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:

ANAKHA THOMAS T (SJC21MCA-2004)

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 MCA 241) ***submitted*** *by* ***Anakha Thomas T*** *student of* **Third** *semester* ***MCA*** *at* ***ST. JOSEPH'S COLLEGE OF ENGINEERING AND TECHNOLOGY****,* ***PALAI*** *in partial fulfillment for the award of Master of Computer Applications is a bonafide record of the lab work carried out by her under our guidance and supervision. This record in any form has not been submitted to any other University or Institute for any purpose.*

Mrs. Liz George	**Mr. Anish Augustine**
Faculty In- Charge	**(HOD In Charge MCA)**

Submitted for the End Semester Examination held on ______________________

Examiner 1:

Examiner 2:

DECLARATION

Me **Anakha Thomas T**, do hereby declare that the *Data Science Lab Record (20 MCA 241)* is a record of work carried out under the guidance of Mrs. 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 forthe award of any Degree / Diploma.

Place: choondacherry
Date:

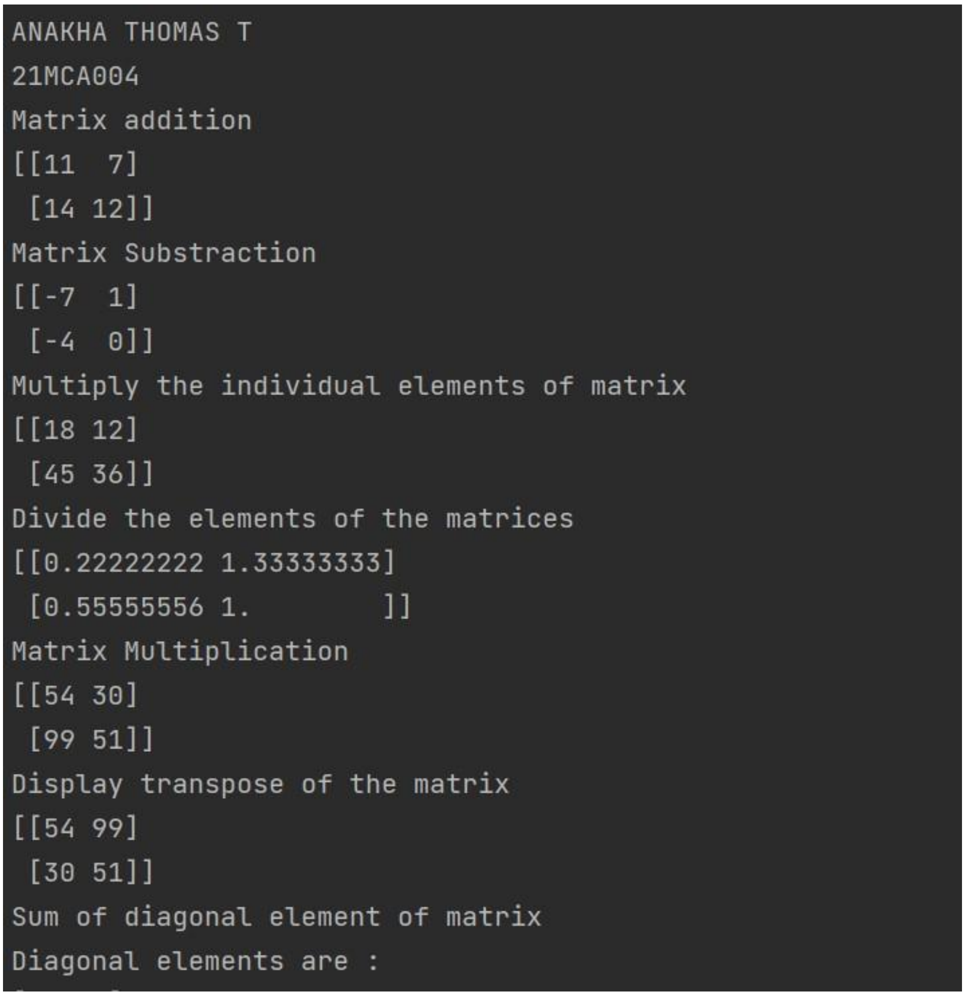
Anakha Thomas T
(SJC21MCA-2004)

