Practical 2

Statistical Data Analysis

SET A

In [1]:
```
#Q1.
import numpy as np
array = np.array([[0,1],[2,3]])
print("\n Original flattened array: \n"
, array)
print(" \n Maximum Value of the above flattened array : \n ", np.max(array))
print(" \n Minimum Value of the above flattened array : \n ", np.min(array))
```

```
Original flattened array:
[[0 1]
[2 3]]

Maximum Value of the above flattened array :
 3

Minimum Value of the above flattened array :
 0
```

In [2]:
```
#Q2.
import numpy as np
#Inserting the two data points
a=np.array((2,3))
b=np.array((4,5))
#Euclidean Distance
print("Euclidean Distance = ", np.linalg.norm(a-b))
```

```
Euclidean Distance =  2.8284271247461903
```

In [3]:
```
#Q3. Create and view a data frame
#import the library
import pandas as pd
import numpy as np
import scipy.stats as s
#Enter Data
data_values={'Name':['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J'],
             'Scores' : [56,62,42,74,32,63,74,84,96,21]
            }
#Create empty dataframe with column names
data=pd.DataFrame.from_dict(data_values)
print(data) #To view the data frame
print("\n Mean Score = ",s.tmean(data["Scores"]) )
print("\n Maximum = ",max(data["Scores"]))
print("\n Minimum = ",min(data["Scores"]))
print("\n Range = ",
 max(data["Scores"]) - min(data["Scores"]) )
q3,q1 = np.percentile(data["Scores"],[75,25])
print("\n Q3 = ", q3)
print("\n Q1 = ", q1)
print("\n IQR = ", q3 - q1)
```

```
  Name  Scores
0    A      56
1    B      62
2    C      42
3    D      74
4    E      32
5    F      63
6    G      74
7    H      84
8    I      96
9    J      21

 Mean Score =  60.4

 Maximum =  96

 Minimum =  21

 Range =  75

 Q3 =  74.0

 Q1 =  45.5

 IQR =  28.5
```

In [5]:
```
# Program to find Manhattan Distance between two points
import math
def manhattan(a,b):
    return sum(abs(val1 - val2) for val1, val2 in zip(a,b))
```

In [6]:
```
#consider any two points
a=[2,3]
b=[4,5]
print ("Points :",a,b)
print("\n Manhattan Distance = ", manhattan(a,b))
```

```
Points : [2, 3] [4, 5]

 Manhattan Distance =  4
```

Program to find Manhattan distance between all pairs of points

In [9]:
```
#Q4. Program to find Manhattan distance between all pairs of points
import math
def manhattan(a,b,n):
    sum = 0
    i = 0
    for i in range(n):
        sum += abs(a[i]-b[i])
    return sum
```

In [10]:
```
#Example
a=[3,5,5,6,5,4,3]
b=[-2,3,2,-5,2,3,-1]
n=len(a) #or len(b)
print("Manhattan Distance = ", manhattan(a,b,n))
```

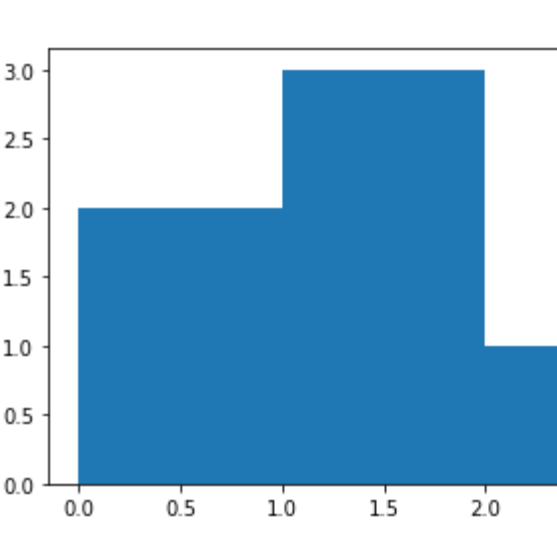
```
Manhattan Distance =  29
```

In [12]:
```
#Manhattan and Euclidean Distance
import scipy.spatial as sp
print("\n Manhattan Distance = ",sp.distance.minkowski(a,b,1))
print("\n Euclidean Distance = ",sp.distance.minkowski(a,b,2))
```

```
 Manhattan Distance =  29.0

 Euclidean Distance =  13.601470508735444
```

