Data Science Lab

(20 MCA 241)

LAB RECORD

Submitted in partial fulfillment of the requirements for the award of the degree of Master of Computer Applications of A P J Abdul Kalam Technological University

Submitted by:

JOMON SEBASTIAN (SJC21MCA-2026)

MASTER OF COMPUTER APPLICATIONS

ST.JOSEPH'S COLLEGE OF ENGINEERING AND TECHNOLOGY, PALAI

CHOONDACHERRY P.O, KOTTAYAM

KERALA

DECEMBER 2022

ST. JOSEPH' S COLLEGE OF ENGINEERING AND TECHNOLOGY, PALAI

(*An ISO 9001: 2008 Certified College*)
CHOONDACHERRY P.O, KOTTAYAM KERALA

CERTIFICATE

This is to certify that the Data Science Lab Record (20 MCA241) ***submitted*** *by* ***Jomon Sebastian*** *student of* ***Third*** *semester* ***MCA*** *at* ***ST. JOSEPH'S COLLEGE OF ENGINEERING AND TECHNOLOGY***, ***PALAI*** *in partial fulfilment for the award of Master of Computer Applications is a bonafide record of the lab work carried out by him under our guidance and supervision. This record in any form has not been submitted to any other University or Institute for any purpose.*

Ms. Liz George
Faculty In- Charge

Mr. Anish Augustine
(HoD In Charge-MCA)

Submitted for the End Semester Examination held on_____

Examiner 1:

Examiner 2:

DECLARATION

I **Jomon Sebastian**, do hereby declare that the *Data Science Lab Record (20 MCA 241)* is a record of work carried out under the guidance **Ms. Liz George**, Asst. Professor, Department of MCA, SJCET, Palai as per the requirement of the curriculum of Master of Computer Applications Programme of A P J Abdul Kalam Technological University, Thiruvananthapuram. Further, I also declare that this record has not been submitted, full or part thereof, in any University / Institution for the award of any Degree / Diploma.

Place: Choondacherry
Date :

Jomon Sebastian
(SJC21MCA-2026)

INDEX

1. Create two 2D arrays using array object and

a. Add the 2 matrices and print it

b. Subtract 2 matrices

c. Multiply the individual elements of matrix

d. Divide the elements of the matrices

e. Perform matrix multiplication

f. Display transpose of the matrix

g. Sum of diagonal elements of a matrix

Program

```
import numpy as np
m1=np.array([[1,2,3],
             [6,7,8],
            [1,4,5]
              ])
m2=np.array([[9,7,6,],
              [3,4,5],
              [4,5,6]
              ])
print("Matrix 1:\n",m1)
print("Matrix  2:\n",m2)
r1 = np.add(m1, m2)
print("Sum of Matrices:")
print(r1)
r2 = np.subtract(m1, m2)
```

```
print("Difference of Matrices:")
print(r2)
r3=np.multiply(m1,m2)
print('\n Multiplication of individual elements of matrix: \n',r3)
r4=np.divide(m1,m2)
print('\n Division: \n',r4)
r5 = np.matmul(m1,m2)
print('\nMultiplication: \n',r5)
print("\n Transpose of Matrix 1:\n",np.transpose(m1));
print("\n transpose of Matrix 2:\n",np.transpose(m2));
print("\n Sum of diagonal elements of Matrix 1:\n",np.trace(m1));
print("\n Sum of diagonal elements of Matrix 2:\n",np.trace(m2));
```

Output

```
Matrix 1:
 [[1 2 3]
 [6 7 8]
 [1 4 5]]
Matrix 2:
 [[9 7 6]
 [3 4 5]
 [4 5 6]]
Sum of Matrices:
[[10  9  9]
 [ 9 11 13]
 [ 5  9 11]]
Difference of Matrices:
[[-8 -5 -3]
 [ 3  3  3]
 [-3 -1 -1]]

 Multiplication of individual elements of matrix:
 [[ 9 14 18]
 [18 28 40]
 [ 4 20 30]]

 Division:
 [[0.11111111 0.28571429 0.5       ]
 [2.         1.75       1.6       ]
 [0.25       0.8        0.83333333]]

Multiplication:
 [[ 27  30  34]
 [107 110 119]
 [ 41  48  56]]
```

```
 Transpose of Matrix 1:
 [[1 6 1]
 [2 7 4]
 [3 8 5]]

 transpose of Matrix 2:
 [[9 3 4]
 [7 4 5]
 [6 5 6]]

 Sum of diagonal elements of Matrix 1:
 13

 Sum of diagonal elements of Matrix 2:
 19

Process finished with exit code 0
```

2. Create a square matrix with random integer values(use randint()) and use appropriate functions to find:

i) inverse
ii) rank of matrix
iii) Determinant
iv) transform matrix into 1D array
v) eigen values and vectors

Program

```
import numpy as np

array = np.random.randint(10, size=(3, 3))

print("\nSquare Matrix is:\n",array)

print("\nInverse Matrix:\n",np.linalg.inv(array))

rank = np.linalg.matrix_rank(array)

print("\nRank of the given Matrix is : ",rank)

det = np.linalg.det(array)

print("\nDeterminant of given matrix:",int(det))

res = array.flatten()

print("\nNew resulting array: ", res)

w,v = np.linalg.eig(array)

print("\nPrinting the Eigen values of the given square array:\n",w)

print("\nPrinting Right eigenvectors of the given square array:\n",v)
```

Output

```
Square Matrix is:
 [[1 8 9]
 [6 3 1]
 [2 2 1]]

Inverse Matrix:
 [[ 0.04347826  0.43478261 -0.82608696]
 [-0.17391304 -0.73913043  2.30434783]
 [ 0.26086957  0.60869565 -1.95652174]]

Rank of the given Matrix is :  3

Determinant of given matrix: 23

New resulting array:  [1 8 9 6 3 1 2 2 1]

Printing the Eigen values of the given square array:
 [10.8291     -5.43857354 -0.39052646]

Printing Right eigenvectors of the given square array:
 [[-0.74577961 -0.81686351  0.31378719]
 [-0.60669499  0.57178538 -0.73321656]
 [-0.27519805  0.07612808  0.60326701]]

Process finished with exit code 0
```

3. Create a matrix X with suitable rows and columns

i) Display the cube of each element of the matrix using different methods (use multiply(), *, power(),)**

ii) Display identity matrix of the given square matrix.

iii) Display each element of the matrix to different powers.

iv) Create a matrix Y with same dimension as X and perform the operation X2+2Y

```
import numpy as np
x=np.array([[2,3],
        [3,4]])
print(x)
print("cube using multiply fn :\n",np.multiply(x,np.multiply(x,x)))
print("cube using power fn :\n",np.power(x,3))
print("cube using '**'\n",x**3)
print("cube using '*'\n",x*x*x)
print('identity matrix is:\n',np.identity(2,dtype=int))
print("Display each element of the matrix to different powers.\n",np.power(x,[[1,2],[3,4]]))
y=np.array([[3,7],
        [8,9]])
print("x^2+2y\n",(x**2)+(2*y))
```

Output

```
[[2 3]
 [3 4]]
cube using multiply fn :
 [[ 8 27]
 [27 64]]
cube using power fn :
 [[ 8 27]
 [27 64]]
cube using '**'
 [[ 8 27]
 [27 64]]
cube using '*'
 [[ 8 27]
 [27 64]]
identity matrix is:
 [[1 0]
 [0 1]]
Display each element of the matrix to different powers.
 [[  2   9]
 [ 27 256]]
x^2+2y
 [[10 23]
 [25 34]]

Process finished with exit code 0
```

4. Create a 2 Dimensional array with 4 rows and 4 columns.

a. Display all elements excluding the first row

b. Display all elements excluding the last column

c. Display the elements of 1st and 2nd column in 2nd and 3rd row

d. Display the elements of 2nd and 3rd column

e. Display 2nd and 3rd element of 1st row

f. Display the elements from indices 4 to 10 in descending order(use –values)

Program

```
import numpy as np
ar=np.array([[1,2,3,4],
        [4,6,7,3],
        [8,9,0,1],
        [5,6,3,2]
        ])
print("Display all elements excluding the first row\n",ar[1:4])
print("Display all elements excluding the last column\n",ar[:,0:3])
print("Display the elements of 1 st and 2 nd column in 2 nd and 3 rd row\n",ar[1:3,0:2])
print("Display the elements of 2 nd and 3 rd column\n",ar[:,1:3])
print("Display 2 nd and 3 rd element of 1 st row\n",ar[0,1:3])
```

