineering Department Submitted To: Dr. Ami Tusharkant Choksi

Submitted By: Name: Vanshi Patel - 200090107010 Dataset: Bigmac Price

[ ]:
```
import pandas as pd
import numpy as np
import math
import scipy
from scipy.stats import chi2
```

[ ]:
```
from google.colab import files
uploaded = files.upload()
```

```
<IPython.core.display.HTML object>
```

```
Saving bigmacPrice.csv.csv to bigmacPrice.csv.csv
```

[ ]:
```
from google.colab import files
uploaded = files.upload()
```

```
<IPython.core.display.HTML object>
```

```
Saving swiggy.csv to swiggy.csv
```

[ ]:
```
from google.colab import files
uploaded = files.upload()
```

```
<IPython.core.display.HTML object>
```

```
Saving profile.csv to profile.csv
```

[ ]:
```
from google.colab import files
uploaded = files.upload()
```

```
<IPython.core.display.HTML object>
```

```
Saving BigmacPrice.csv to BigmacPrice.csv
```

[ ]:
```
from google.colab import files
uploaded = files.upload()
```

```
<IPython.core.display.HTML object>

Saving Ubereat_US_Merchant.csv to Ubereat_US_Merchant.csv
```

[ ]:
```
from google.colab import files
uploaded = files.upload()
```

```
<IPython.core.display.HTML object>

Saving zomato_Hyderabad.csv to zomato_Hyderabad (1).csv
```

#CO2160714.1 Assignment:

1.Analyze 5 data sets from the UCI repository. Print the following details about each data set number of records/instances (b) number of incomplete records (c) number of attributes

[ ]:
```
dataList = ['swiggy.csv','profile.csv','BigmacPrice.csv','Ubereat_US_Merchant.
↪csv','zomato_Hyderabad.csv']

for i in range(5):
  current_data=pd.read_csv(dataList[i])
  df_current_data=pd.DataFrame(current_data)
  print(df_current_data.head())
  print("Number of records/instances = " + str(df_current_data.count()))
  print("Number of incomplete records = " + str(df_current_data.isna().sum().
↪sum()))
  print("Number of attributes = " + str(df_current_data.shape[1]))
```

```
       id                 name    city rating     rating_count   cost  \
0  567335      AB FOODS POINT  Abohar     --  Too Few Ratings    200
1  531342  Janta Sweet House  Abohar    4.4      50+ ratings    200
2  158203  theka coffee desi  Abohar    3.8     100+ ratings    100
3  187912           Singh Hut  Abohar    3.7      20+ ratings    250
4  543530       GRILL MASTERS  Abohar     --  Too Few Ratings    250

                     cuisine          lic_no  \
0            Beverages,Pizzas  22122652000138
1               Sweets,Bakery  12117201000112
2                   Beverages  22121652000190
3            Fast Food,Indian  22119652000167
4  Italian-American,Fast Food  12122201000053

                                                link  \
0  https://www.swiggy.com/restaurants/ab-foods-po…
1  https://www.swiggy.com/restaurants/janta-sweet…
2  https://www.swiggy.com/restaurants/theka-coffe…
3  https://www.swiggy.com/restaurants/singh-hut-n…
4  https://www.swiggy.com/restaurants/grill-maste…
```

```
  sum=0
  for i in range(0,n):
      sum = sum +  float(bin2[i])

  return float(sum/n)
print("Mean:",Mean(bin2,n))


#OUTPUT
```

```
Mean: 4.1499999999999995
```

[ ]:
```
#calculating mean value for bin3
def Mean(bin3,n):

  sum=0
  for i in range(0,n):
      sum = sum +  float(bin3[i])

  return float(sum/n)
print("Mean:",Mean(bin3,n))

#OUTPUT
```

```
Mean: 4.324999999999999
```

[ ]:
```
#replacing bin values with the mean
a1=np.full((1,4),(Mean(bin1,n)))
a2=np.full((1,4),(Mean(bin2,n)))
a3=np.full((1,4),(Mean(bin3,n)))
print("bin1:",a1)
print("bin2:",a2)
print("bin3:",a3)

#OUTPUT
```

```
bin1: [[3.875 3.875 3.875 3.875]]
bin2: [[4.15 4.15 4.15 4.15]]
bin3: [[4.325 4.325 4.325 4.325]]
```

Smoothing by bin medians

[ ]:
```
#calculating median for bin1
def Median(bin1,n):
  #check for even case
  if n%2!=0:
    return bin1[int(n/2)]
```

```
  else:
    return ((float(bin1[int((n-1)/2)])+float(bin1[int(n/2)]))/2.0)
print("Median:",Median(bin1,n))

#OUTPUT
```

```
Median: 3.9
```

[ ]:
```
#calculating median for bin2
def Median(bin2,n):
  #check for even case
  if n%2!=0:
    return bin2[int(n/2)]
  else:
    return ((float(bin2[int((n-1)/2)])+float(bin2[int(n/2)]))/2.0)
print("Median:",Median(bin2,n))

#OUTPUT
```

```
Median: 4.15
```

[ ]:
```
#calculating median for bin3
def Median(bin3,n):
  #check for even case
  if n%2!=0:
    return bin3[int(n/2)]
  else:
    return ((float(bin3[int((n-1)/2)])+float(bin3[int(n/2)]))/2.0)
print("Median:",Median(bin3,n))

#OUTPUT
```

```
Median: 4.3
```

[ ]:
```
#replacing bin values with the median
b1=np.full((1,4),(Median(bin1,n)))
b2=np.full((1,4),(Median(bin2,n)))
b3=np.full((1,4),(Median(bin3,n)))
print("bin1:",b1)
print("bin2:",b2)
print("bin3:",b3)

#OUTPUT
```

```
bin1: [[3.9 3.9 3.9 3.9]]
bin2: [[4.15 4.15 4.15 4.15]]
bin3: [[4.3 4.3 4.3 4.3]]
```

Smoothing by bin boundaries

[ ]:
```
#checking for the boundry conditions for bin1
a=float(bin1[1])-float(bin1[0])

b=float(bin1[3])-float(bin1[1])

if(a<b):
  bin1[1]=bin1[0]
else:
  bin1[1]=bin1[3]

c=float(bin1[2])-float(bin1[0])

d=float(bin1[3])-float(bin1[2])

if(c<d):
  bin1[2]=bin1[0]
else:
  bin1[2]=bin1[3]
#replacing bin values with the valid boundary conditin value

print("smooth bin1:",bin1)


#OUTPUT
```

```
smooth bin1: ['3.7' '4' '4' '4']
```

[ ]:
```
#checking for the boundry conditions for bin2
a=float(bin2[1])-float(bin2[0])

b=float(bin2[3])-float(bin2[1])

if(a<b):
  bin2[1]=bin2[0]
else:
  bin2[1]=bin2[3]

c=float(bin2[2])-float(bin2[0])

d=float(bin2[3])-float(bin2[2])

if(c<d):
  bin2[2]=bin2[0]
else:
  bin2[2]=bin2[3]
#replacing bin values with the valid boundary conditin value
```

```
print("smooth bin2:",bin2)

#OUTPUT
```

```
smooth bin2: ['4.1' '4.1' '4.2' '4.2']
```

[ ]:
```
#checking for the boundry conditions for bin3
a=float(bin3[1])-float(bin3[0])

b=float(bin3[3])-float(bin3[1])

if(a<b):
  bin3[1]=bin3[0]
else:
  bin3[1]=bin3[3]

c=float(bin3[2])-float(bin3[0])

d=float(bin3[3])-float(bin3[2])

if(c<d):
  bin3[2]=bin3[0]
else:
  bin3[2]=bin3[3]
#replacing bin values with the valid boundary conditin value

print("smooth bin3:",bin3)


#OUTPUT
```

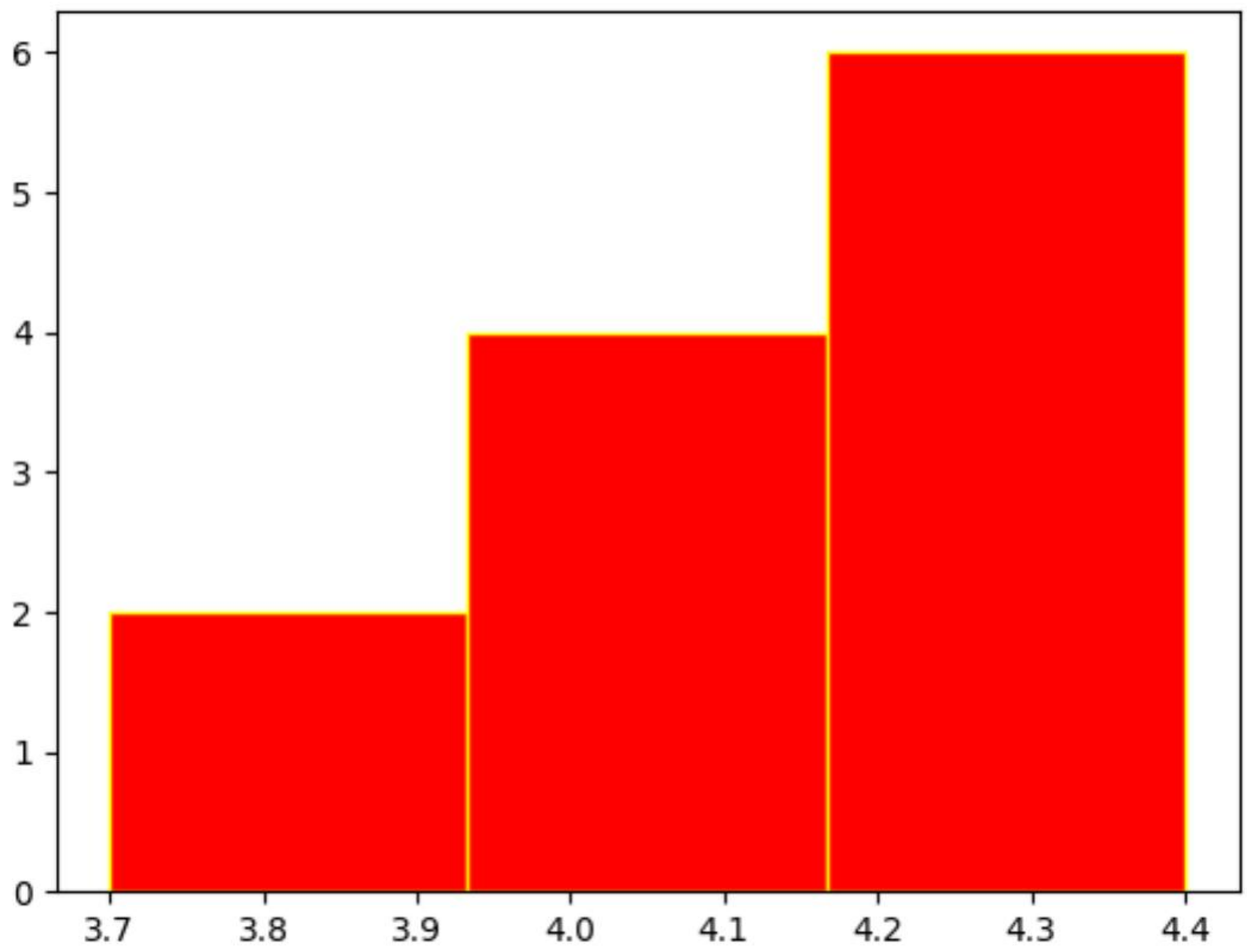
```
smooth bin3: ['4.3' '4.3' '4.3' '4.4']
```