1. Write a program to perform different matrix operations on a 2D Matrix

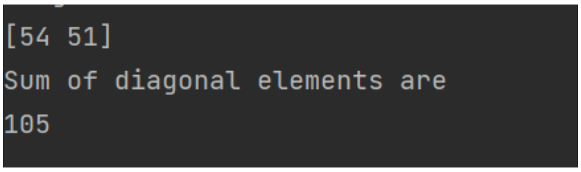
CODE:

```
print("ANAKHA THOMAS T")
print("21MCA004")
import numpy as np
A=np.array([ [2, 4],[5, 6] ])
B=np.array([ [9, 3],[9, 6] ])
print("Matrix addition ")
C=A+B
print(C)
print("Matrix Substraction ")
C=A-B
print(C)
print("Multiply the individual elements of matrix ")
C=np.multiply(A,B)
print(C)
print("Divide the elements of the matrices ")
C=np.divide(A,B)
print(C)
print("Matrix Multiplication" )
C=np.matmul(A,B)
print(C)
print("Display transpose of the matrix ")
C=np.transpose(C)
print(C)
print("Sum of diagonal element of matrix ")
C=np.diagonal(C)
print("Diagonal elements are :")
print(C)
print("Sum of diagonal elements are ")
print(sum(C))
```

OUTPUT



```
ANAKHA THOMAS T
21MCA004
Matrix addition
[[11  7]
 [14 12]]
Matrix Substraction
[[-7  1]
 [-4  0]]
Multiply the individual elements of matrix
[[18 12]
 [45 36]]
Divide the elements of the matrices
[[0.22222222 1.33333333]
 [0.55555556 1.        ]]
Matrix Multiplication
[[54 30]
 [99 51]]
Display transpose of the matrix
[[54 99]
 [30 51]]
Sum of diagonal element of matrix
Diagonal elements are :
```



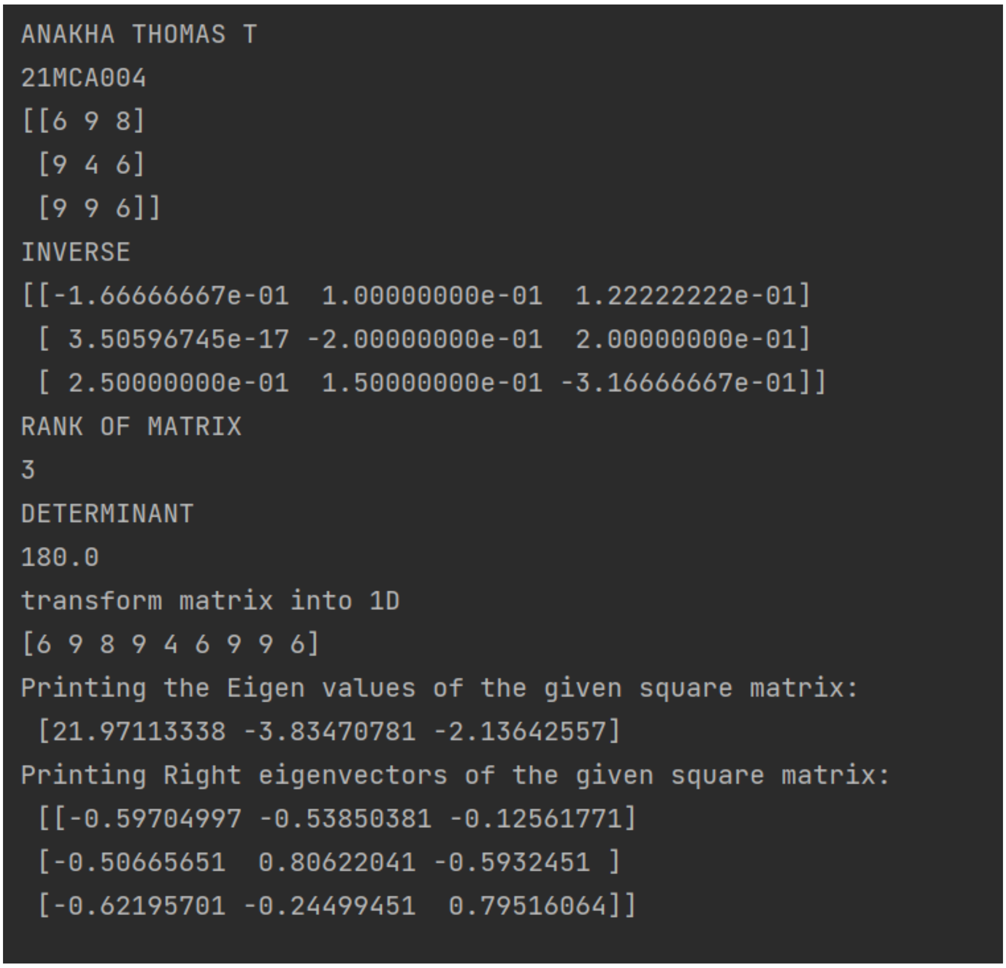
```
[54 51]
Sum of diagonal elements are
105
```