In [13]:
```
#Q5.
import numpy as np
import matplotlib.pyplot as plt
n=np.array([0.5, 0.7, 1.0, 1.2, 1.3, 2.1])
b=np.array([0,1,2,3])
print("\n nums:",n)
print("\n bins:",b )
print("\n Result: \n"
,np.histogram(n,b))
print("\n")
plt.hist(n,b)
plt.show()
```

```
 nums: [0.5 0.7 1.  1.2 1.3 2.1]

 bins: [0 1 2 3]

 Result:
 (array([2, 3, 1], dtype=int64), array([0, 1, 2, 3]))
```



In [14]:
```
#Q6.Create and view a data frame
#import the library
import pandas as pd
import numpy as np
import scipy.stats as s
#Enter Data
data_values={'Name':['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J'],
 'Graduation Percentage' : [56,62,42,74,32,63,74,84,96,21],
 'Age' : [26, 28, 20, 15, 20, 16, 18, 17, 22, 21]
 }
#Create empty dataframe with column names
data=pd.DataFrame.from_dict(data_values)
data #To view the data frame
```

Out[14]:

	Name	Graduation Percentage	Age
0	A	56	26
1	B	62	28
2	C	42	20
3	D	74	15
4	E	32	20
5	F	63	16
6	G	74	18
7	H	84	17
8	I	96	22
9	J	21	21

In [15]:
```
print("\n Average age of students = ",s.tmean(data["Age"]) )
print("\n Average Graduation Percentage = ",s.tmean(data["Graduation Percentage"]) )
print("\n All Basic Statistics of Data \n ")
data.describe(include='all')
```

```
 Average age of students =  20.3

 Average Graduation Percentage =  60.4

 All Basic Statistics of Data
```

Out[15]:

	Name	Graduation Percentage	Age
count	10	10.000000	10.000000
unique	10	NaN	NaN
top	A	NaN	NaN
freq	1	NaN	NaN
mean	NaN	60.400000	20.300000
std	NaN	23.381854	4.191261
min	NaN	21.000000	15.000000
25%	NaN	45.500000	17.250000
50%	NaN	62.500000	20.000000
75%	NaN	74.000000	21.750000
max	NaN	96.000000	28.000000

In [18]:
```
print("Measures of Dispersion and Position in the Distribution")
r=max(data["Graduation Percentage"]) - min(data["Graduation Percentage"])
print("Value of Range in the Distribution = ", r)
s=round(data["Graduation Percentage"].std(),3)
print("Value of Standard Deviation in the Distribution = ", s)
v=round(data["Graduation Percentage"].var(),3)
print("Value of Variance in the Distribution = ", v)
```

```
Measures of Dispersion and Position in the Distribution
Value of Range in the Distribution =  75
Value of Standard Deviation in the Distribution =  23.382
Value of Variance in the Distribution =  546.711
```

SET B

In [20]:
```
#Q1.
import pandas as pd
data=pd.read_csv("C:\\iris.csv")
```

In [21]:
```
data.sample(13)
```

Out[21]:

	sepal.length	sepal.width	petal.length	petal.width	variety
102	7.1	3.0	5.9	2.1	Virginica
43	5.0	3.5	1.6	0.6	Setosa
100	6.3	3.3	6.0	2.5	Virginica
131	7.9	3.8	6.4	2.0	Virginica
72	6.3	2.5	4.9	1.5	Versicolor
61	5.9	3.0	4.2	1.5	Versicolor
12	4.8	3.0	1.4	0.1	Setosa
117	7.7	3.8	6.7	2.2	Virginica
68	6.2	2.2	4.5	1.5	Versicolor
44	5.1	3.8	1.9	0.4	Setosa
114	5.8	2.8	5.1	2.4	Virginica
73	6.1	2.8	4.7	1.2	Versicolor
119	6.0	2.2	5.0	1.5	Virginica