[ ]:
```
print("Bin 1:",bin1,"\nBin 2:",bin2,"\nBin 3:",bin3)

#OUTPUT
```

```
Bin 1: ['3.7' '4' '4' '4']
Bin 2: ['4.1' '4.1' '4.2' '4.2']
Bin 3: ['4.3' '4.3' '4.3' '4.4']
```

[ ]:
```
import matplotlib.pyplot as plt

height=[3.7, 4, 3.7, 4,
        4.1, 4.1, 4.2, 4.2,
        4.3, 4.3, 4.3, 4.4,
```

```
        ]
plt.hist(height,bins=3,edgecolor="yellow",color="red")
plt.show()

#OUTPUT
```



[B] Find covariance(cov) and correlation(r), Sx and Sy are standard deviation, x and $\bar{y}$ are means. Cov(x,y) = i=1n(Xi-X)(Yi-Y)/(n-1) r(x,y) =Cov(x,y)/sxsy Plot the correlation, to show whether two variables are positively correlated, negatively correlated or no relation between them.

Covariance

[ ]:
```
#covariance

b=[72, 69, 90, 47, 76, 71, 88, 40, 64, 38, 58, 40]
c=[74, 88, 93, 44, 75, 78, 92, 39, 67, 50, 52, 43]

def covariance(x, y):
# Finding the mean of the series x and y
    mean_x = sum(b)/float(len(b))
    mean_y = sum(c)/float(len(c))
```

```
# Subtracting mean from the individual elements
    sub_x = [i - mean_x for i in x]
    sub_y = [i - mean_y for i in y]
    numerator = sum([sub_x[i]*sub_y[i] for i in range(len(sub_x))])
    denominator = len(x)-1
    cov = numerator/denominator
    return cov

#print(mean_x)
cov_func = covariance(b, c)
print("Covariance :", cov_func)

#OUTPUT
```

```
Covariance : 342.70454545454544
```

Correlation

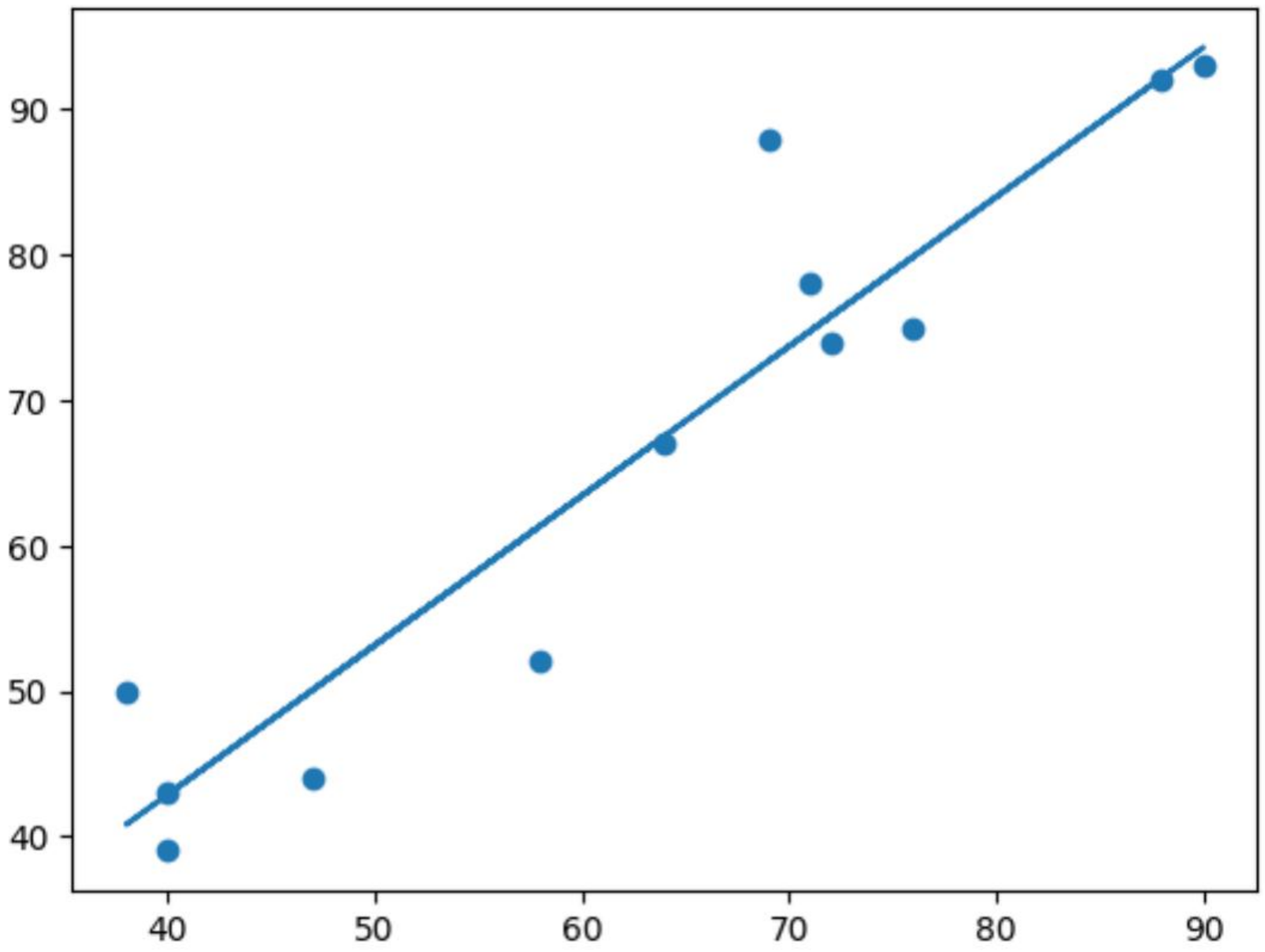
[ ]:
```
#correlation between x and y

x=[72, 69, 90, 47, 76, 71, 88, 40, 64, 38, 58, 40]
y=[74, 88, 93, 44, 75, 78, 92, 39, 67, 50, 52, 43]
import math
mean_x = sum(x)/float(len(x))
#print(mean_x)
sub_x = [i - mean_x for i in x]
#print(sub_x)
#d=sum(sub_x)
d=121
e=(d*d)/len(x)
Sx = round(math.sqrt(e),2)
#print("Standard Deviation of x is :")
#print(Sx)

mean_y = sum(y)/float(len(y))
#print(mean_y)
sub_y = [i - mean_y for i in y]
#print(sub_y)
#f=sum(sub_y)
f=3
g = (f*f)/len(y)
Sy=round(math.sqrt(g),2)
#print("Standard Deviation of y is :")
#print(Sy)
z = Sx*Sy
#print(z)

corr=round(cov_func/z,2)
```

```
print("Correlation between x and y is :",corr)
if corr>0:
    print("x and y are positively correlated")
elif corr<0:
    print("x and y are negatively correlated")
else:
    print("x and y are not correlated")

#OUTPUT
```

```
Correlation between x and y is : 11.28
x and y are positively correlated
```

[ ]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
#define data
x = np.array([72, 69, 90, 47, 76, 71, 88, 40, 64, 38, 58, 40])
y = np.array([74, 88, 93, 44, 75, 78, 92, 39, 67, 50, 52, 43])
#find line of best fit
a, b = np.polyfit(x, y, 1)

#add points to plot
plt.scatter(x, y)

#add line of best fit to plot
plt.plot(x, a*x+b)

#OUTPUT
```

[ ]:
```
[<matplotlib.lines.Line2D at 0x7f7dbe865990>]
```



0.1 3 Assignment

Implement a chi-square test to detect whether two variables are dependent or independent variables for your dataset.

[ ]:
```
#creating table

ar=np.array([[2.85,9.9],[2.85,1.24]])
df=pd.DataFrame(ar,columns=["Local_Price","Dollar_Price"])
df.index=['CANADA','CHINA']
df

#OUTPUT
```

[ ]:
```
         Local_Price  Dollar_Price
CANADA          2.85          9.90
CHINA           2.85          1.24
```

[ ]:
```
df2=df.copy()#create contingency table with the marginal totals
df2.loc['Column_Total']=df2.sum(numeric_only=True, axis=0)
```