2. Write a program to find the inverse, rank, determinant, Eigen values of a given matrix. Also transform the matrix to 1D array.

CODE:

```
print("ANAKHA THOMAS T")
print("21MCA004")
import numpy as np
m = np.random.randint(10, size=(3, 3))
print(m)
print("INVERSE")
inverse=np.linalg.inv(m)
print(inverse)
print("RANK OF MATRIX")
rank = np.linalg.matrix_rank(m)
print(rank)

print("DETERMINANT")
det=np.linalg.det(m)
print(det)
print("transform matrix into 1D")
tmatrix = np.ravel(m)
print(tmatrix)
w, v = np.linalg.eig(m)
print("Printing the Eigen values of the given square matrix:\n",w)
print("Printing Right eigenvectors of the given square matrix:\n",v)
```

OUTPUT



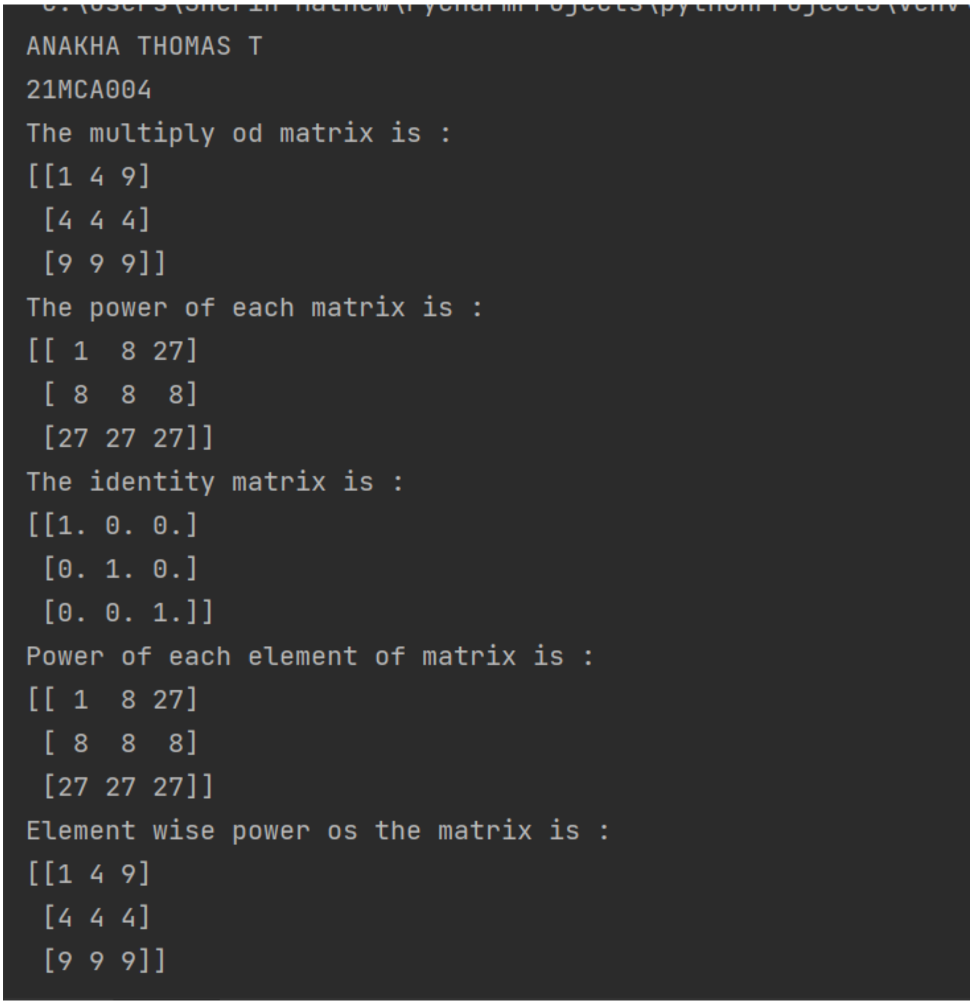
```
ANAKHA THOMAS T
21MCA004
[[6 9 8]
 [9 4 6]
 [9 9 6]]
INVERSE
[[-1.66666667e-01  1.00000000e-01  1.22222222e-01]
 [ 3.50596745e-17 -2.00000000e-01  2.00000000e-01]
 [ 2.50000000e-01  1.50000000e-01 -3.16666667e-01]]
RANK OF MATRIX
3
DETERMINANT
180.0
transform matrix into 1D
[6 9 8 9 4 6 9 9 6]
Printing the Eigen values of the given square matrix:
 [21.97113338 -3.83470781 -2.13642557]
Printing Right eigenvectors of the given square matrix:
 [[-0.59704997 -0.53850381 -0.12561771]
 [-0.50665651  0.80622041 -0.5932451 ]
 [-0.62195701 -0.24499451  0.79516064]]
```