In [22]:
```
from pandas.api.types import is_numeric_dtype
print("Minimum and Maximum for all numeric attributes\n")
for col in data.columns:
 if is_numeric_dtype(data[col]):
    print('%s:' % (col))
    print('\t Minimum = ',data[col].min())
    print('\t Maximum = ',data[col].max())
```

```
Minimum and Maximum for all numeric attributes

sepal.length:
         Minimum =  4.3
         Maximum =  7.9
sepal.width:
         Minimum =  2.0
         Maximum =  4.4
petal.length:
         Minimum =  1.0
         Maximum =  6.9
petal.width:
         Minimum =  0.1
         Maximum =  2.5
```

In [23]:
```
#Q2.
print("Number of records for different variety/class attribute \n")
data['variety'].value_counts()
```

```
Number of records for different variety/class attribute
```

Out[23]:
```
Setosa        50
Versicolor    50
Virginica     50
Name: variety, dtype: int64
```

In [24]:
```
#Q3.
import pandas as pd
from pandas.api.types import is_numeric_dtype
print("Iris Dataset : Column wise Mean and Median \n")
for col in data.columns:
    if is_numeric_dtype(data[col]):
        print('%s:' % (col))
        print('\t Mean = %.2f' % data[col].mean())
        print('\t Median = %.2f' % data[col].median())
```

```
Iris Dataset : Column wise Mean and Median

sepal.length:
         Mean = 5.84
         Median = 5.80
sepal.width:
         Mean = 3.06
         Median = 3.00
petal.length:
         Mean = 3.76
         Median = 4.35
petal.width:
         Mean = 1.20
         Median = 1.30
```

SET C

In [25]:
```
#Q1. Program to find Minkowskii Distance between two points
from math import *
from decimal import Decimal
def nth_root(value,root):
    root_value = 1/float(root)
    return round(Decimal(value)**
                 Decimal(root_value),3)
def minkowski(a,b,n):
    return(nth_root(sum(pow(abs(i-j),n)
                        for i,j in zip(a,b)),n))
```

In [26]:
```
a=[-1,5]
b=[2,4]
n=len(a) #OR root value
print("\n Minkowski Distance = ",minkowski(a,b,n))
```

```
 Minkowski Distance =  3.162
```

In [29]:
```
#Q2.
import numpy as np
a = np.arange(9).reshape((3,3))
print("\nOriginal flattened array:")
print(a)
print("\nWeighted average along the specified axis of the above flattened array:")
print(np.average(a, axis=1, weights=[1./4, 2./4, 2./4]))
```

```
Original flattened array:
[[0 1 2]
 [3 4 5]
 [6 7 8]]

Weighted average along the specified axis of the above flattened array:
[1.2 4.2 7.2]
```

In [30]:
```
#Q3.
import numpy as np
x = np.array([0, 1, 3])
y = np.array([2, 4, 5])
print("\nOriginal array1:")
print(x)
print("\nOriginal array1:")
print(y)
print("\nCross-correlation of the said arrays:\n",np.cov(x, y))
```

```
Original array1:
[0 1 3]

Original array1:
[2 4 5]

Cross-correlation of the said arrays:
 [[2.33333333 2.16666667]
 [2.16666667 2.33333333]]
```

In [33]:
```
#Q4. Wholesale Customers Data from UCI
import pandas as pd
data=pd.read_csv("C:\\Wholesale customers data.csv")
```

In [34]:
```
data.describe()
```

Out[34]:

	Channel	Region	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicassen
count	440.000000	440.000000	440.000000	440.000000	440.000000	440.000000	440.000000	440.000000
mean	1.322727	2.543182	12000.297727	5796.265909	7951.277273	3071.931818	2881.493182	1524.870455
std	0.468052	0.774272	12647.328865	7380.377175	9503.162829	4854.673333	4767.854448	2820.105937
min	1.000000	1.000000	3.000000	55.000000	3.000000	25.000000	3.000000	3.000000
25%	1.000000	2.000000	3127.750000	1533.000000	2153.000000	742.250000	256.750000	408.250000
50%	1.000000	3.000000	8504.000000	3627.000000	4755.500000	1526.000000	816.500000	965.500000
75%	2.000000	3.000000	16933.750000	7190.250000	10655.750000	3554.250000	3922.000000	1820.250000
max	2.000000	3.000000	112151.000000	73498.000000	92780.000000	60869.000000	40827.000000	47943.000000