Output

```
Display all elements excluding the first row
 [[4 6 7 3]
 [8 9 0 1]
 [5 6 3 2]]
Display all elements excluding the last column
 [[1 2 3]
 [4 6 7]
 [8 9 0]
 [5 6 3]]
Display the elements of 1 st and 2 nd column in 2 nd and 3 rd row
 [[4 6]
 [8 9]]
Display the elements of 2 nd and 3 rd column
 [[2 3]
 [6 7]
 [9 0]
 [6 3]]
Display 2 nd and 3 rd element of 1 st row
 [2 3]

Process finished with exit code 0
```

5. Write a program to perform the SVD of a given matrix.

Program

```
import numpy as np

arr = np.array([[0, 0, 0, 0, 1], [2, 0, 0, 1, 3],
                [4, 0, 2, 0, 0], [3, 2, 0, 0, 1]],
               dtype=np.float32)

print("Original array:")
print(arr)
U, s, V = np.linalg.svd(arr, full_matrices=False)
print("\nFactor of the given array by Singular Value Decomposition:")
print("\nU=", U, "\n\ns=",s,"\n\nV=",V)
```

Output

```
Original array:
[[0. 0. 0. 0. 1.]
 [2. 0. 0. 1. 3.]
 [4. 0. 2. 0. 0.]
 [3. 2. 0. 0. 1.]]

Factor of the given array  by Singular Value Decomposition:

U= [[-0.05757337  0.27878675 -0.09291478 -0.95411223]
 [-0.48248452  0.74779236 -0.35807404  0.28248587]
 [-0.66688335 -0.6010665  -0.43040496 -0.09347294]
 [-0.564943    0.04246859  0.8233477  -0.03368138]]

s= [5.9961324  3.0330286  1.9235512  0.38351715]

V= [[-0.8884613  -0.1884358  -0.22243783 -0.08046596 -0.34521753]
 [-0.25758928  0.02800408 -0.39634743  0.24654973  0.8455681 ]
 [ 0.01677891  0.85607046 -0.4475108  -0.18615259 -0.17872632]
 [ 0.23476347 -0.1756447  -0.4874512   0.7365664  -0.3659184 ]]

Process finished with exit code 0
```

6. Solving systems of equations with numpy

One of the more common problems in linear algebra is solving a matrix-vector equation.

Here is an example. We seek the vector x that solves the equation

A X = b

Where

And X=A-1 b.

Numpy provides a function called solve for solving such eauations.

Write a program to find out the value of X using solve().

Program

```
import numpy as np

A=np.matrix([[2,1,-2],

        [3,0,1],

        [1,1,-1]])


b=np.matrix([-3,5,-2])

y = np.linalg.inv(A)

X=np.matmul(b,y)

print("X:\n\n{}\n".format(X))
```

Output

```
X:
[[-4.25 -1.25  9.25]]


Process finished with exit code 0
```

7. Program to create a line graph with the specified style properties, given the information regarding the car details.

Sarah bought a new car in 2001 for $24,000. The dollar value of her car changed each year as shown in the table below.

Value of Sarah's Car

Year	Value
2001	$24,000
2002	$22,500
2003	$19,700
2004	$17,500
2005	$14,500
2006	$10,000
2007	$ 5,800

Represent the following information using a line graph with following style properties

- **X- axis - Year**

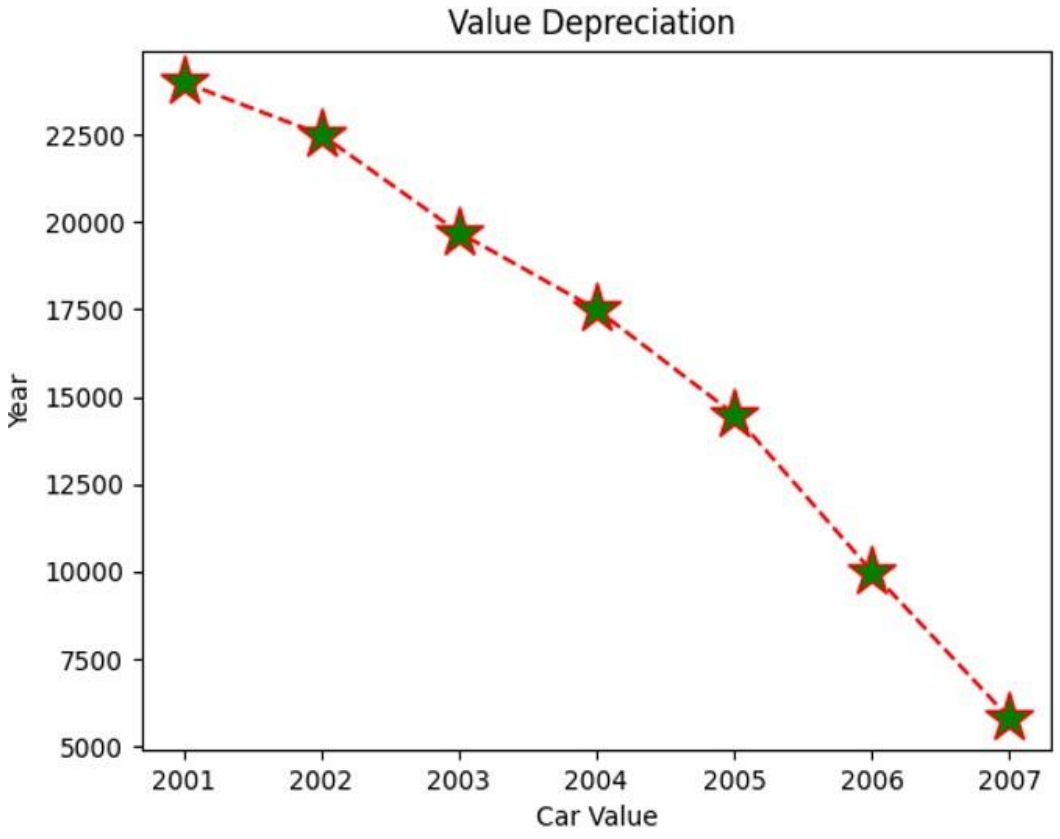
  Y –axis - Car Value
- **title –Value Depreciation (left Aligned)**
- **Line Style dashdot and Line-color should be red**
- **point using * symbol with green color and size 20**

Program

```
import matplotlib.pyplot as plt

# initializing the data

x = [2001, 2002, 2003, 2004, 2005, 2006, 2007]

y = [24000, 22500, 19700, 17500, 14500, 10000, 5800 ]
```

```
plt.plot(x, y, color = 'red',
        linestyle = '--', marker = '*',
        markerfacecolor = 'green', markersize = 20)

plt.title("Value Depreciation ")

plt.ylabel("Year")
plt.xlabel("Car Value")
plt.show()
```

Output



8. Program to represent the daily sales of the 2 items in a shop using line graph with grids and appropriate style properties.

Following table gives the daily sales of the following items in a shop

Day	Mon	Tues	Wed	Thurs	Fri
Drinks	300	450	150	400	650
Food	400	500	350	300	500

Use subplot function to draw the line graphs with grids(color as blue and line style dotted) for the above information as 2 separate graphs in two rows

a) Properties for the Graph 1:

- **X label- Days of week**
- **Y label-Sale of Drinks**
- **Title-Sales Data1 (right aligned)**
- **Line –dotted with cyan color**
- **Points- hexagon shape with color magenta and outline black**

b) Properties for the Graph 2:

- **X label- Days of Week**
- **Y label- Sale of Food**
- **Title-Sales Data2 (center aligned)**
- **Line –dashed with yellow color**
- **Points- diamond shape with color green and outline red**

Program

```
import matplotlib.pyplot as plt

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 5))

x1 = ['Mon', 'Tues', 'Wed','Thurs', 'Fri']

y1 = [300, 450, 150, 400, 650]
```

```
y2 = [400, 500, 350, 300, 500]
fig.suptitle('Daily Sales', fontsize=20)
plt.subplot(1, 2, 1)
plt.title(label="Sales Data1 ", loc="right")
plt.subplot(1, 2, 2)
plt.title(label="Sales Data2 ",loc="center")
l1 = ax1.plot(x1, y1, color = 'cyan',linestyle = '-.', marker = 'h',
        markerfacecolor = 'magenta', mec='k',markersize = 10)
l2 = ax2.plot(x1, y2, color = 'yellow',linestyle = '--', marker = 'D',
        markerfacecolor = 'green', mec='r', markersize = 10)
ax1.set_ylabel('Days of week', fontsize=25)
ax2.set_ylabel('Days of week', fontsize=25)
ax1.set_xlabel('Sale of Drinks', fontsize=25)
ax2.set_xlabel('Sale of Food', fontsize=25)
plt.subplots_adjust(right=0.9)
plt.show()
```