3. Write a program to display the elements of the matrix X to different powers and identity matrix of a given matrix .Also create another matrix Y with same dimensions and display X^2+2Y.

CODE:

```
print("ANAKHA   THOMAS   T")
print("21MCA004")
import numpy as np
A = np.array([ [1, 2, 3], [2, 2, 2], [3, 3, 3] ])
#B = np.array([ [3, 2, 1], [1, 2, 3], [1, 2, 3] ])
arrA = np.multiply(A,A)
print("The multiply od matrix is :")
print(arrA)
arrB = np.power(A, 3)
print("The power of each matrix is :")
print(arrB)

arrC = np.identity(3)
print("The identity matrix is :")
print(arrC)
arrD = np.power(A,3)
print("Power of each element of matrix is : ")
print(arrD)
arrE=np.power(A,2)
print("Element wise power os the matrix is :")
print(arrE)
```

OUTPUT



```
 C:\Users\Sherin mathew\PycharmProjects\pythonProjects\venv
ANAKHA THOMAS T
21MCA004
The multiply od matrix is :
[[1 4 9]
 [4 4 4]
 [9 9 9]]
The power of each matrix is :
[[ 1  8 27]
 [ 8  8  8]
 [27 27 27]]
The identity matrix is :
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
Power of each element of matrix is :
[[ 1  8 27]
 [ 8  8  8]
 [27 27 27]]
Element wise power os the matrix is :
[[1 4 9]
 [4 4 4]
 [9 9 9]]
```