```
df2.loc[:,'Row_Total']=df2.sum(numeric_only=True, axis=1)
df2

#OUTPUT
```

[ ]:

	Local_Price	Dollar_Price	Row_Total
CANADA	2.85	9.90	12.75
CHINA	2.85	1.24	4.09
Column_Total	5.70	11.14	16.84

[ ]:
```
from sqlalchemy import String
n=df2.at["Column_Total","Row_Total"] # grand total
exp=df2.copy()                       #create dataframe with expected counts
for x in exp.index[0:-1]:
  for y in exp.columns[0:-1]:
    #round expected values to 6 decimal places to get the maximum available
    v=str (((df2.at[x,"Row_Total"])*(df2.at["Column_Total",y]) )/n)
    exp.at[x,y]=float(v)

exp=exp.iloc[[0,1],[0,1]]
exp

#OUTPUT
```

[ ]:

	Local_Price	Dollar_Price
CANADA	4.315618	8.434382
CHINA	1.384382	2.705618

[ ]:
```
#calculate chi-squared test statistics
tstat=np.sum(((df-exp)**2/exp).values)
print("chi square test statistic:",tstat)

#OUTPUT
```

```
chi square test statistic: 3.09794748789061O6
```

[ ]:
```
dof=(len(df.columns)-1)*(len(df.index)-1)#determine degrees of freedom
dof

#OUTPUT
```

[ ]:
```
1
```

[ ]:
```
pval=1-chi2.cdf(tstat,dof)
pval

#OUTPUT
```

[ ]: 0.07839107056704153

[ ]:
```
#print value of p
alpha=0.05
print("p value is"+str(round(pval,5)))
if pval<=alpha:
  print("Dependent (reject HO)")
else:
  print("Independent (HO holds true)")


#OUTPUT
```

```
p value is0.07839
Independent (HO holds true)
```

0.2 4 Assignment

Write a program to implement normalization techniques (a)min max (b) z-score (c) decimal scaling on your data set.

[ ]:
```
import statistics
```

[ ]:
```
a=[0]*10
for i in range(10):
  b=pd.read_csv('BigmacPrice.csv')
  a[i]=b.dollar_ex[i]
```

[ ]:
```
data=a
print(data)

#OUTPUT
```

```
[1, 1, 1, 1, 1, 514, 8, 39, 8, 1]
```

[ ]:
```
data=np.sort(data)
print(data)

#OUTPUT
```

```
[  1   1   1   1   1   1   8   8  39 514]
```

[ ]:
```
# A MINMAX METHOD

def minMax(num,list):
  minNum=int(input("Enter Minimum setting:\t"))
  maxNum=int(input("Enter Maximum setting:\t"))
  ans=round(((num-min(list))/(max(list)-min(list))*(maxNum-minNum))+minNum,2)
```

```
                                                address                menu
0  AB FOODS POINT, NEAR RISHI NARANG DENTAL CLINI…  Menu/567335.json
1  Janta Sweet House, Bazar No.9, Circullar Road,…  Menu/531342.json
2         theka coffee desi, sahtiya sadan road city  Menu/158203.json
3    Singh Hut, CIRCULAR ROAD NEAR NEHRU PARK ABOHAR  Menu/187912.json
4  GRILL MASTERS, ADA Heights, Abohar - Hanumanga…  Menu/543530.json
Number of records/instances = id              148541
name            148455
city            148541
rating          148455
rating_count    148455
cost            148410
cuisine         148442
lic_no          148312
link            148541
address         148455
menu            148541
dtype: int64
Number of incomplete records = 803
Number of attributes = 11
   Unnamed: 0 gender  age                                id  became_member_on  \
0           0    NaN  118  68be06ca386d4c31939f3a4f0e3dd783          20170212
1           1      F   55  0610b486422d4921ae7d2bf64640c50b          20170715
2           2    NaN  118  38fe809add3b4fcf9315a9694bb96ff5          20180712
3           3      F   75  78afa995795e4d85b5d9ceeca43f5fef          20170509
4           4    NaN  118  a03223e636434f42ac4c3df47e8bac43          20170804

     income
0       NaN
1  112000.0
2       NaN
3  100000.0
4       NaN
Number of records/instances = Unnamed: 0          17000
gender                14825
age                   17000
id                    17000
became_member_on      17000
income                14825
dtype: int64
Number of incomplete records = 4350
Number of attributes = 6
         date currency_code       name  local_price  dollar_ex  dollar_price
0  2000-04-01           ARS  Argentina         2.50          1          2.50
1  2000-04-01           AUD  Australia         2.59          1          2.59
2  2000-04-01           BRL     Brazil         2.95          1          2.95
3  2000-04-01           GBP    Britain         1.90          1          1.90
4  2000-04-01           CAD     Canada         2.85          1          2.85
```

```
  return ans

#OUTPUT
```

[ ]:
```
# B ZSCORE METHOD

def zscore(num,mean,stdDv):
  return round((num-mean)/stdDv,2)
```

[ ]:
```
#   C DECIMAL SCALING METHOD

def descaling(num,maxNum):
  digit=len(str(maxNum))
  div=pow(10,digit)
  return num/div
```

[ ]:
```
num=int(input("enetr an item from data: \t"))
print("After doing min-max normalization:",minMax(num,data))
print("After doing z-score normalization:",zscore(num,statistics.
  ↪mean(data),statistics.stdev(data)))
print("After doing descaling normalization:",descaling(num,max(data)))

#OUTPUT
```

```
enetr an item from data:        39
Enter Minimum setting:  0
Enter Maximum setting:  3
After doing min-max normalization: 0.22
After doing z-score normalization: -0.11
After doing descaling normalization: 0.039
```

0.3 5 Assignment

Write a program to implement data reduction techniques for your data.

[ ]:
```
df=pd.read_csv('BigmacPrice.csv')
df.head(10)

#OUTPUT
```

[ ]:

	date	currency_code	name	local_price	dollar_ex	\
0	2000-04-01	ARS	Argentina	2.50	1	
1	2000-04-01	AUD	Australia	2.59	1	
2	2000-04-01	BRL	Brazil	2.95	1	
3	2000-04-01	GBP	Britain	1.90	1	
4	2000-04-01	CAD	Canada	2.85	1	
5	2000-04-01	CLP	Chile	1260.00	514	

```
6  2000-04-01          CNY            China          9.90          8
7  2000-04-01          CZK  Czech Republic         54.37         39
8  2000-04-01          DKK          Denmark         24.75          8
9  2000-04-01          EUR        Euro area          2.56          1

   dollar_price
0          2.50
1          2.59
2          2.95
3          1.90
4          2.85
5          2.45
6          1.24
7          1.39
8          3.09
9          2.56
```

[ ]:
```
k=int(input())
data=df['dollar_ex'].head(k)
data

#OUTPUT
```

```
10
```

[ ]:
```
0      1
1      1
2      1
3      1
4      1
5    514
6      8
7     39
8      8
9      1
Name: dollar_ex, dtype: int64
```

[ ]:
```
d=[i for i in data]
d.sort()
d

#OUTPUT
```

[ ]:
```
[1, 1, 1, 1, 1, 1, 8, 8, 39, 514]
```

[ ]:
```
a=min(d)
b=max(d)
```

```
n=3
w=round((b-1)/n,2)
w


#OUTPUT
```

[ ]:
```
171.0
```

[ ]:
```
#using equi-width binning
dict={'b1':[],'b2':[],'b3':[]}
i=0
j=1
k=0
while i<3:
  if i==2:
    for z in range(k,len(d)):
      dict['b3'].append(d[z])
    break
  if d[k]<=(a+j*w):
    dict[f'b{i+1}'].append(d[k])
    k +=1
  else:
    i +=1
    j +=1
dict

#OUTPUT
```

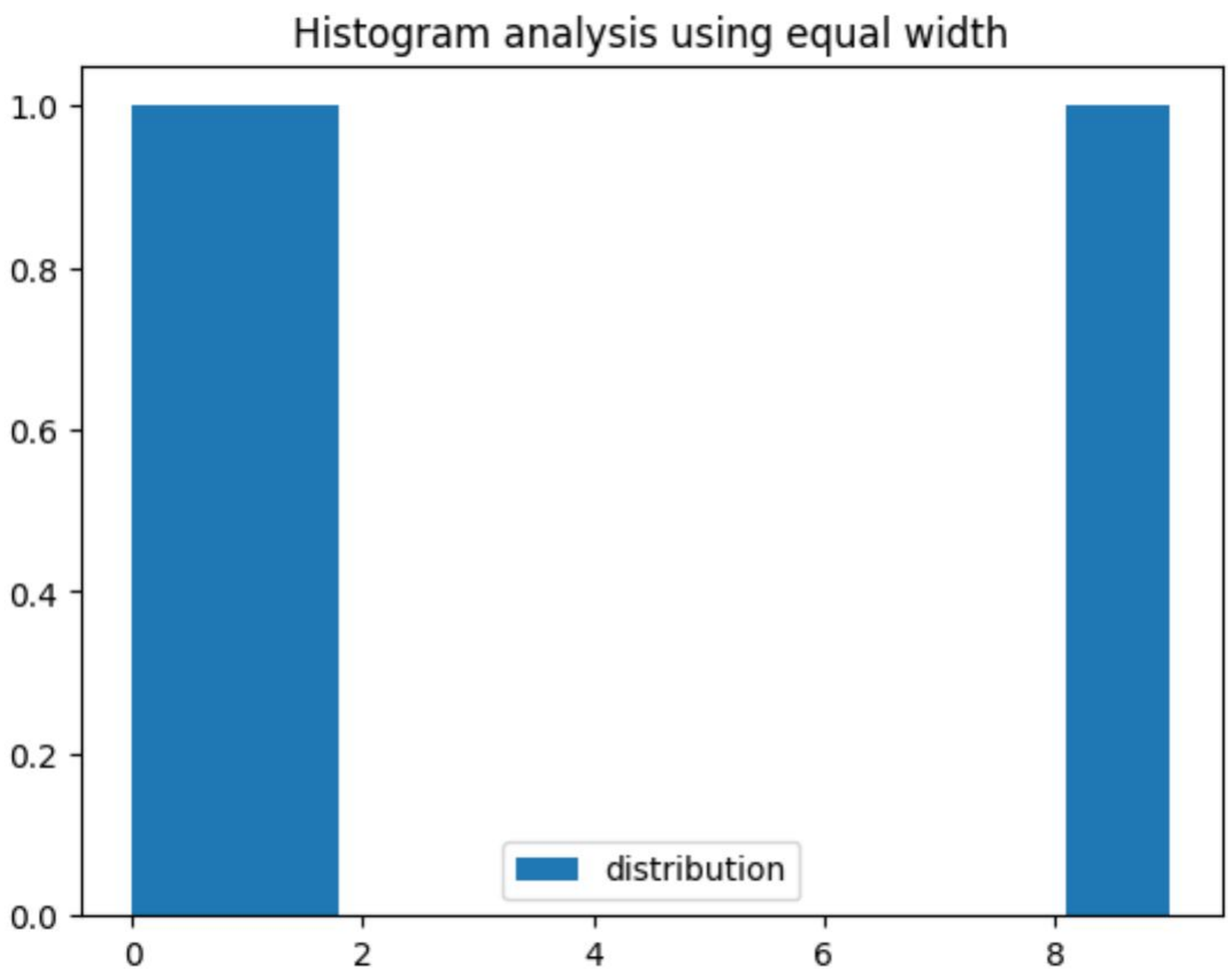
[ ]:
```
{'b1': [1, 1, 1, 1, 1, 1, 8, 8, 39], 'b2': [], 'b3': [514]}
```

[ ]:
```
#using equi-frequency binning
d=[i for i in data]
d.sort()
div=len(d)%3
i,j,k=0,0,0
array=[[],[],[]]
while i<3:
  array[i].append(d[j])
  j +=1
  k +=1
  if k==len(d)//3:
    if (div>0):
      array[i].append(d[j+1])
      j +=1
      div -=1
    k=0
```