Output

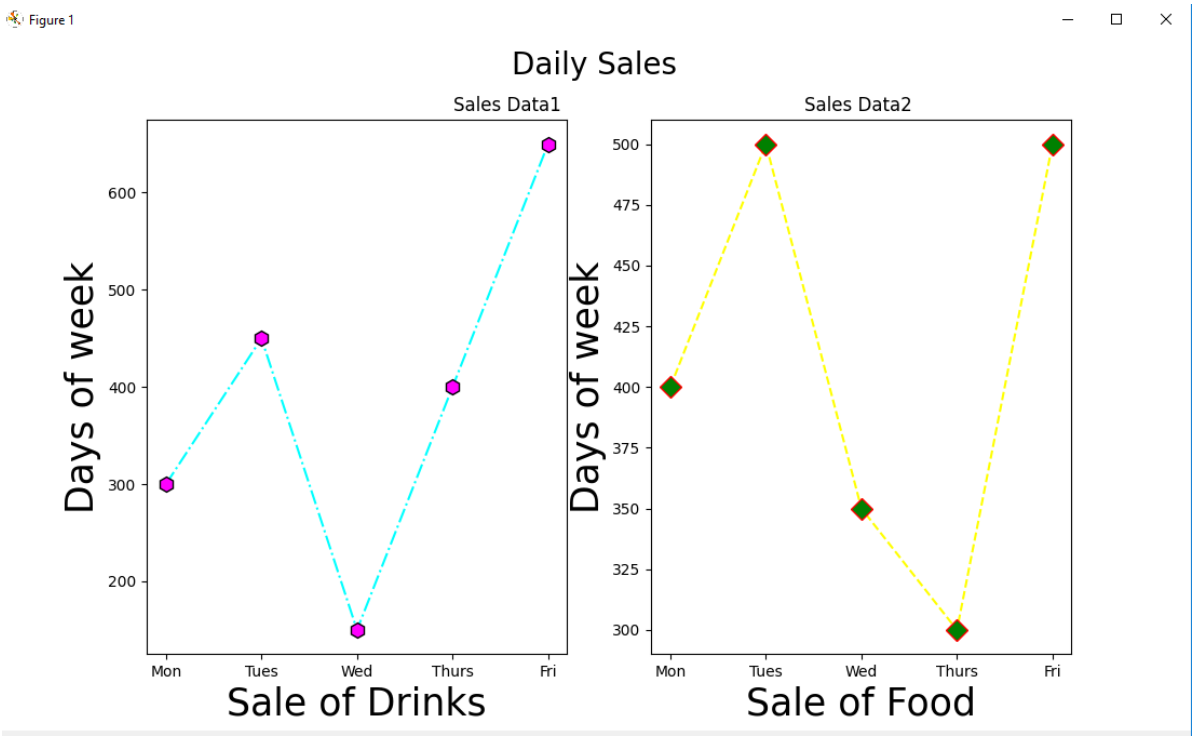


Figure 1
Daily Sales
Sales Data1
Days of week
600
500
400
300
200
Mon
Tues
Wed
Thurs
Fri
Sale of Drinks
Sales Data2
Days of week
500
475
450
425
400
375
350
325
300
Mon
Tues
Wed
Thurs
Fri
Sale of Food

9. Create scatter plot for the below data:(use Scatter function)

Product	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
Affordable Segment	173	153	195	147	120	144	148	109	174	130	172	131
Luxury Segment	189	189	105	112	173	109	151	197	174	145	177	161
Super Luxury Segment	185	185	126	134	196	153	112	133	200	145	167	110

Create scatter plot for each Segment with following properties within one graph

- **X Label- Months of Year with font size 18**
- **Y-Label- Sales of Segments**
- **Title –Sales Data**
- **Color for Affordable segment- pink**
- **Color for Luxury Segment- Yellow**
- **Color for Super luxury segment-blue**

Program

```
import matplotlib.pyplot as plt
import numpy as np
x = np.array(['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec'])
y1 = np.array([173, 153, 195, 147, 120, 144, 148, 109, 174, 130, 172, 131])
y2 = np.array([189, 189, 105, 112, 173, 109, 151, 197, 174, 145, 177, 161])
y3 = np.array([185, 185, 126, 134, 196, 153, 112, 133, 200, 145, 167, 110])
plt.scatter(x, y1, color = 'hotpink')
plt.scatter(x, y2, color = 'yellow')
plt.scatter(x,y3, color = 'blue')
plt.title(label="Sales Data ", loc="center")
plt.ylabel("Sales of Segments")
plt.xlabel("Months of Year with font size 18")
plt.show()
```

Output

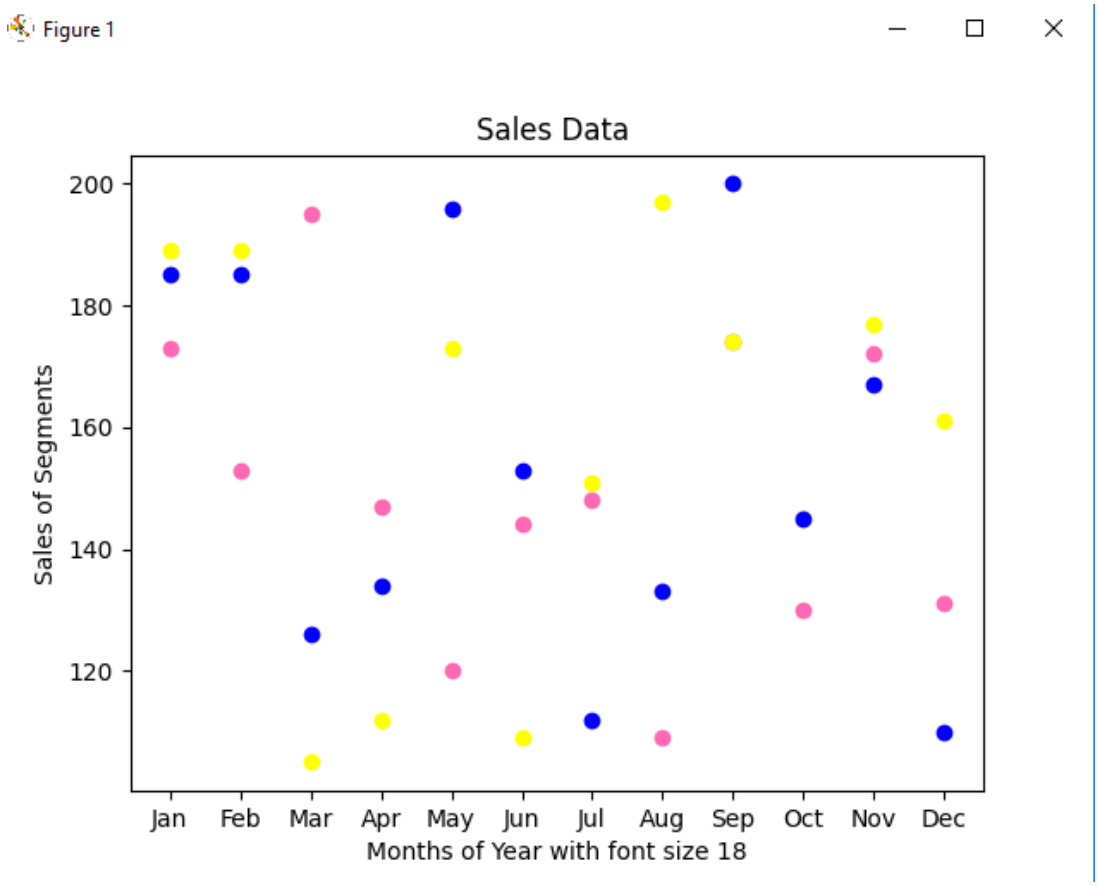


Figure 1
Sales Data
Sales of Segments
200
180
160
140
120
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec
Months of Year with font size 18

10. Program to create bar chart for given data regarding 'Primary mode of transport'

100 students were asked what their primary mode of transport for getting to school was. The results of this survey are recorded in the table below. Construct a bar graph representing this information.