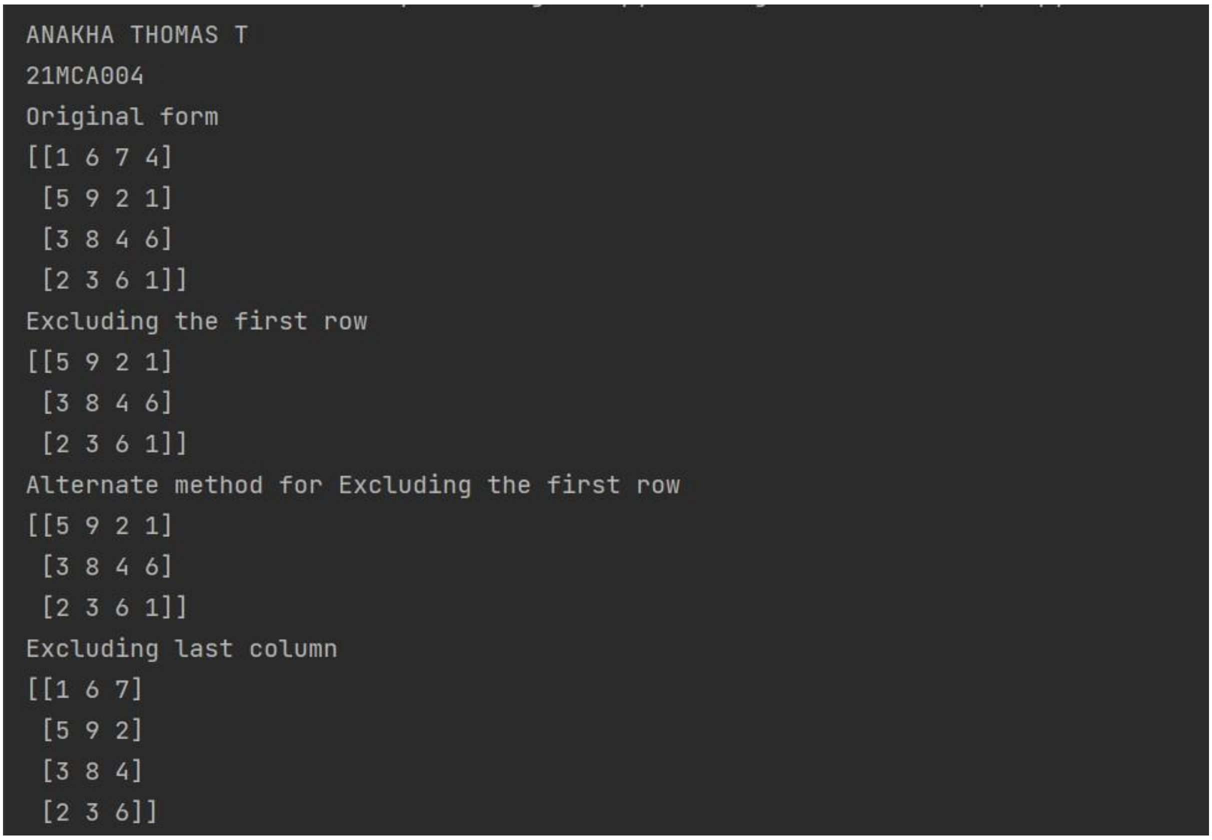
4. Write a Program to display various elements of a give 4x4 matrix specifying appropriate indices.

CODE:

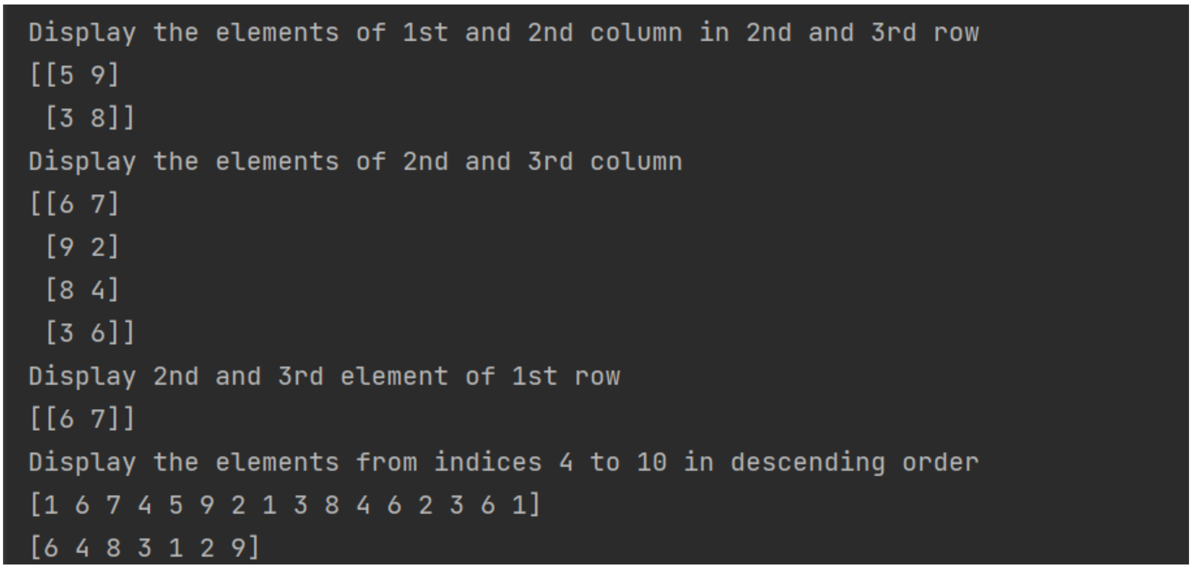
```
print("ANAKHATHOMAST")
print("21MCA004")
import numpy as np
X = np.array( [ [ 1, 6, 7, 4],
[ 5, 9, 2, 1],
[ 3, 8, 4, 6],
[ 2, 3, 6, 1] ] )
print("Original form")
print(X)
print("Excluding the first row")
print(X[1:,])
print("Alternate method for Excluding the first row")
num=np.delete(X,0,axis=0)
print(num)
print("Excluding last column")
print(X[:, :-1])
print("Display the elements of 1st and 2nd column in 2nd and 3rd row")
print(X[1:3,0:2])
print("Display the elements of 2nd and 3rd column")
print(X[:,[1,2]])
print("Display 2nd and 3rd element of 1st row")
print(X[0:1,1:3])
print("Display the elements from indices 4 to 10 in descending order")
flat_array=X.flatten()
```

```
print(flat_array)
new=sorted(flat_array[-3:-10])
index=flat_array[11:4:-1]
print(index)
```