In [36]:
```
import pandas as pd
from pandas.api.types import is_numeric_dtype
print("Wholesale Customers Dataset : Column wise Mean for numeric attributes \n")
for col in data.columns:
    if is_numeric_dtype(data[col]):
        print('%s:' % (col))
        print('\t Mean = %.2f' % data[col].mean())
print(" \nCount total NaN at each column in a DataFrame : \n",data.isnull().sum())
print(" \nTotal number of missing values in the dataset : ",data.isnull().sum().sum())
```

```
Wholesale Customers Dataset : Column wise Mean for numeric attributes

Channel:
         Mean = 1.32
Region:
         Mean = 2.54
Fresh:
         Mean = 12000.30
Milk:
         Mean = 5796.27
Grocery:
         Mean = 7951.28
Frozen:
         Mean = 3071.93
Detergents_Paper:
         Mean = 2881.49
Delicassen:
         Mean = 1524.87

Count total NaN at each column in a DataFrame :
 Channel             0
Region              0
Fresh               0
Milk                0
Grocery             0
Frozen              0
Detergents_Paper    0
Delicassen          0
dtype: int64

Total number of missing values in the dataset :  0
```

In [33]:
```
#Q5.
import pandas as pd
data=pd.read_csv("C:\\Users\\farhe\\Downloads\\nursery.data.csv",header=None)
```

In [34]:
```
data.columns=['one','two','three','four','five','six','seven','eight','nine']
```

In [37]:
```
data.head(20)
```

Out[37]:

	one	two	three	four	five	six	seven	eight	nine
0	usual	proper	complete	1	convenient	convenient	nonprob	recommended	recommend
1	usual	proper	complete	1	convenient	convenient	nonprob	priority	priority
2	usual	proper	complete	1	convenient	convenient	nonprob	not_recom	not_recom
3	usual	proper	complete	1	convenient	convenient	slightly_prob	recommended	recommend
4	usual	proper	complete	1	convenient	convenient	slightly_prob	priority	priority
5	usual	proper	complete	1	convenient	convenient	slightly_prob	not_recom	not_recom
6	usual	proper	complete	1	convenient	convenient	problematic	recommended	priority
7	usual	proper	complete	1	convenient	convenient	problematic	priority	priority
8	usual	proper	complete	1	convenient	convenient	problematic	not_recom	not_recom
9	usual	proper	complete	1	convenient	inconv	nonprob	recommended	very_recom
10	usual	proper	complete	1	convenient	inconv	nonprob	priority	priority
11	usual	proper	complete	1	convenient	inconv	nonprob	not_recom	not_recom
12	usual	proper	complete	1	convenient	inconv	slightly_prob	recommended	very_recom
13	usual	proper	complete	1	convenient	inconv	slightly_prob	priority	priority
14	usual	proper	complete	1	convenient	inconv	slightly_prob	not_recom	not_recom
15	usual	proper	complete	1	convenient	inconv	problematic	recommended	priority
16	usual	proper	complete	1	convenient	inconv	problematic	priority	priority
17	usual	proper	complete	1	convenient	inconv	problematic	not_recom	not_recom
18	usual	proper	complete	1	less_conv	convenient	nonprob	recommended	very_recom
19	usual	proper	complete	1	less_conv	convenient	nonprob	priority	priority

In [39]:
```
#Check categories names
data.eight.value_counts().index
```

Out[39]:
```
Index(['recommended', 'priority', 'not_recom'], dtype='object')
```

In [42]:
```
data.eight.value_counts()
```

Out[42]:
```
recommended    4320
priority       4320
not_recom      4320
Name: eight, dtype: int64
```