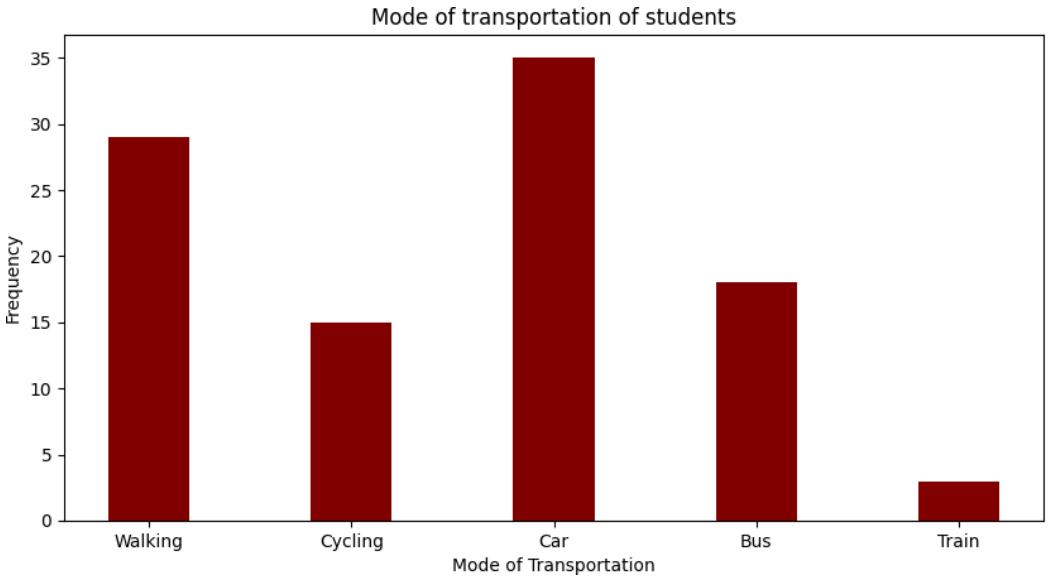
Create a bar graph with

- **X axis -mode of Transport and Y axis 'frequency'**
- **Provide appropriate labels and title**
- **Width .1, color green**

Program

```
import numpy as np

import matplotlib.pyplot as plt

data = {'Walking': 29, 'Cycling': 15, 'Car': 35, 'Bus': 18, 'Train':3}

courses = list(data.keys())

values = list(data.values())

fig = plt.figure(figsize=(10, 5))

plt.bar(courses, values, color='maroon', width=0.4)

plt.xlabel("Mode of Transportation")

plt.ylabel("Frequency")

plt.title("Mode of transportation of students")

plt.show()
```

Output


Mode of transportation of students
Frequency
0
5
10
15
20
25
30
35
Walking
Cycling
Car
Bus
Train
Mode of Transportation

11. Program to create histogram with bin size of 5 for the given data regarding height of cherry trees.

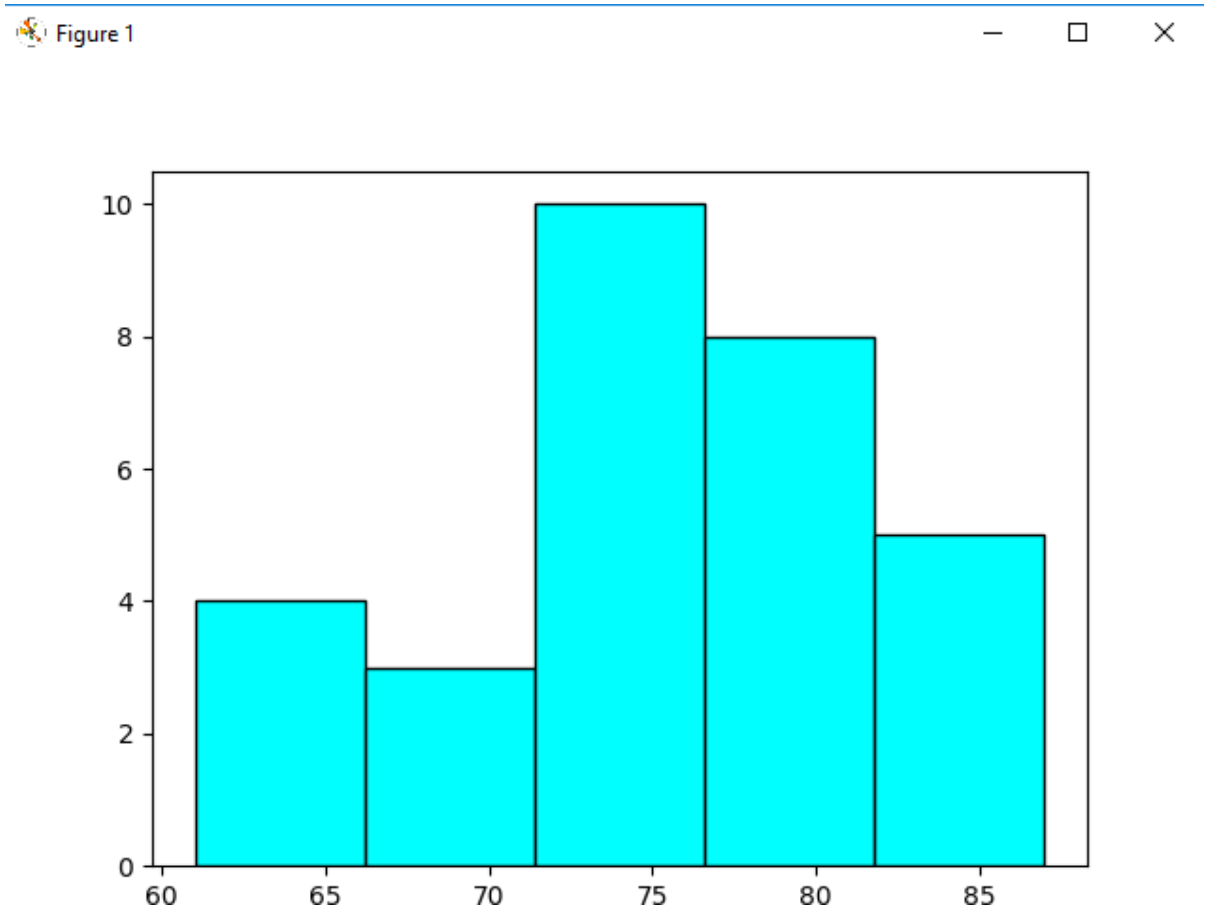
We are provided with the height of 30 cherry trees.
The height of the trees (in inches): 61, 63, 64, 66, 68, 69, 71, 71.5, 72, 72.5, 73, 73.5, 74, 74.5, 76, 76.2, 76.5, 77, 77.5, 78, 78.5, 79, 79.2, 80, 81, 82, 83, 84, 85, 87.

Create a histogram with a bin size of 5

Program

```
import matplotlib.pyplot as plt

height = [61, 63, 64, 66, 68, 69, 71, 71.5, 72, 72.5, 73, 73.5, 74, 74.5, 76, 76.2, 76.5, 77, 77.5, 78, 78.5, 79, 79.2, 80, 81, 82, 83, 84, 85, 87]

plt.hist(height,color='cyan', edgecolor="black", bins=5)

plt.show()
```

Output



12. Write a program to implement KNN algorithm using iris data Set. Use different values for K and different values for text and training data.

Program

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import sklearn as sl
dataset=pd.read_csv("iris.csv")
X=dataset.iloc[:, :-1].values
y=dataset.iloc[:,4].values

from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.20)
print(X)
print(y)
from sklearn.neighbors import KNeighborsClassifier
classifier=KNeighborsClassifier(n_neighbors=3)
classifier.fit(X_train, y_train)
y_pred=classifier.predict(X_test)
from sklearn.metrics import classification_report,confusion_matrix
print(classification_report(y_test,y_pred))
```

Output