```
    i +=1
array

#OUTPUT
```

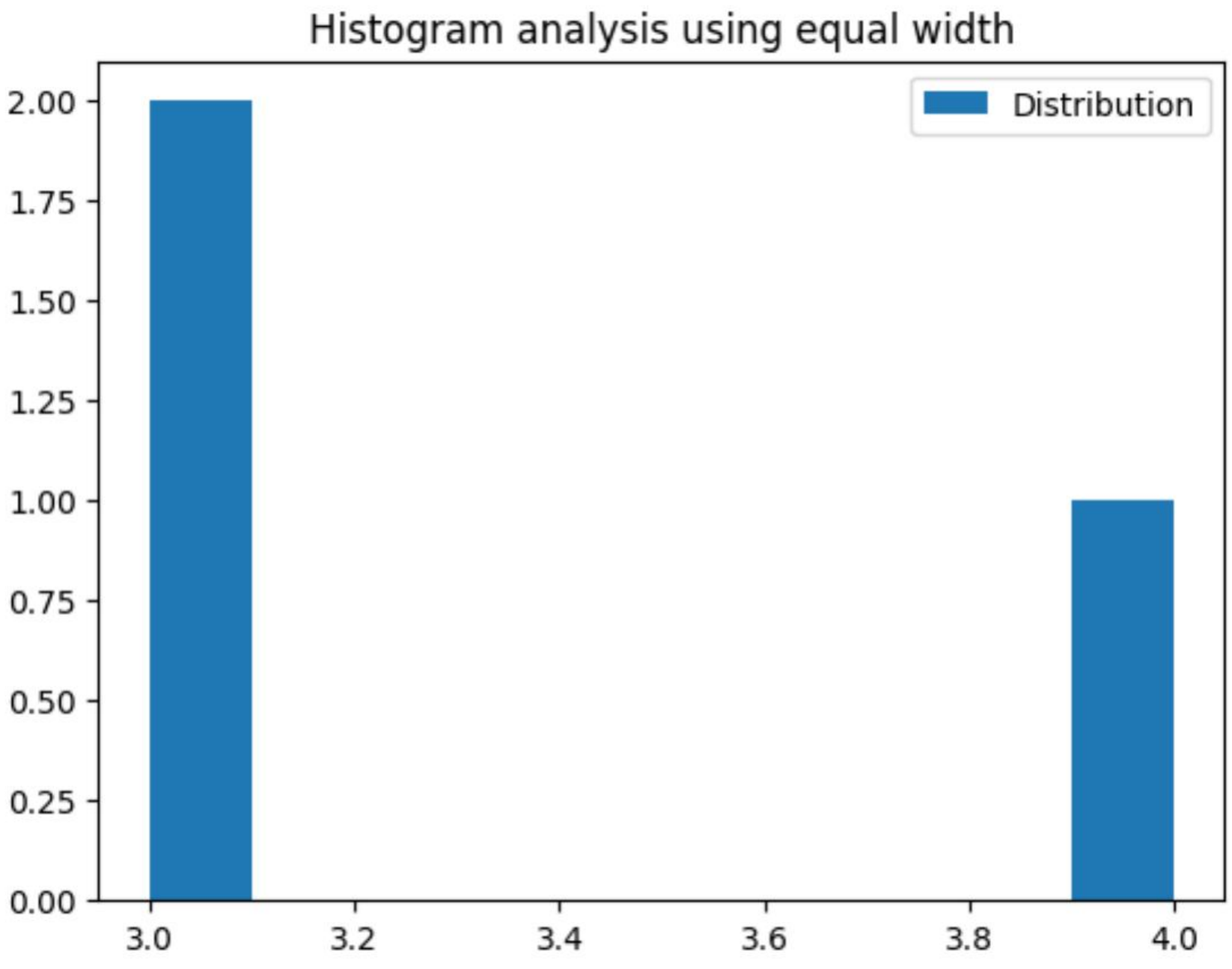
[ ]: [[1, 1, 1, 1], [1, 1, 8], [8, 39, 514]]

[ ]:
```
import matplotlib.pyplot as plt
p=[len(dict[f'b{i+1}']) for i in range(3)]
plt.title('Histogram analysis using equal width')
legend=['distribution']
bin=[f'[-,{a+w+1})',f'[{a+w+1},{a+2*w+2})',f'[{a+2*w+2},+)']
plt.hist(p)
plt.legend(legend)
plt.show()

#OUTPUT
```



[ ]:
```
p=[len(array[i]) for i in range(3)]
plt.title('Histogram analysis using equal width')
legend=['Distribution']
```

```
bin=[f'[-,{a+w+1})',f'[{a+w+1},{a+2*w+2})',f'[{a+2*w+2},+)']
plt.hist(p)
plt.legend(legend)
plt.show()

#OUTPUT
```



0.4 6 Assignment

Write a program to implement any method of data discretization.

[ ]:
```
a=[0]*12
for i in range(12):
  b=pd.read_csv('BigmacPrice.csv')
  a[i]=b.dollar_price[i]
```

[ ]:
```
#printing the data
data=a
print(data)

#OUTPUT
```

```
[2.5, 2.59, 2.95, 1.9, 2.85, 2.45, 1.24, 1.39, 3.09, 2.56, 1.46, 1.22]
```

[ ]:
```
#sorting the data
data=np.sort(data)
print(data)

#OUTPUT
```

```
[1.22 1.24 1.39 1.46 1.9  2.45 2.5  2.56 2.59 2.85 2.95 3.09]
```

[ ]:
```
#splitting the data into equal parts
y=np.split(data,3)
print(y)

#OUTPUT
```

```
[array([1.22, 1.24, 1.39, 1.46]), array([1.9 , 2.45, 2.5 , 2.56]), array([2.59,
2.85, 2.95, 3.09])]
```

[ ]:
```
#taking empty array for sorting data in bins
bin1=np.zeros((1,4))
bin2=np.zeros((1,4))
bin3=np.zeros((1,4))
print(bin1)
print(bin2)
print(bin3)

#OUTPUT
```

```
[[0. 0. 0. 0.]]
[[0. 0. 0. 0.]]
[[0. 0. 0. 0.]]
```

[ ]:
```
#sorting the data in the bins
bin1=y[0]
bin2=y[1]
bin3=y[2]
print('bin1:',bin1)
print('bin2:',bin2)
print('bin3:',bin3)

#OUTPUT
```

```
bin1: [1.22 1.24 1.39 1.46]
bin2: [1.9  2.45 2.5  2.56]
bin3: [2.59 2.85 2.95 3.09]
```

Binning by Equal Width

[ ]:
```
#take number of bins equals to 3
#finding width
a=[]
b=[]
c=[]
width=(data[11]-data[0])/3;
d=math.ceil(width)#take upper bound
print("width=",d)
e=d+data[0]
f=e+d
g=f+d

#OUTPUT
```

```
width= 1
```

[ ]:
```
z=0
for i in range(12):
  if(data[i]>=e):
    a.append(data[i])
    z=z+1
    break
  else:
    a.append(data[i])
    z=z+1
print(a)

#OUTPUT
```

```
[1.22, 1.24, 1.39, 1.46, 1.9, 2.45]
```

[ ]:
```
k=z
y=0
for i in range(z,12):
  if(data[i]>=f):
    break
  else:
    b.append(data[i])
    k=k+1
print(b)

#OUTPUT
```

```
[2.5, 2.56, 2.59, 2.85, 2.95, 3.09]
```

[ ]:
```
for i in range(k,12):
  if(data[i]>=g):
    break
```

```
  else:
    c.append(data[i])
print(c)

#OUTPUT
```

```
[]
```

[ ]:
```
#printing the bin values
print("Bin1=",a)
print("Bin2=",b)
print("Bin3=",c)

#OUTPUT
```

```
Bin1= [1.22, 1.24, 1.39, 1.46, 1.9, 2.45]
Bin2= [2.5, 2.56, 2.59, 2.85, 2.95, 3.09]
Bin3= []
```

#CO2160714.2 Assignment:

0.5 7 Assignment

Implement apriori algorithm and show the output as candidate sets in each iteration, as well as show association rules generated, without using standard apriori method from the python libraries.

[ ]:
```
# minimum support, minimum confidence
a=3
b=75
print("Min_Support_count=",a)
print("Min_Confidence=",b,"%")
import pandas as pd
import pandas as np
#creating the Dataframe
r=  pd.DataFrame({
'TID':[1,2,3,4],
'Item':[["11", "12", "13", "14"], ["12", "13"],["13", "14"], ["12", "13", "14"]]
})
r

#OUTPUT
```

```
Min_Support_count= 3
Min_Confidence= 75 %
```

[ ]:

	TID	Item
0	1	[11, 12, 13, 14]
1	2	[12, 13]
2	3	[13, 14]