In [40]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12960 entries, 0 to 12959
Data columns (total 9 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   one     12960 non-null  object
 1   two     12960 non-null  object
 2   three   12960 non-null  object
 3   four    12960 non-null  object
 4   five    12960 non-null  object
 5   six     12960 non-null  object
 6   seven   12960 non-null  object
 7   eight   12960 non-null  object
 8   nine    12960 non-null  object
dtypes: object(9)
memory usage: 911.4+ KB
```

In [41]:
```
#Group by proper
import numpy as np
data_by_proper=data.groupby('eight')
data_by_proper.count()
```

Out[41]:

	one	two	three	four	five	six	seven	nine
eight								
not_recom	4320	4320	4320	4320	4320	4320	4320	4320
priority	4320	4320	4320	4320	4320	4320	4320	4320
recommended	4320	4320	4320	4320	4320	4320	4320	4320

In [46]:
```
data_by_proper.describe()
```

Out[46]:

	one				two				three	...	six				seven			
	count	unique	top	freq	count	unique	top	freq	count	unique	...	top	freq	count	unique	top	freq
eight																	
not_recom	4320	3	usual	1440	4320	5	proper	864	4320	4	...	convenient	2160	4320	3	nonprob	1440
priority	4320	3	usual	1440	4320	5	proper	864	4320	4	...	convenient	2160	4320	3	nonprob	1440
recommended	4320	3	usual	1440	4320	5	proper	864	4320	4	...	convenient	2160	4320	3	nonprob	1440

3 rows × 32 columns

In [47]:
```
#Q6.Create and view a data frame
#import the library
import pandas as pd
import numpy as np
import scipy.stats as s
#Enter Data
data_values={'Student' : ["1","2","3","4","5","6","7","8","9","10"],
 'Subject 1':[41,62,35,15,21,65,84,75,42,95],
 'Subject 2' : [56,62,42,74,32,63,74,84,96,21],
 'Subject 3' : [26, 28, 20, 15, 20, 16, 18, 17, 22, 21],
 'Subject 4' : [41,75,84,62,13,56,42,84,95,23],
 'Subject 5' : [45,74,62,31,21,54,45,86,95,32]
 }
#Create empty dataframe with column names
data=pd.DataFrame.from_dict(data_values)
data #To view the data frame
```

Out[47]:

	Student	Subject 1	Subject 2	Subject 3	Subject 4	Subject 5
0	1	41	56	26	41	45
1	2	62	62	28	75	74
2	3	35	42	20	84	62
3	4	15	74	15	62	31
4	5	21	32	20	13	21
5	6	65	63	16	56	54
6	7	84	74	18	42	45
7	8	75	84	17	84	86
8	9	42	96	22	95	95
9	10	95	21	21	23	32

In [48]:
```
from pandas.api.types import is_numeric_dtype
from scipy.stats.mstats import gmean
import statistics as stat
print("Subject wise Mean \n")
for col in data.columns:
    if is_numeric_dtype(data[col]):
        print('%s:' % (col))
        print('\t Arithmetic Mean = %.2f' % data[col].mean())
        print('\t Geometric Mean = %.2f' % gmean(data[col]))
        print('\t Harmonic Mean = %.2f' % stat.harmonic_mean(data[col]))
```

```
Subject wise Mean

Subject 1:
         Arithmetic Mean = 53.50
         Geometric Mean = 46.35
         Harmonic Mean = 38.71
Subject 2:
         Arithmetic Mean = 60.40
         Geometric Mean = 55.41
         Harmonic Mean = 49.53
Subject 3:
         Arithmetic Mean = 20.30
         Geometric Mean = 19.93
         Harmonic Mean = 19.58
Subject 4:
         Arithmetic Mean = 57.50
         Geometric Mean = 49.59
         Harmonic Mean = 39.96
Subject 5:
         Arithmetic Mean = 54.50
         Geometric Mean = 49.33
         Harmonic Mean = 44.27
```

In [49]:
```
#Q7.
import pandas as pd
data=pd.read_csv("C:\\iris.csv")
```

In [ ]:
```
! pip install https://github.com/pandas-profiling/pandas-profiling/archive/master.zip
```

In [51]:
```
import pandas_profiling
profile = data.profile_report(title="Statistal Data Analysis")
profile
```

Statistal Data Analysis

Overview | Variables | Interactions | Correlations | Missing values | Sample | Duplicate rows

Overview

Overview | Alerts 7 | Reproduction

Dataset statistics

Number of variables	5
Number of observations	150
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	1
Duplicate rows (%)	0.7%
Total size in memory	6.0 KiB
Average record size in memory	40.9 B

Variable types

Numeric	4
Categorical	1

Variables

sepal.length
Real number ($\mathbb{R}_{\geq 0}$)

Distinct	35	Minimum	4.3
Distinct	23.3%	Maximum	7.9

Out[51]:

In [52]:
```
#Saving the file
profile.to_file("Data Analysis.html")
```