```
 [7.7 3.  6.1 2.3]
 [6.3 3.4 5.6 2.4]
 [6.4 3.1 5.5 1.8]
 [6.  3.  4.8 1.8]
 [6.9 3.1 5.4 2.1]
 [6.7 3.1 5.6 2.4]
 [6.9 3.1 5.1 2.3]
 [5.8 2.7 5.1 1.9]
 [6.8 3.2 5.9 2.3]
 [6.7 3.3 5.7 2.5]
 [6.7 3.  5.2 2.3]
 [6.3 2.5 5.  1.9]
 [6.5 3.  5.2 2. ]
 [6.2 3.4 5.4 2.3]
 [5.9 3.  5.1 1.8]]
['Setosa' 'Setosa' 'Setosa' 'Setosa' 'Setosa' 'Setosa' 'Setosa' 'Setosa'
 'Setosa' 'Setosa' 'Setosa' 'Setosa' 'Setosa' 'Setosa' 'Setosa' 'Setosa'
 'Setosa' 'Setosa' 'Setosa' 'Setosa' 'Setosa' 'Setosa' 'Setosa' 'Setosa'
 'Setosa' 'Setosa' 'Setosa' 'Setosa' 'Setosa' 'Setosa' 'Setosa' 'Setosa'
 'Setosa' 'Setosa' 'Setosa' 'Setosa' 'Setosa' 'Setosa' 'Setosa' 'Setosa'
 'Setosa' 'Setosa' 'Setosa' 'Setosa' 'Setosa' 'Setosa' 'Setosa' 'Setosa'
 'Setosa' 'Setosa' 'Versicolor' 'Versicolor' 'Versicolor' 'Versicolor'
 'Versicolor' 'Versicolor' 'Versicolor' 'Versicolor' 'Versicolor'
 'Versicolor' 'Versicolor' 'Versicolor' 'Versicolor' 'Versicolor'
 'Versicolor' 'Versicolor' 'Versicolor' 'Versicolor' 'Versicolor'
 'Versicolor' 'Versicolor' 'Versicolor' 'Versicolor' 'Versicolor'
 'Versicolor' 'Versicolor' 'Versicolor' 'Versicolor' 'Versicolor'
 'Versicolor' 'Versicolor' 'Versicolor' 'Versicolor' 'Versicolor'
 'Versicolor' 'Versicolor' 'Versicolor' 'Versicolor' 'Versicolor'
 'Versicolor' 'Versicolor' 'Versicolor' 'Versicolor' 'Versicolor'
 'Versicolor' 'Virginica' 'Virginica' 'Virginica' 'Virginica' 'Virginica'
 'Virginica' 'Virginica' 'Virginica' 'Virginica' 'Virginica' 'Virginica'
 'Virginica' 'Virginica' 'Virginica' 'Virginica' 'Virginica' 'Virginica'
 'Virginica' 'Virginica' 'Virginica' 'Virginica' 'Virginica' 'Virginica'
 'Virginica' 'Virginica' 'Virginica' 'Virginica' 'Virginica' 'Virginica'
 'Virginica' 'Virginica' 'Virginica' 'Virginica' 'Virginica' 'Virginica'
 'Virginica' 'Virginica' 'Virginica' 'Virginica' 'Virginica' 'Virginica'
 'Virginica' 'Virginica' 'Virginica' 'Virginica' 'Virginica' 'Virginica'
 'Virginica' 'Virginica' 'Virginica']
              precision    recall  f1-score   support

      Setosa       1.00      1.00      1.00         8
  Versicolor       0.77      1.00      0.87        10
   Virginica       1.00      0.75      0.86        12

    accuracy                           0.90        30
   macro avg       0.92      0.92      0.91        30
weighted avg       0.92      0.90      0.90        30


Process finished with exit code 0
```

13. Write a program to implement naive bayes classification using different naive Bayes classification algorithms.

i) Using iris data set, implement naive bayes classification for different naive Bayes classification algorithms.((i) gaussian (ii) bernoulli

ii) Find out the accuracy level w.r.t to each algorithm

iii) Display the no:of mislabeled classification from test data set

iv) List out the class labels of the mismatching record

Gaussian

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

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
data=pd.read_csv("iris.csv")
x=data.iloc[:,:-1].values
y=data.iloc[:,4].values
data.head(5)
X_train,X_test,y_train,y_test=train_test_split(x,y,test_size=0.2)
sc=StandardScaler()
X_train=sc.fit_transform(X_train)
X_test=sc.transform(X_test)
classifier=GaussianNB()
classifier.fit(X_train,y_train)
y_pred=classifier.predict(X_test)
print(y_pred)
cm=confusion_matrix(y_test,y_pred)
print("Accuracy :",accuracy_score(y_test,y_pred))
print("array",cm)
df=pd.DataFrame({'Real Values':y_test,'predicted values':y_pred})
print(df)
```

Output

```
['Versicolor' 'Versicolor' 'Setosa' 'Setosa' 'Setosa' 'Versicolor'
 'Setosa' 'Versicolor' 'Versicolor' 'Versicolor' 'Virginica' 'Setosa'
 'Virginica' 'Setosa' 'Setosa' 'Setosa' 'Virginica' 'Versicolor'
 'Virginica' 'Virginica' 'Versicolor' 'Setosa' 'Virginica' 'Virginica'
 'Virginica' 'Setosa' 'Versicolor' 'Setosa' 'Virginica' 'Setosa']
Accuracy : 0.9333333333333333
array [[12  0  0]
 [ 0  8  1]
 [ 0  1  8]]
   Real Values predicted values
0   Versicolor       Versicolor
1   Versicolor       Versicolor
2       Setosa           Setosa
3       Setosa           Setosa
4       Setosa           Setosa
5   Versicolor       Versicolor
6       Setosa           Setosa
7   Versicolor       Versicolor
8    Virginica       Versicolor
9   Versicolor       Versicolor
10   Virginica        Virginica
11      Setosa           Setosa
12   Virginica        Virginica
13      Setosa           Setosa
14      Setosa           Setosa
15      Setosa           Setosa
16   Virginica        Virginica
```

```
17  Versicolor       Versicolor
18   Virginica        Virginica
19   Virginica        Virginica
20  Versicolor       Versicolor
21      Setosa           Setosa
22   Virginica        Virginica
23   Virginica        Virginica
24   Virginica        Virginica
25      Setosa           Setosa
26  Versicolor       Versicolor
27      Setosa           Setosa
28  Versicolor        Virginica
29      Setosa           Setosa

Process finished with exit code 0
```

Bernoulli

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score

dataset = pd.read_csv("iris.csv")

X = dataset.iloc[:,:4].values
y = dataset.iloc[:, 4].values

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.5)

classifier = GaussianNB()
classifier.fit(X_train, y_train)

y_pred = classifier.predict(X_test)
print(y_pred)
cm = confusion_matrix(y_test, y_pred)

print ("Accuracy : ", accuracy_score(y_test, y_pred))
print(cm)

df = pd.DataFrame({'Real Values':y_test, 'Predicted Values':y_pred})
print(df)
```

Output

```
['Versicolor' 'Setosa' 'Virginica' 'Virginica' 'Setosa' 'Versicolor'
 'Virginica' 'Versicolor' 'Versicolor' 'Setosa' 'Versicolor' 'Virginica'
 'Virginica' 'Virginica' 'Setosa' 'Versicolor' 'Setosa' 'Versicolor'
 'Virginica' 'Virginica' 'Setosa' 'Versicolor' 'Virginica' 'Setosa'
 'Virginica' 'Versicolor' 'Setosa' 'Setosa' 'Versicolor' 'Virginica'
 'Virginica' 'Setosa' 'Setosa' 'Setosa' 'Setosa' 'Setosa' 'Virginica'
 'Virginica' 'Setosa' 'Versicolor' 'Setosa' 'Virginica' 'Virginica'
 'Versicolor' 'Versicolor' 'Versicolor' 'Versicolor' 'Virginica'
 'Versicolor' 'Virginica' 'Versicolor' 'Virginica' 'Versicolor'
 'Versicolor' 'Versicolor' 'Versicolor' 'Setosa' 'Versicolor' 'Setosa'
 'Virginica' 'Versicolor' 'Setosa' 'Virginica' 'Setosa' 'Versicolor'
 'Versicolor' 'Setosa' 'Versicolor' 'Setosa' 'Virginica' 'Setosa'
 'Virginica' 'Virginica' 'Virginica' 'Setosa']
Accuracy :  0.9466666666666667
[[24  0  0]
 [ 0 23  1]
 [ 0  3 24]]
   Real Values Predicted Values
0   Versicolor       Versicolor
1       Setosa           Setosa
2    Virginica        Virginica
3    Virginica        Virginica
4       Setosa           Setosa
..         ...              ...
70      Setosa           Setosa
71   Virginica        Virginica
72   Virginica        Virginica
73   Virginica        Virginica
74      Setosa           Setosa

[75 rows x 2 columns]