```
3    4      [12, 13, 14]
```

Genarating Candidate Set C1

[ ]:
```
Items=["11", "12", "13", "14"]
sum=0
t=[0,0,0,0]
for i in range(4):
  for j in range(4):
    sum=sum+r.Item[j].count(r.Item[0][i])
  #print(sum)
  t[i]=sum
  sum=0
#print(t)
Items=["11","12", "13", "14"]
support=(t)
c1=pd.DataFrame({
  "Itemset":Items,
  "support_count":support
})
c1

#OUTPUT
```

[ ]:

	Itemset	support_count
0	11	1
1	12	3
2	13	4
3	14	3

Frequent Itemset L1

[ ]:
```
#removing the itemsets whose support count is less than minimum support count
 ↪L1 = c1[c1['support_count'] >= 3]
#frequent itemset L1
L1 = c1[c1['support_count']>=3]
L1
L1.index=['0', '1', '2']
print(" Frequent Itemset L1")
L1

#OUTPUT
```

```
Frequent Itemset L1
```

[ ]:

	Itemset	support_count
0	12	3
1	13	4

2 14 3

Generating Candidate Set C2

[ ]:
```
#candidate set c2 obtained by pairing itemsets of L1 with itself
c2= [(a, b) for i, a in enumerate (L1.Itemset) for b in L1.Itemset[i + 1:]]
#print("possible pairs for candidate set c2 are : ", c2)
```

[ ]:
```
#finding the support count of paired itemsets
sum=0
f=[0,0,0]
for i in range(3):
  for j in range(4):
    if (c2[i][0] in r.Item[j]) and (c2[i][1] in r.Item[j]):
      sum=sum+1
  f[i]=sum
  sum=0
#print (f)

# candidate set C2 will be
Item1= [('12', '13'), ('12', '14'), ('13', '14')]
support1=(f)
C2=pd.DataFrame({
  "Itemset": Item1,
  "support_count": support1
})
C2

#OUTPUT
```

[ ]:

	Itemset	support_count
0	(12, 13)	3
1	(12, 14)	2
2	(13, 14)	3

Frequent Itemset L2

[ ]:
```
L2= C2[C2['support_count'] >= 3]
L2
L2.index=['0', '1']
print(" Frequent Itemset L2")
L2

#OUTPUT
```

Frequent Itemset L2

```
Number of records/instances = date              1946
currency_code    1946
name             1946
local_price      1946
dollar_ex        1946
dollar_price     1946
dtype: int64
Number of incomplete records = 0
Number of attributes = 6
   index            city state zipcode  \
0      0  Alexander City    AL   35010
1      1     Albertville    AL   35951
2      2  Alexander City    AL   35010
3      3     Albertville    AL   35950
4      4  Alexander City    AL   35010

                                        address  \
0  4097 U S Highway 280, Alexander City, AL 35010
1       7300 Hwy 431 North, Albertville, AL 35951
2   4097 Us Highway 280, Alexander City, AL 35010
3          7959 Us Hwy 431, Albertville, AL 35950
4      977 Jefferson St, Alexander City, AL 35010

                                loc_name  \
0  The Saucy Hen (4097 U. S. HIGHWAY 280)
1        Burger King (7300 Hwy 431 North)
2    MrBeast Burger (4097 US Highway 280)
3         Taco Bell (7959 Us Highway 431)
4                             The Station

                             loc_number  \
0  0623b7ac-598d-5016-bdd2-febb44d79b12
1  62a60773-5644-4d73-b969-a4922ce70fa6
2  308b1654-60f1-51d4-bfe2-ed7c849442ac
3  ef86513f-3973-4315-b938-bb6f230c5c58
4  9507eb1b-5afc-4ee1-a566-526d9e2ba2d0

                                                 url            promotion  \
0  https://www.ubereats.com/store/the-saucy-hen-4…                  NaN
1  https://www.ubereats.com/store/burger-king-730…                  NaN
2  https://www.ubereats.com/store/mrbeast-burger-…                  NaN
3  https://www.ubereats.com/store/taco-bell-7959-…  Spend $15, Save $5
4  https://www.ubereats.com/store/the-station/950…                  NaN

    latitude  …  review_count  review_rating price_bucket  \
0  32.923880  …           NaN            NaN           $$
1  34.277260  …           NaN            NaN            $
2  32.923880  …           NaN            NaN           $$
```

[ ]:

	Itemset	support_count
0	(12, 13)	3
1	(13, 14)	3

Generating Candidate Set C3

[ ]:
```
itemset_list_2 = L2['Itemset']
b = []
c3_itemset_list = []
#Loop for making of 3-itemset candidate generation
for i in itemset_list_2:
  for j in i:
    if j not in b:
      b.append(j)

#print(b)

sum=0
for i in range(1):
  for j in range(4):
    if (b[0] in r.Item[j]) and (b[1] in r.Item[j]) and (b[2] in r.Item[j]):
      sum=sum+1
#print (sum)

Item1=[b]
support1=(sum)
C3=pd.DataFrame({
  "Itemset": Item1,
  "support_count": support1
})
C3

#OUTPUT
```

[ ]:

	Itemset	support_count
0	[12, 13, 14]	2

[ ]:
```
genearting_rules= L2['Itemset'].tolist()
#print(gr)
rules = []
for item in genearting_rules:
  reverse_item = item[::-1]
  if item not in rules:
    rules.append(item)
  if reverse_item not in rules:
    rules.append(reverse_item)
```

```
rules

#calculating the confidence
print('The rules are as follows: ')
for i in range(4):
  print(f'R{i}:', rules[i][0], '=>', rules[i][1])
  if(i==0 or i==1):
    for j in range(1):
      b=100*L2.support_count[j]/L1.support_count[i]
      print('Confidence = ',b,'%')
      if b >= 75:
        print (f'R(i+1) is accepted\n')
      else:
        print(f'R{i+1} is rejected\n')
  else:
    for j in range(1):
      b=100*L2.support_count[j+1]/L1.support_count[i-1]
      print('Confidence = ',b,'%')
      if b >= 75:
        print (f'R{i+1} is accepted\n')
      else:
        print(f'R{i+1} is rejected\n')

#OUTPUT
```

```
The rules are as follows:
R0: 12 => 13
Confidence =  100.0 %
R(i+1) is accepted

R1: 13 => 12
Confidence =  75.0 %
R(i+1) is accepted

R2: 13 => 14
Confidence =  75.0 %
R3 is accepted

R3: 14 => 13
Confidence =  100.0 %
R4 is accepted
```

0.6 8 Assignment

Implement any algorithm that removes the limitations of an apriori algorithm. (Transaction reduction, DIC, DHT, FP-tree) without using any python machine learning library.

[ ]:
```
def transaction_reduction(data, min_support):
    # Get the support count for each item
    support_count = {}
    for transaction in data:
        for item in transaction:
            if item not in support_count:
                support_count[item] = 1
            else:
                support_count[item] += 1

    # Remove transactions that do not contain frequent items
    reduced_data = []
    for transaction in data:
        reduced_transaction = []
        for item in transaction:
            if support_count[item] >= min_support:
                reduced_transaction.append(item)
        if len(reduced_transaction) > 0:
            reduced_data.append(reduced_transaction)

    return reduced_data

#OUTPUT
```

[ ]:
```
data = [["11", "12", "13", "14"], ["12", "13"],["13", "14"], ["12", "13", "14"]]
min_support = 3
reduced_data = transaction_reduction(data, min_support)
print(reduced_data)

#OUTPUT
```

```
[['12', '13', '14'], ['12', '13'], ['13', '14'], ['12', '13', '14']]
```

#CO2160714.3 Assignment:

0.7 9 Assignment

Write programs to implement the following Classification methods. (a)distance based (b)statistics based (c)tree based (d)neural Network based

(a) Distance based

[ ]:
```
import pandas as pd
import numpy as np
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
```

```
df = pd.read_csv('bigmacPrice.csv.csv')

# Select the features and target
X = df[['HP', 'Attack', 'Defense', 'Sp_Atk', 'Sp_Def', 'Speed']]
y = df['isLegendary']

# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5)

# Create a k-NN classifier with k=5
knn = KNeighborsClassifier(n_neighbors=5)

# Fit the classifier to the training data
knn.fit(X_train, y_train)

# Predict the labels of the test set
y_pred = knn.predict(X_test)

# Calculate the confusion matrix
cm = confusion_matrix(y_test, y_pred)
print(f'Confusion matrix:\n{cm}')

# Calculate the accuracy of the classifier
accuracy = knn.score(X_test, y_test)
print(f'Accuracy: {accuracy}')

#OUTPUT
```

```
Confusion matrix:
[[337   3]
 [ 11  10]]
Accuracy: 0.961218836565097
```

(b) Statistics based

[ ]:
```
import pandas as pd
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix

df = pd.read_csv('bigmacPrice.csv.csv')

# Select the features and target
X = df[['HP', 'Attack', 'Defense', 'Sp_Atk', 'Sp_Def', 'Speed']]
y = df['isLegendary']

# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5)
```

```
# Create a Gaussian Naive Bayes classifier
gnb = GaussianNB()

# Fit the classifier to the training data
gnb.fit(X_train, y_train)

# Predict the labels of the test set
y_pred = gnb.predict(X_test)

# Calculate the confusion matrix
cm = confusion_matrix(y_test, y_pred)
print(f'Confusion matrix:\n{cm}')

# Calculate the accuracy of the classifier
accuracy = knn.score(X_test, y_test)
print(f'Accuracy: {accuracy}')

#OUTPUT
```

```
Confusion matrix:
[[328  12]
 [  2  19]]
Accuracy: 0.9529085872576177
```

(c) Tree based

[ ]:
```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix


df = pd.read_csv('bigmacPrice.csv.csv')

# Select the features and target
X = df[['HP', 'Attack', 'Defense', 'Sp_Atk', 'Sp_Def', 'Speed']]
y = df['isLegendary']

# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

# Create a decision tree classifier
clf = DecisionTreeClassifier()

# Fit the classifier to the training data
clf.fit(X_train, y_train)
```

```
# Predict the labels of the test set
y_pred = clf.predict(X_test)