OUTPUT



```
ANAKHA THOMAS T
21MCA004
Original form
[[1 6 7 4]
 [5 9 2 1]
 [3 8 4 6]
 [2 3 6 1]]
Excluding the first row
[[5 9 2 1]
 [3 8 4 6]
 [2 3 6 1]]
Alternate method for Excluding the first row
[[5 9 2 1]
 [3 8 4 6]
 [2 3 6 1]]
Excluding last column
[[1 6 7]
 [5 9 2]
 [3 8 4]
 [2 3 6]]
```



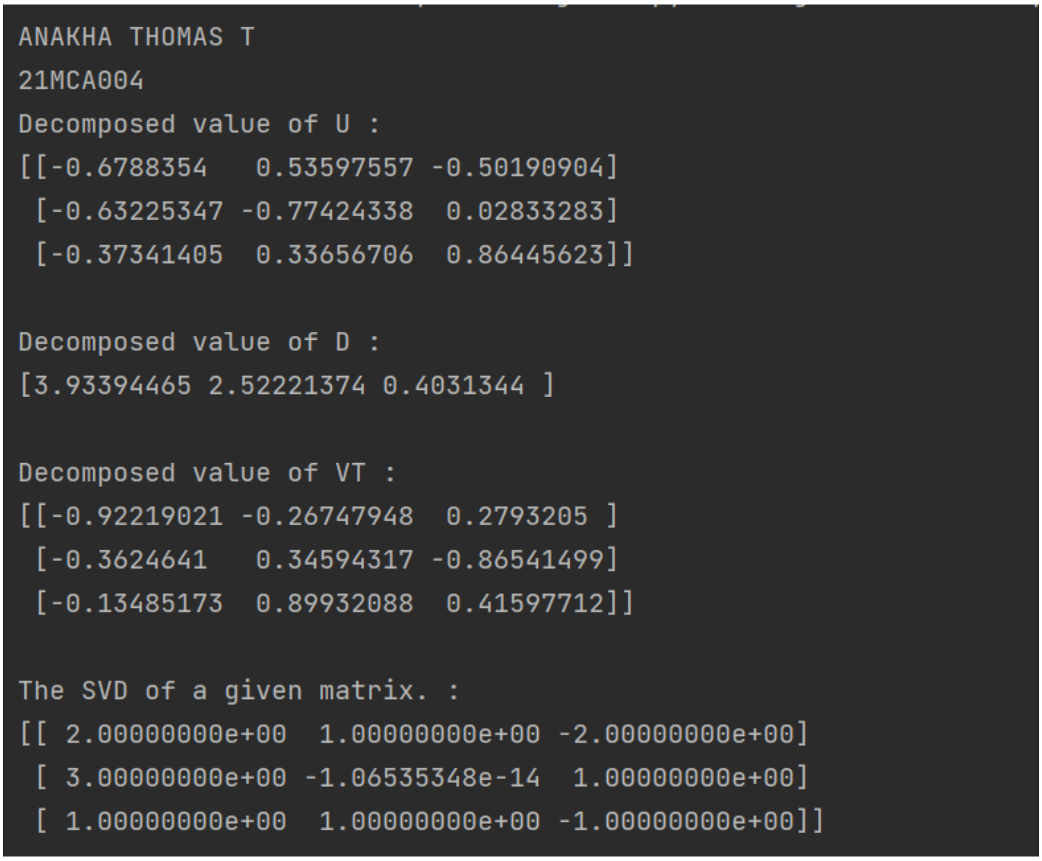
```
Display the elements of 1st and 2nd column in 2nd and 3rd row
[[5 9]
 [3 8]]
Display the elements of 2nd and 3rd column
[[6 7]
 [9 2]
 [8 4]
 [3 6]]
Display 2nd and 3rd element of 1st row
[[6 7]]
Display the elements from indices 4 to 10 in descending order
[1 6 7 4 5 9 2 1 3 8 4 6 2 3 6 1]
[6 4 8 3 1 2 9]
```