Process finished with exit code 0
```

14. Write a program to implement decision tree algorithm using the given data set

Program

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import tree, metrics
from sklearn.model_selection import train_test_split
data = pd.read_csv('car.csv', names=['buying', 'maint', 'doors', 'persons', 'lug_boot', 'safety', 'class'])
print(data.head())

data.info()

data['class'],class_names = pd.factorize(data['class'])
print(class_names)
print(data['class'].unique())

data['buying'],_ = pd.factorize(data['buying'])
data['maint'],_ = pd.factorize(data['maint'])
data['doors'],_ = pd.factorize(data['doors'])
data['persons'],_ = pd.factorize(data['persons'])
data['lug_boot'],_ = pd.factorize(data['lug_boot'])
data['safety'],_ = pd.factorize(data['safety'])
print(data.head())

X = data.iloc[:,:-1]
y = data.iloc[:,-1]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)

dtree = tree.DecisionTreeClassifier(criterion='entropy', max_depth=3, random_state=0)
dtree.fit(X_train, y_train)


y_pred = dtree.predict(X_test)
count_misclassified = (y_test != y_pred).sum()
print('Misclassified samples: {}'.format(count_misclassified))
accuracy = metrics.accuracy_score(y_test, y_pred)
print('Accuracy: {:.2f}'.format(accuracy))
```

Output

```
  buying  maint doors persons lug_boot safety  class
0  vhigh  vhigh     2       2    small    low  unacc
1  vhigh  vhigh     2       2    small    med  unacc
2  vhigh  vhigh     2       2    small   high  unacc
3  vhigh  vhigh     2       2      med    low  unacc
4  vhigh  vhigh     2       2      med    med  unacc
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1728 entries, 0 to 1727
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   buying    1728 non-null   object
 1   maint     1728 non-null   object
 2   doors     1728 non-null   object
 3   persons   1728 non-null   object
 4   lug_boot  1728 non-null   object
 5   safety    1728 non-null   object
 6   class     1728 non-null   object
dtypes: object(7)
memory usage: 94.6+ KB
Index(['unacc', 'acc', 'vgood', 'good'], dtype='object')
[0 1 2 3]
   buying  maint  doors  persons  lug_boot  safety  class
0       0      0      0        0         0       0      0
1       0      0      0        0         0       1      0
2       0      0      0        0         0       2      0
3       0      0      0        0         1       0      0
4       0      0      0        0         1       1      0
Misclassified samples: 96
Accuracy: 0.82

Process finished with exit code 0
```

15. Write a program to demonstrate Simple Linear Regressionusinggiven 'student_scores.csv' data set

Program

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
student=pd.read_csv('student_scores.csv')
student.head()
X=student.iloc[:,:-1]
Y=student.iloc[:,1]
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2)
print(X_train)

from sklearn.linear_model import LinearRegression
regressor=LinearRegression()
regressor.fit(X_train,Y_train)
print(regressor.intercept_)
print(regressor.coef_)

Y_pred=regressor.predict(X_test)
for(i,j) in zip (Y_test,Y_pred):
if i!=j:
print("Actual value:",i,"predicted value:",j)
print("Number of misplaced points from the test
dataset:",(Y_test!=Y_pred).sum())
from sklearn import metrics
print("Mean Absolute
error:",metrics.mean_absolute_error(Y_test,Y_pred))
print("Mean Squared error:",metrics.mean_squared_error(Y_test,Y_pred))print("Root Mean Squared
error:",np.sqrt(metrics.mean_squared_error(Y_test,Y_pred)))
```

Output

```
    Hours
21    4.8
10    7.7
1     5.1
6     9.2
17    1.9
14    1.1
12    4.5
4     3.5
19    7.4
11    5.9
7     5.5
8     8.3
3     8.5
15    8.9
5     1.5
23    6.9
0     2.5
18    6.1
2     3.2
24    7.8
1.9161461851768493
[9.85201338]
Actual value: 25 predicted value: 28.516582322986917
Number of misplaced points from the test dataset: 5
Actual value: 42 predicted value: 34.42779035361137
Number of misplaced points from the test dataset: 5
```

```
Actual value: 30 predicted value: 28.516582322986917
Number of misplaced points from the test dataset: 5
Actual value: 30 predicted value: 26.546179646112094
Number of misplaced points from the test dataset: 5
Actual value: 35 predicted value: 39.35379704579842
Number of misplaced points from the test dataset: 5
Mean Absolute error: 4.075965409214992
Mean Squared error: 20.557932384122683
Root Mean Squared error: 4.5340856172025115

Process finished with exit code 0
```

16. Write a program to implement Multiple Linear Regression using appropriate data set

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

advertising=pd.read_csv('Company_data.csv')
advertising.head()
advertising.describe()
advertising.info()
x=advertising.iloc[:,:1]
print(x)
y=advertising.iloc[:,-1]
print(y)

from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)
print(x_train)

from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(x_train,y_train)
print(regressor.intercept_)
print(regressor.coef_)
y_pred=regressor.predict(x_test)
for(i,j) in zip(y_test,y_pred):
   if i!=j:
      print("Actual value:",i,"predicted value:",j)
      print("Number of mislabeled points from test data set",(y_test!=y_pred).sum())
```

Output

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   TV         200 non-null    float64
 1   Radio      200 non-null    float64
 2   Newspaper  200 non-null    float64
 3   Sales      200 non-null    float64
dtypes: float64(4)
memory usage: 6.4 KB
        TV
0    230.1
1     44.5
2     17.2
3    151.5
4    180.8
..     ...
195   38.2
196   94.2
197  177.0
198  283.6
199  232.1

[200 rows x 1 columns]
0      22.1
1      10.4
2      12.0
3      16.5
4      17.9
       ...
195     7.6
196    14.0
```

```
197    14.8
198    25.5
199    18.4
Name: Sales, Length: 200, dtype: float64
        TV
124  229.5
33   265.6
156   93.9
174  222.4
117   76.4
..     ...
112  175.7
199  232.1
125   87.2
173  168.4
70   199.1

[160 rows x 1 columns]
6.946305516811114
[0.05651502]
Actual value: 10.7 predicted value: 12.62041341670926
Number of mislabeled points from test data set 40
Actual value: 17.6 predicted value: 21.289817319541687
Number of mislabeled points from test data set 40
Actual value: 11.3 predicted value: 10.857144826302665
Number of mislabeled points from test data set 40
Actual value: 10.3 predicted value: 14.830150656609833
Number of mislabeled points from test data set 40
Actual value: 3.2 predicted value: 7.178017094396596
Number of mislabeled points from test data set 40
Actual value: 13.2 predicted value: 14.64365109416298
Number of mislabeled points from test data set 40
Actual value: 18.0 predicted value: 19.28918564965728
```

```
Number of mislabeled points from test data set 40
Actual value: 7.6 predicted value: 8.053999887707565
Number of mislabeled points from test data set 40
Actual value: 17.2 predicted value: 19.294837151549608
Number of mislabeled points from test data set 40
Actual value: 16.9 predicted value: 20.19907745432222
Number of mislabeled points from test data set 40
Actual value: 21.2 predicted value: 17.265947972203556
Number of mislabeled points from test data set 40
Actual value: 6.7 predicted value: 8.003136370676605
Number of mislabeled points from test data set 40
Actual value: 18.4 predicted value: 19.707396789689614
Number of mislabeled points from test data set 40
Actual value: 6.9 predicted value: 8.500468537201542
Number of mislabeled points from test data set 40
Actual value: 16.6 predicted value: 18.390596848776994
Number of mislabeled points from test data set 40
Actual value: 10.8 predicted value: 9.178648764281002
Number of mislabeled points from test data set 40
Actual value: 17.5 predicted value: 20.362971009199757
Number of mislabeled points from test data set 40
Actual value: 11.9 predicted value: 12.354792827769804
Number of mislabeled points from test data set 40
Actual value: 18.4 predicted value: 18.85402000394796
Number of mislabeled points from test data set 40
Actual value: 8.0 predicted value: 7.957924355537974
Number of mislabeled points from test data set 40
Actual value: 6.6 predicted value: 8.042696883922908
Number of mislabeled points from test data set 40
Actual value: 12.0 predicted value: 11.25840146065801
Number of mislabeled points from test data set 40
Actual value: 8.4 predicted value: 9.772056462975529
Number of mislabeled points from test data set 40
Actual value: 13.3 predicted value: 11.789642638536922
```