# Calculate the confusion matrix
cm = confusion_matrix(y_test, y_pred)
print(f'Confusion matrix:\n{cm}')

# Calculate the accuracy of the classifier
accuracy = clf.score(X_test, y_test)
print(f'Accuracy: {accuracy}')

#OUTPUT
```

```
Confusion matrix:
[[132   5]
 [  4   4]]
Accuracy: 0.937931034482758
```

(d) Neural Network based

[ ]:
```
import pandas as pd
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import train_test_split


df = pd.read_csv('bigmacPrice.csv.csv')

# Select the features and target
X = df[['HP', 'Attack', 'Defense', 'Sp_Atk', 'Sp_Def', 'Speed']]
y = df['isLegendary']

# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

# Create a multi-layer perceptron classifier
clf = MLPClassifier(hidden_layer_sizes=(5,), max_iter=1000)

# Fit the classifier to the training data
clf.fit(X_train, y_train)

# Predict the labels of the test set
y_pred = clf.predict(X_test)

# Calculate the confusion matrix
cm = confusion_matrix(y_test, y_pred)
print(f'Confusion matrix:\n{cm}')

# Calculate the accuracy of the classifier
```

```
accuracy = clf.score(X_test, y_test)
print(f'Accuracy: {accuracy}')

#OUTPUT
```

```
Confusion matrix:
[[133   0]
 [ 12   0]]
Accuracy: 0.9172413793103448
```

0.8 10 Assignment

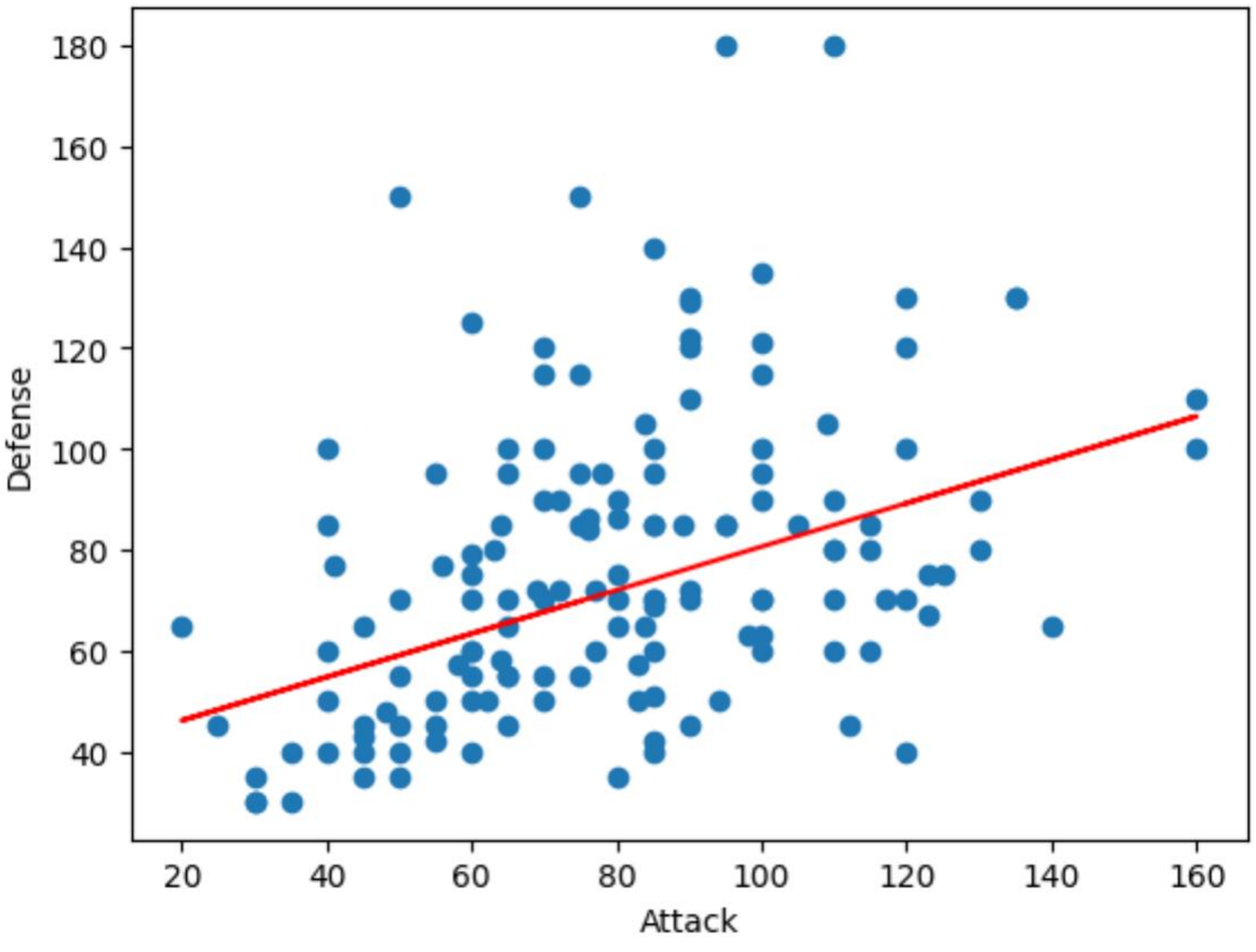
Write a program to implement following Prediction methods: (a) Linear (b) Logistic regression

(a) Linear

[ ]:
```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split


df = pd.read_csv('bigmacPrice.csv.csv')

X = df['Attack'].values.reshape(-1, 1)
y = df['Defense'].values.reshape(-1, 1)

# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

# Create a linear regression model
reg = LinearRegression()

# Fit the model to the training data
reg.fit(X_train, y_train)

# Predict the target values of the test set
y_pred = reg.predict(X_test)

# Calculate the accuracy of the classifier
accuracy = reg.score(X_test, y_test)
print(f'Accuracy: {accuracy}')

# Plot the true vs predicted values
plt.scatter(X_test, y_test)
plt.xlabel('Attack')
plt.ylabel('Defense')
```

```
# Add a regression line to the plot
plt.plot(X_test, y_pred, color='red')

plt.show()

#OUTPUT
```

```
Accuracy: 0.13421000406535633
```



(b) Logistic regression

```
[ ]: import pandas as pd
     from sklearn.linear_model import LogisticRegression
     from sklearn.model_selection import train_test_split

     # Load the Pokémon dataset
     df = pd.read_csv('bigmacPrice.csv.csv')

     # Select the features and target variable
     X = df[['Attack','Defense']]
```

```
y = df['isLegendary']

# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

# Create a logistic regression model
log_reg = LogisticRegression()

# Fit the model to the training data
log_reg.fit(X_train, y_train)

# Predict the target values of the test set
y_pred = log_reg.predict(X_test)

# Calculate the accuracy of the model
accuracy = log_reg.score(X_test, y_test)

print(f'Accuracy: {accuracy}')

#OUTPUT
```

```
Accuracy: 0.9448275862068966
```

#CO2160714.4 Assignment:

##11 Assignment

Using the weka tool for your data set, make the following table for classification and clustering algorithms.

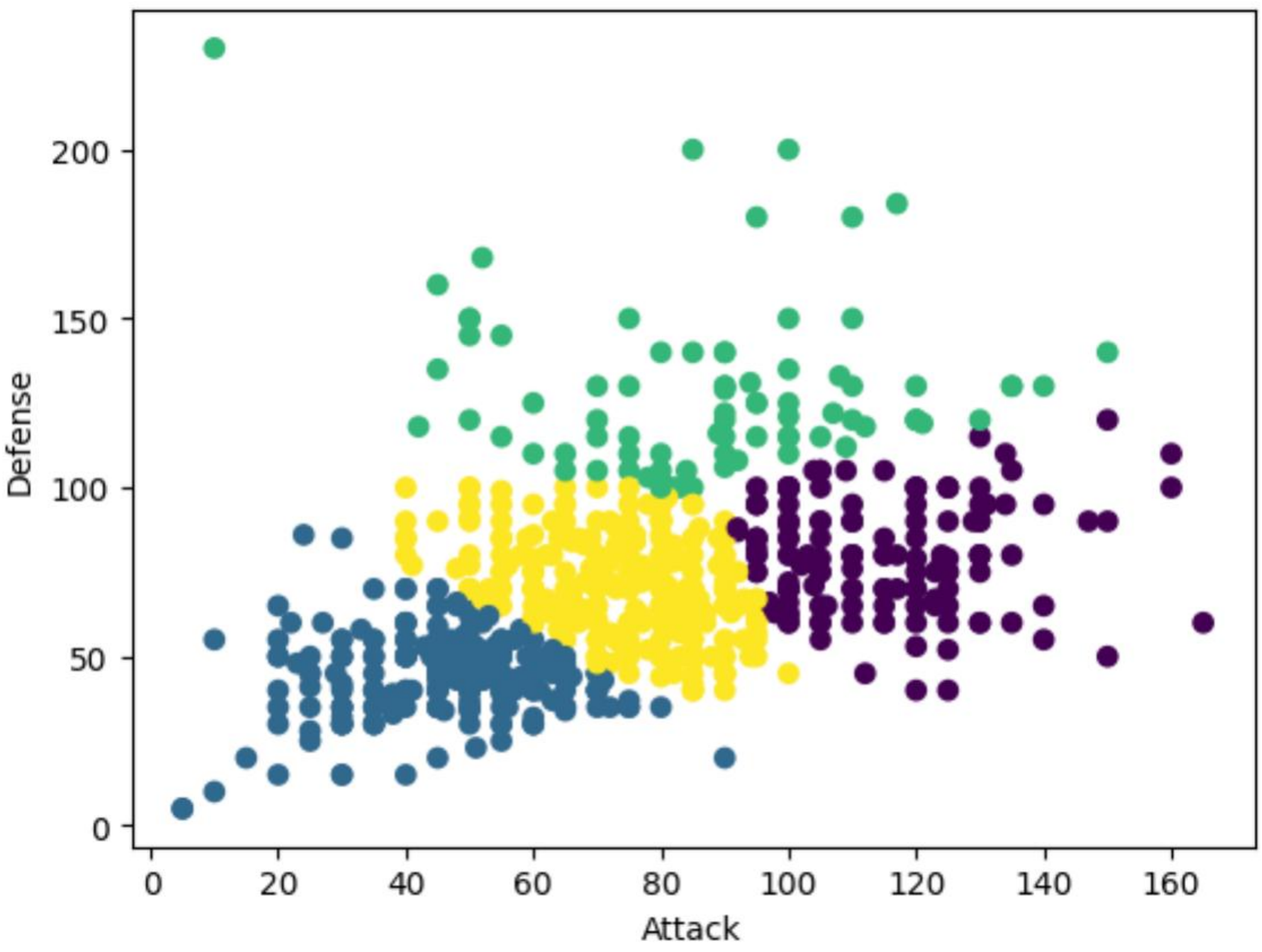
#CO2160714.5 Assignment:

##12 Assignment Implement any 2 unsupervised clustering algorithms.

[ ]:
```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans

# Load the Pokémon dataset
df = pd.read_csv('bigmacPrice.csv.csv')

# Select the features to use for clustering
X = df[['Attack', 'Defense']]

# Create a KMeans model with 4 clusters
kmeans = KMeans(n_clusters=4)
```

```
# Fit the model to the data
kmeans.fit(X)

# Predict the cluster labels for each data point
y_pred = kmeans.predict(X)

# Plot the data points and color them by cluster label
plt.scatter(X['Attack'], X['Defense'], c=y_pred)
plt.xlabel('Attack')
plt.ylabel('Defense')
plt.show()

#OUTPUT
```

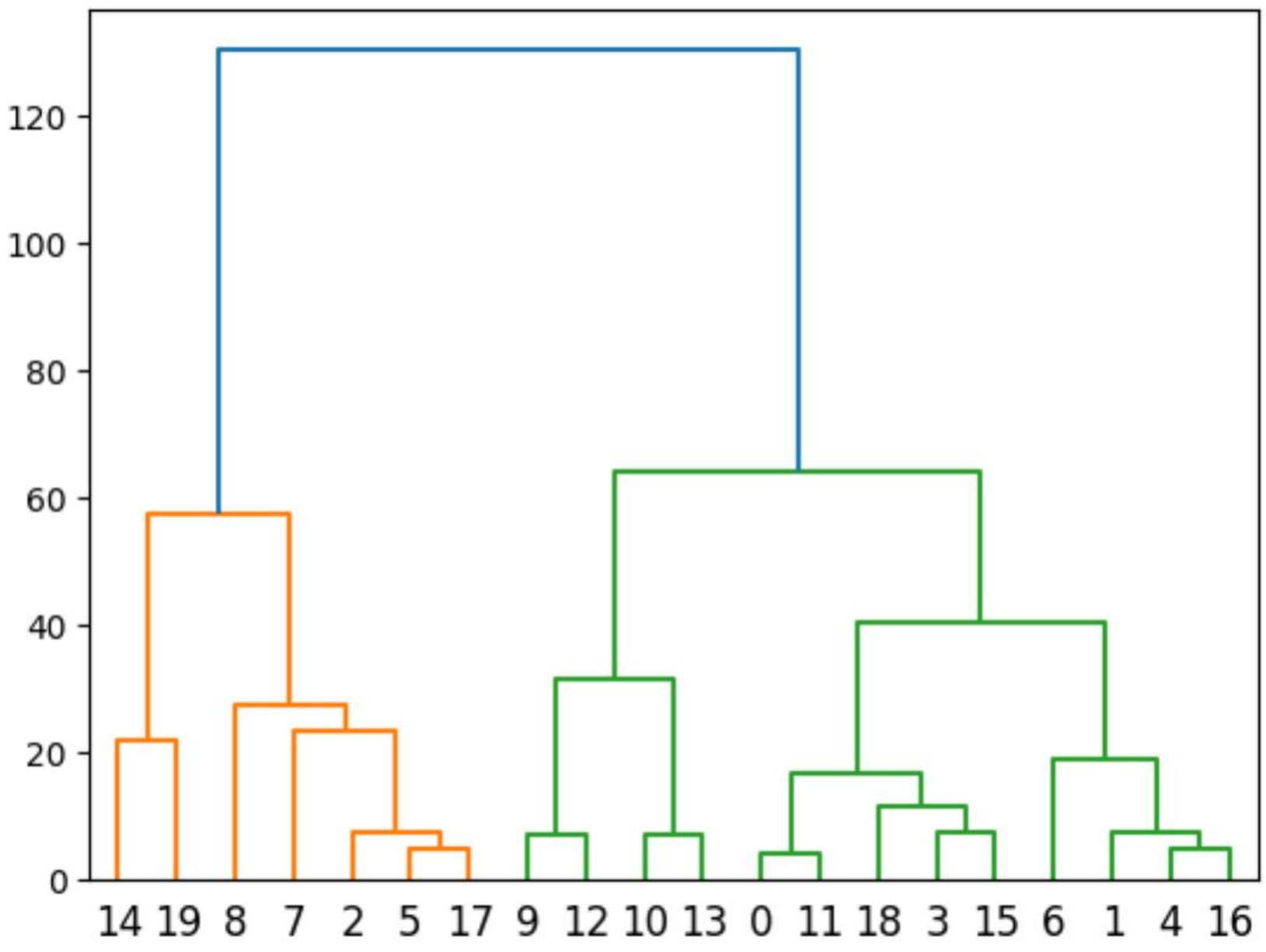
```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
```



[ ]:
```
import pandas as pd
from scipy.cluster.hierarchy import dendrogram, linkage
```

```
3  34.280000  …           NaN           NaN           $
4  32.956127  …           NaN           NaN           $

                                                  img1  \
0  https://tb-static.uber.com/prod/image-proc/pro…
1  https://d1ralsognjng37.cloudfront.net/028932d2…
2  https://d1ralsognjng37.cloudfront.net/a22dc334…
3  https://d1ralsognjng37.cloudfront.net/1c1b3198…
4  https://d1ralsognjng37.cloudfront.net/3b0a4d53…

                                                  img2  \
0  https://tb-static.uber.com/prod/image-proc/pro…
1  https://d1ralsognjng37.cloudfront.net/86583cc1…
2  https://d1ralsognjng37.cloudfront.net/3f86d609…
3  https://d1ralsognjng37.cloudfront.net/c8f6f1ea…
4  https://d1ralsognjng37.cloudfront.net/bb83adfa…

                                                  img3  \
0  https://tb-static.uber.com/prod/image-proc/pro…
1  https://d1ralsognjng37.cloudfront.net/0601a57e…
2  https://d1ralsognjng37.cloudfront.net/e0829c89…
3  https://d1ralsognjng37.cloudfront.net/e669e864…
4  https://d1ralsognjng37.cloudfront.net/2156d6be…

                                                  img4  \
0  https://tb-static.uber.com/prod/image-proc/pro…
1  https://d1ralsognjng37.cloudfront.net/b745dbc7…
2  https://d1ralsognjng37.cloudfront.net/0e41e2d9…
3  https://d1ralsognjng37.cloudfront.net/e4053d9a…
4  https://d1ralsognjng37.cloudfront.net/aa8f2ad2…

                                                  img5             scan_date TID
0  https://tb-static.uber.com/prod/image-proc/pro…  2022-11-09 18:03:43   1
1  https://d1ralsognjng37.cloudfront.net/33efde32…  2022-11-09 18:03:43   2
2  https://d1ralsognjng37.cloudfront.net/6284a890…  2022-11-09 18:03:43   3
3  https://d1ralsognjng37.cloudfront.net/30fe7bae…  2022-11-09 18:03:43   4
4  https://d1ralsognjng37.cloudfront.net/8a90ff8a…  2022-11-09 18:03:43   5

[5 rows x 25 columns]
Number of records/instances = index              1000
city                 999
state               1000
zipcode              997
address             1000
loc_name            1000
loc_number          1000
url                 1000
promotion            110
```

```
# Load the Pokémon dataset
df = pd.read_csv('bigmacPrice.csv.csv')