```
Number of mislabeled points from test data set 40
Actual value: 17.3 predicted value: 20.49860705461565
Number of mislabeled points from test data set 40
Actual value: 10.5 predicted value: 10.936265852795268
Number of mislabeled points from test data set 40
Actual value: 12.0 predicted value: 7.918363842291672
Number of mislabeled points from test data set 40
Actual value: 16.7 predicted value: 13.15165459458817
Number of mislabeled points from test data set 40
Actual value: 11.8 predicted value: 11.264052962550341
Number of mislabeled points from test data set 40
Actual value: 4.8 predicted value: 7.432334679551393
Number of mislabeled points from test data set 40
Actual value: 18.9 predicted value: 20.362971009199757
Number of mislabeled points from test data set 40
Actual value: 13.7 predicted value: 12.456519861831723
Number of mislabeled points from test data set 40
Actual value: 16.4 predicted value: 16.695146281078344
Number of mislabeled points from test data set 40
Actual value: 20.0 predicted value: 23.01352539670198
Number of mislabeled points from test data set 40
Actual value: 16.0 predicted value: 14.35542449765421
Number of mislabeled points from test data set 40
Actual value: 17.3 predicted value: 17.74632563305151
Number of mislabeled points from test data set 40
Actual value: 12.9 predicted value: 11.93658168773747
Number of mislabeled points from test data set 40
Actual value: 10.4 predicted value: 12.857776496187071
Number of mislabeled points from test data set 40
Actual value: 18.0 predicted value: 16.186511110768752
Number of mislabeled points from test data set 40
Actual value: 11.6 predicted value: 9.675980930805938
Number of mislabeled points from test data set 40
```

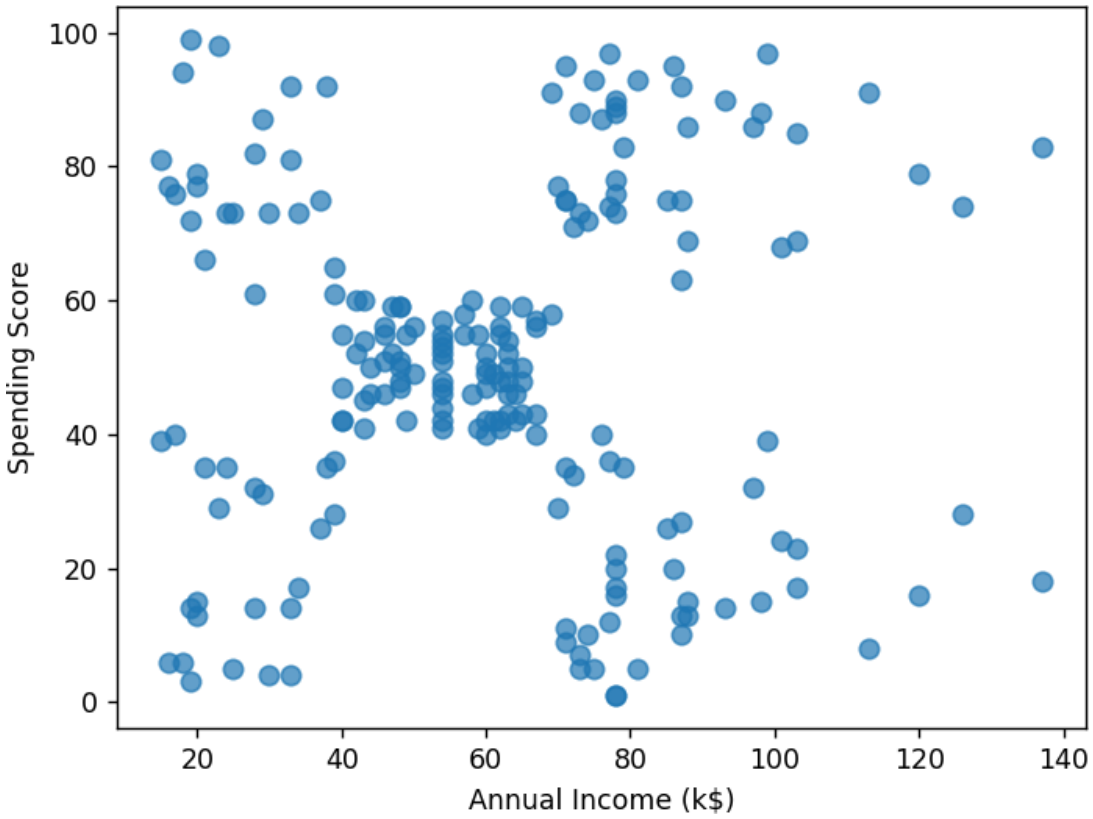
17. Write a program to implement K –Means Clustering Algorithm with k=6. Create a scatter plot to visualize the same.

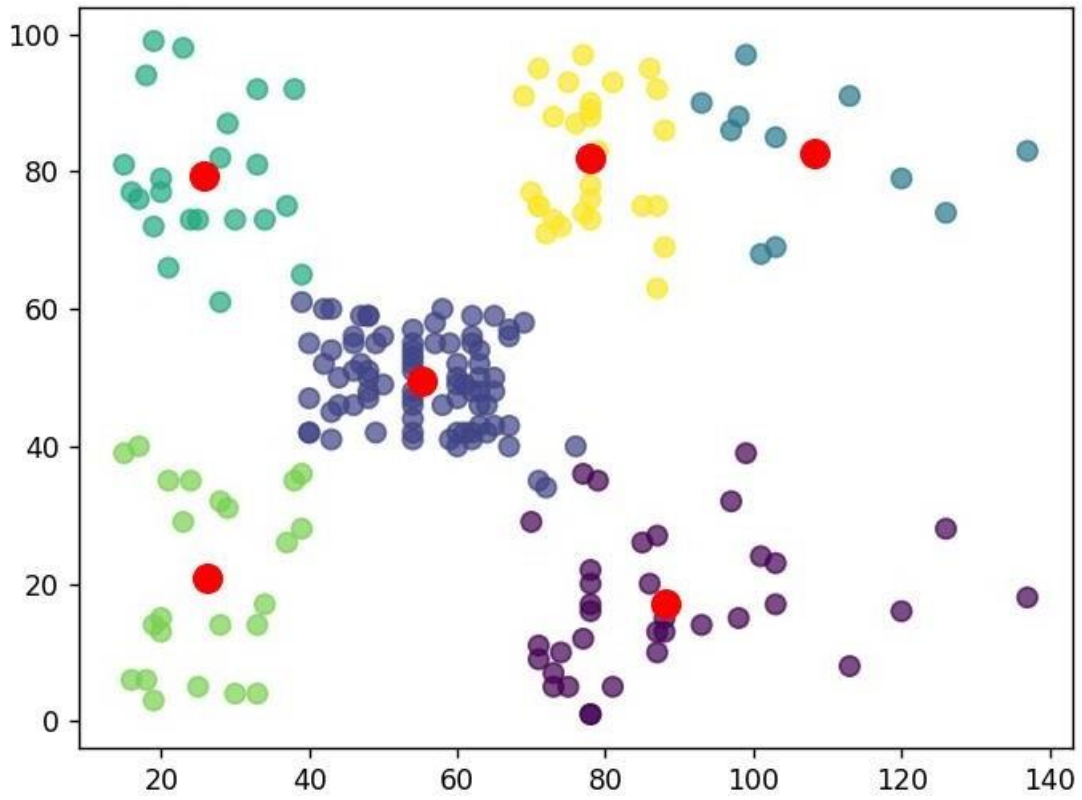
Given dataset contains 200 records and five columns, two of which describe the customer's annual income and spending score. The latter is a value from 0 to 100. The higher the number, the more this customer has spent with the company in the past:

Program

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

customers = pd.read_csv('customer_data.csv')
customers.head()
points = customers.iloc[:, 3:5].values
x = points[:, 0]
y = points[:, 1]
plt.scatter(x, y, s=50, alpha=0.7)
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score')
plt.show()
kmeans = KMeans(n_clusters=6, random_state=0)
kmeans.fit(points)
predicted_cluster_indexes = kmeans.predict(points)
plt.scatter(x, y, c=predicted_cluster_indexes, s=50, alpha=0.7, cmap='viridis')
centers = kmeans.cluster_centers_
plt.scatter(centers[:, 0], centers[:, 1], c='red', s=100)
plt.show()
```

Output


Spending Score
Annual Income (k$)
0
20
40
60
80
100
20
40
60
80
100
120
140


100
80
60
40
20
0
20
40
60
80
100
120
140

18. For given text:
1) perform word and sentence tokenization.
2) Remove the stop words from the given text
3) Perform Part of Speech tagging
4) create n-grams for different values of n=2,4.

Program

```
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import sent_tokenize, word_tokenize
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
nltk.download('stopwords')

text1 = "The data set given satisfies the requirement for model generation. This is used in Data Science Lab"
print("sentence tokenization:")
for i in sent_tokenize(text1):
    print(i)
print("word tokenization:")
for i in word_tokenize(text1):
    print(i)
text = word_tokenize(text1)
print("parts of  Speech:")
for i in nltk.pos_tag(text):
    print(i)
print("after removing stop words")
text = [word for word in text if word not in stopwords.words('english')]
print(text)
# 2 grams
print("2 grams are:")
temp = zip(*[text[i:] for i in range(0, 2)])
ans = [' '.join(ngram) for ngram in temp]
print(ans)
# 4 grams
print("4 grams are:")
temp = zip(*[text[i:] for i in range(0, 4)])
ans = [' '.join(ngram) for ngram in temp]
print(ans)
```

Output

```
C:\Users\donat\PycharmProjects\DS\venv\Scripts\python.exe C:/Users/donat/PycharmProjects/DS/re18.py
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\donat\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping tokenizers\punkt.zip.
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     C:\Users\donat\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping taggers\averaged_perceptron_tagger.zip.
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\donat\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\stopwords.zip.
sentence tokenization:
The data set given satisfies the requirement for model generation.
This is used in Data Science Lab
word tokenization:
The
data
set
given
satisfies
the
requirement
for
model
generation
.
This
is
used
in
Data
Science
Lab
parts of Speech:
('The', 'DT')
```

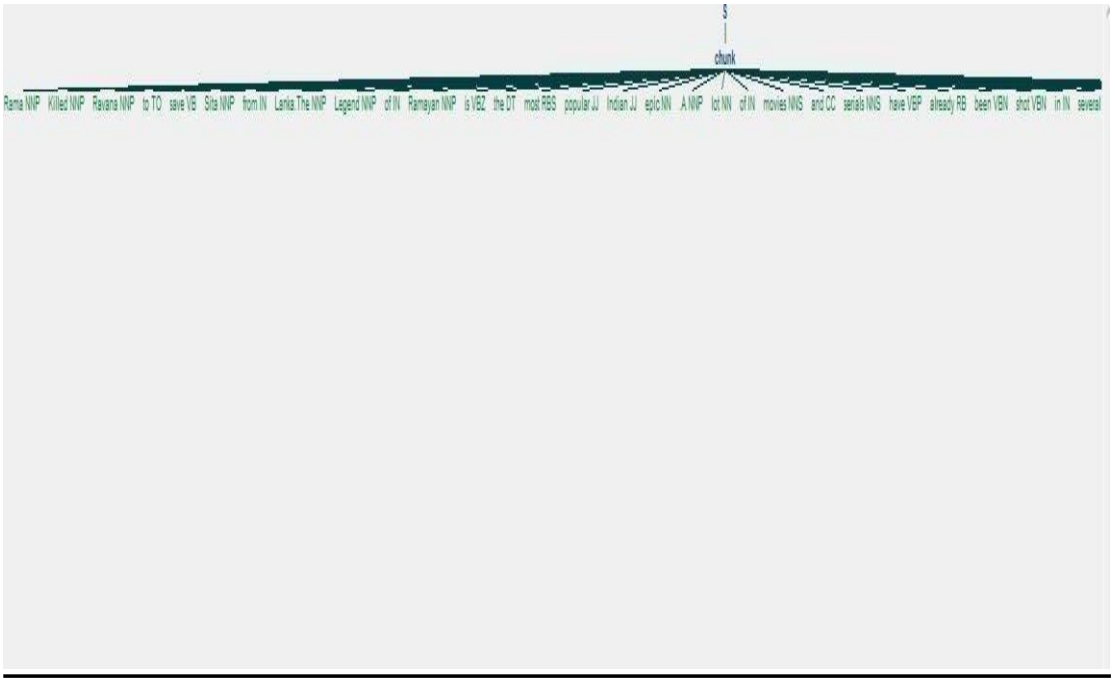
```
('data', 'NN')
('set', 'NN')
('given', 'VBN')
('satisfies', 'VBZ')
('the', 'DT')
('requirement', 'NN')
('for', 'IN')
('model', 'NN')
('generation', 'NN')
('.', '.')
('This', 'DT')
('is', 'VBZ')
('used', 'VBN')
('in', 'IN')
('Data', 'NNP')
('Science', 'NNP')
('Lab', 'NNP')
after removing stop words
['The', 'data', 'set', 'given', 'satisfies', 'requirement', 'model', 'generation', '.', 'This', 'used', 'Data', 'Science', 'Lab']
2 grams are:
['The data', 'data set', 'set given', 'given satisfies', 'satisfies requirement', 'requirement model', 'model generation', 'generation .', '. This', 'This used', 'used Data', 'Data
4 grams are:
['The data set given', 'data set given satisfies', 'set given satisfies requirement', 'given satisfies requirement model', 'satisfies requirement model generation', 'requirement mod

Process finished with exit code 0
```

19. Write a program to perform chunking on given text by creating a chunk containing every word.

Program

```
import nltk
from nltk.corpus import stopwords
sample_text="Rama Killed Ravana to save Sita from Lanka.The Legend of Ramayan is the most popular Indian epic.A lot of Movies and serials have already been shot in several languages here in india based on the ramayan."
tokenized=nltk.sent_tokenize(sample_text)
for i in tokenized:
words=nltk.word_tokenize(i)
tagged_words=nltk.pos_tag(words)
#following statement will chunk every word in the sentence
chunkGram=r"""chunk: {<.*>+ }"""
chunkParser=nltk.RegexpParser(chunkGram)
chunked=chunkParser.parse(tagged_words)
print(chunked)
chunked.draw()
```

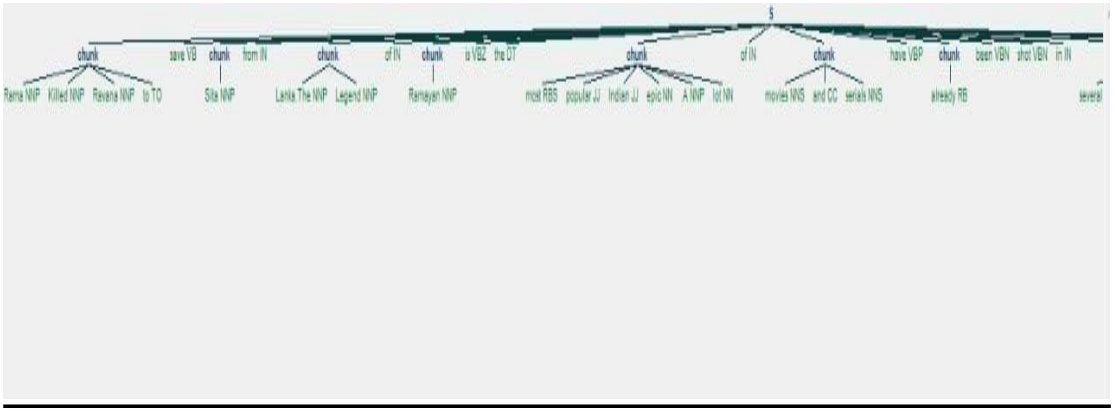
Output


S
chunk
Rama NNP Killed NNP Ravana NNP to TO save VB Sita NNP from IN Lanka The NNP Legend NNP of IN Ramayan NNP is VBZ the DT most RBS popular JJ Indian JJ epic NN A NNP lot NN of IN movies NNS and CC serials NNS have VBP already RB been VBN shot VBN in IN several

20. Write a program to create chunks using words in the given sentence -except Verbs(VB), determiner(DT) and propositions(IN)

```
import nltk
from nltk.corpus import stopwords
sample_text="Rama Killed Ravana to save Sita from Lanka.The Legend of Ramayan is the most popular Indian epic.A lot of Movies and serials have already been shot in several languages here in india based on the ramayan."
tokenized=nltk.sent_tokenize(sample_text)
for i in tokenized:
words=nltk.word_tokenize(i)
tagged_words=nltk.pos_tag(words)
chunkGram=r"""chunk: {<.*>+}
                }<VB.?|IN|DT|>{"""
chunkParser=nltk.RegexpParser(chunkGram)
chunked=chunkParser.parse(tagged_words)
print(chunked)
chunked.draw()
```

Output


S
chunk
Rama NNP
Killed NNP
Ravana NNP
to TO
save VB
chunk
Sita NNP
from IN
chunk
Lanka The NNP
Legend NNP
of IN
chunk
Ramayan NNP
is VBZ
the DT
chunk
most RBS
popular JJ
Indian JJ
epic NN
A NNP
lot NN
of IN
chunk
movies NNS
and CC
serials NNS
have VBP
chunk
already RB
been VBN
shot VBN
in IN
several