# Select the features to use for clustering
X = df[['Attack', 'Defense']].head(20)

# Perform hierarchical clustering on the data
Z = linkage(X, 'ward')

# Plot the dendrogram
dendrogram(Z)
plt.show()

#OUTPUT
```



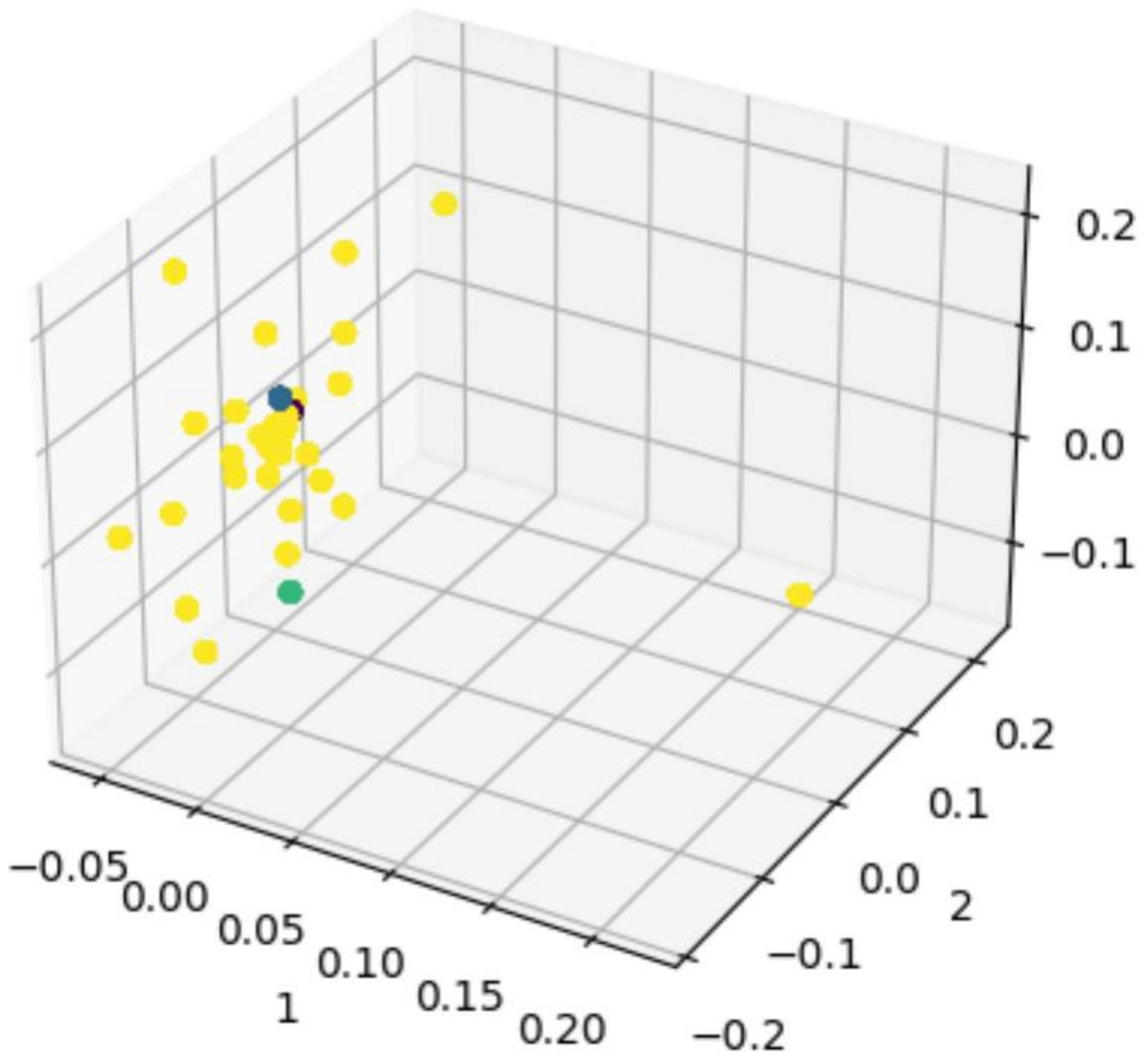
#CO2160714.6 Assignment:

##OEP

Implement text mining/clustering algorithm.

[ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
from mpl_toolkits.mplot3d import Axes3D

df = pd.read_csv('BigmacPrice.csv')

df['name'] = df['currency_code'] + ' ' + str(df['local_price']) + ' ' +␣
 ↪str(df['dollar_price'])
# Apply TfidfVectorizer to the text column
vectorizer = TfidfVectorizer(stop_words='english')
text_matrix = vectorizer.fit_transform(df['name'])
# Apply K-Means clustering with k=4
kmeans = KMeans(n_clusters=4, random_state=42)
kmeans.fit(text_matrix)
# Visualize the clusters in 3D with PCA
pca = PCA(n_components=3)
text_pca = pca.fit_transform(text_matrix.toarray())
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.scatter(text_pca[:, 0], text_pca[:, 1], text_pca[:, 2], c=kmeans.labels_)
ax.set_xlabel('1')
ax.set_ylabel('2')
ax.set_zlabel('3')
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
```


0.2
0.1
0.0
−0.1
0.2
0.1
0.0
−0.1
−0.2
−0.05
0.00
0.05
0.10
0.15
0.20
1
2

```
latitude          1000
longitude         1000
is_open           1000
closed_message     986
delivery_fee         3
delivery_time       14
review_count       374
review_rating      418
price_bucket       856
img1               953
img2               953
img3               953
img4               953
img5               953
scan_date         1000
TID               1000
dtype: int64
Number of incomplete records = 4478
Number of attributes = 25
                                                links                names  \
0  https://www.zomato.com/hyderabad/sahara-bakers…        Sahara Bakers
1   https://www.zomato.com/hyderabad/kfc-abids/order                  KFC
2  https://www.zomato.com/hyderabad/subbaiah-gari…  Subbaiah Gari Hotel
3  https://www.zomato.com/hyderabad/paradise-biry…     Paradise Biryani
4  https://www.zomato.com/hyderabad/pista-house-b…   Pista House Bakery

  ratings                                        cuisine  price
0     3.7            Chinese, Bakery, Sichuan, Pizza, Burger     100
1     3.9   Burger, Fast Food, Biryani, Desserts, Beverages     100
2     4.1                      South Indian, Andhra, Mithai     100
3     3.9                Biryani, Kebab, Desserts, Beverages     100
4     4.3  Fast Food, Sandwich, Pizza, Burger, Wraps, Rol…     100
Number of records/instances = links      657
names      657
ratings    657
cuisine    657
price      657
dtype: int64
Number of incomplete records = 0
Number of attributes = 5
```

##Dataset 1: Swiggy

[ ]:
```
print("total number of rcords:",(data_swiggy.shape[0]))
print("total number of incomplete records:",pd.isnull(data_swiggy).sum())
print("total number of attributes:",(data_swiggy.shape[1]))

#OUTPUT
```

```
total number of rcords: 148541
total number of incomplete records: id                0
name              86
city               0
rating            86
rating_count      86
cost             131
cuisine           99
lic_no           229
link               0
address           86
menu               0
dtype: int64
total number of attributes: 11
```

##Dataset 2: Zomato

[ ]:
```
print("total number of rcords:",(data_zomato.shape[0]))
print("total number of incomplete records:",pd.isnull(data_zomato).sum())
print("total number of attributes:",(data_zomato.shape[1]))

#OUTPUT
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-5-fdfaa3485e81> in <cell line: 1>()
----> 1 print("total number of rcords:",(data_zomato.shape[0]))
      2 print("total number of incomplete records:",pd.isnull(data_zomato).sum())
      3 print("total number of attributes:",(data_zomato.shape[1]))
      4
      5 #OUTPUT

NameError: name 'data_zomato' is not defined
```

##Dataset 3: Uber

[ ]:
```
print("total number of rcords:",(data_Ubereat.shape[0]))
print("total number of incomplete records:",pd.isnull(data_Ubereat).sum())
print("total number of attributes:",(data_Ubereat.shape[1]))

#OUTPUT
```

```
total number of rcords: 1000
total number of incomplete records: index             0
city                1
state               0
zipcode             3
address             0
```

```
loc_name            0
loc_number          0
url                 0
promotion         890
latitude            0
longitude           0
is_open             0
closed_message     14
delivery_fee      997
delivery_time     986
review_count      626
review_rating     582
price_bucket      144
img1               47
img2               47
img3               47
img4               47
img5               47
scan_date           0
TID                 0
dtype: int64
total number of attributes: 25
```

##Dataset 4: McD

[ ]:
```
print("total number of rcords:",(data_BigmacPrice.shape[0]))
print("total number of incomplete records:",pd.isnull(data_BigmacPrice).sum())
print("total number of attributes:",(data_BigmacPrice.shape[1]))

#OUTPUT
```

```
total number of rcords: 1946
total number of incomplete records: date             0
currency_code    0
name             0
local_price      0
dollar_ex        0
dollar_price     0
dtype: int64
total number of attributes: 6
```

##Dataset 5: Profiles

[ ]:
```
print("total number of rcords:",(data_profile.shape[0]))
print("total number of incomplete records:",pd.isnull(data_profile).sum())
print("total number of attributes:",(data_profile.shape[1]))

#OUTPUT
```

```
total number of rcords: 17000
total number of incomplete records: Unnamed: 0             0
gender               2175
age                     0
id                      0
became_member_on        0
income               2175
dtype: int64
total number of attributes: 6
```

##2.Assignment:

2. Write a program to implement data cleaning(incomplete, noisy, inconsistent, redundant) on your data set. Implement each technique.

[A] Binning with means and/or mode, boundary

[ ]:
```
a=[0]*12
for i in range(12):
  b=pd.read_csv('zomato_Hyderabad.csv')
  a[i]=b.ratings[i]
```

[ ]:
```
#printing the data
data=a
print(data)

#OUTPUT
```

```
['3.7', '3.9', '4.1', '3.9', '4.3', '4', '4.2', '4.2', '4.1', '4.3', '4.3',
'4.4']
```

[ ]:
```
#sorting the data
data=np.sort(data)
print(data)

#OUTPUT
```

```
['3.7' '3.9' '3.9' '4' '4.1' '4.1' '4.2' '4.2' '4.3' '4.3' '4.3' '4.4']
```

[ ]:
```
#splitting the data into equal parts
y=np.split(data,3)
print(y)

#OUTPUT
```

```
[array(['3.7', '3.9', '3.9', '4'], dtype='<U3'), array(['4.1', '4.1', '4.2',
'4.2'], dtype='<U3'), array(['4.3', '4.3', '4.3', '4.4'], dtype='<U3')]
```

[ ]:
```
#taking empty array for sorting data in bins
bin1=np.zeros((1,4))
bin2=np.zeros((1,4))
bin3=np.zeros((1,4))
print(bin1)
print(bin2)
print(bin3)

#OUTPUT
```

```
[[0. 0. 0. 0.]]
[[0. 0. 0. 0.]]
[[0. 0. 0. 0.]]
```

[ ]:
```
#sorting the data in the bins
bin1=y[0]
bin2=y[1]
bin3=y[2]
print('bin1:',bin1)
print('bin2:',bin2)
print('bin3:',bin3)

#OUTPUT
```

```
bin1: ['3.7' '3.9' '3.9' '4']
bin2: ['4.1' '4.1' '4.2' '4.2']
bin3: ['4.3' '4.3' '4.3' '4.4']
```

Smoothing by bin means

[ ]:
```
n=len(bin1)
```

[ ]:
```
#calculating mean value for bin1
def Mean(bin1,n):

  sum=0
  for i in range(0,n):
      sum = sum +  float(bin1[i])

  return float(sum/n)
print("Mean:",Mean(bin1,n))

#OUTPUT
```

```
Mean: 3.875
```

[ ]:
```
#calculating mean value for bin2
def Mean(bin2,n):
```

Example table

This is an example of a data table.

Disability Category	Participants	Ballots Completed	Ballots Incomplete/ Terminated	Results	
				Accuracy	Time to complete
Blind	5	1	4	34.5%, n=1	1199 sec, n=1
Low Vision	5	2	3	98.3% n=2 (97.7%, n=3)	1716 sec, n=3 (1934 sec, n=2)
Dexterity	5	4	1	98.3%, n=4	1672.1 sec, n=4
Mobility	3	3	0	95.4%, n=3	1416 sec, n=3