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

CODE:

```
print(“ANAKHA THOMAS T”)
print(“21MCA004”)
import numpy as np
A = np.array([[2, 1, -2],
              [3, 0, 1],
              [1, 1, -1]])
U, D, VT = np.linalg.svd(A)
print("Decomposed value of U :")
print(U)
print()
print("Decomposed value of D :")
print(D)
print()
print("Decomposed value of VT :")
print(VT)
print()
A_remake = (U @ np.diag(D) @ VT)
print("The SVD of a given matrix. :")
print(A_remake)
```

OUTPUT



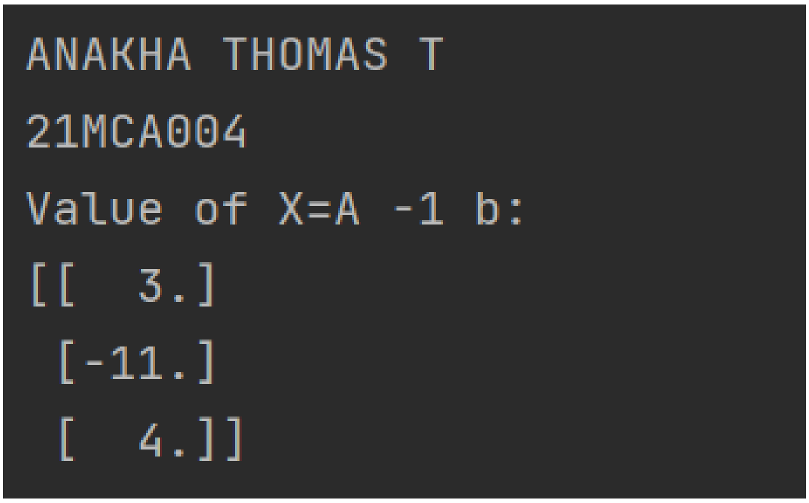
```
ANAKHA THOMAS T
21MCA004
Decomposed value of U :
[[-0.6788354   0.53597557 -0.50190904]
 [-0.63225347 -0.77424338  0.02833283]
 [-0.37341405  0.33656706  0.86445623]]

Decomposed value of D :
[3.93394465 2.52221374 0.4031344 ]

Decomposed value of VT :
[[-0.92219021 -0.26747948  0.2793205 ]
 [-0.3624641   0.34594317 -0.86541499]
 [-0.13485173  0.89932088  0.41597712]]

The SVD of a given matrix. :
[[ 2.00000000e+00  1.00000000e+00 -2.00000000e+00]
 [ 3.00000000e+00 -1.06535348e-14  1.00000000e+00]
 [ 1.00000000e+00  1.00000000e+00 -1.00000000e+00]]
```

6. Write a program to Solve systems of equations with numpy

CODE:

```
print(“ANAKHA THOMAS T”)
print(“21MCA004”)
import numpy as np
A = np.array([[2, 1, -2],
            [3, 0, 1],
            [1, 1, -1]])
b = np.array([[-3],
            [5],
            [-2]])
a=np.linalg.inv(A)
x= np.linalg.solve(a, b)
print("Value of X=A -1 b: ")
print(x)
```

OUTPUT



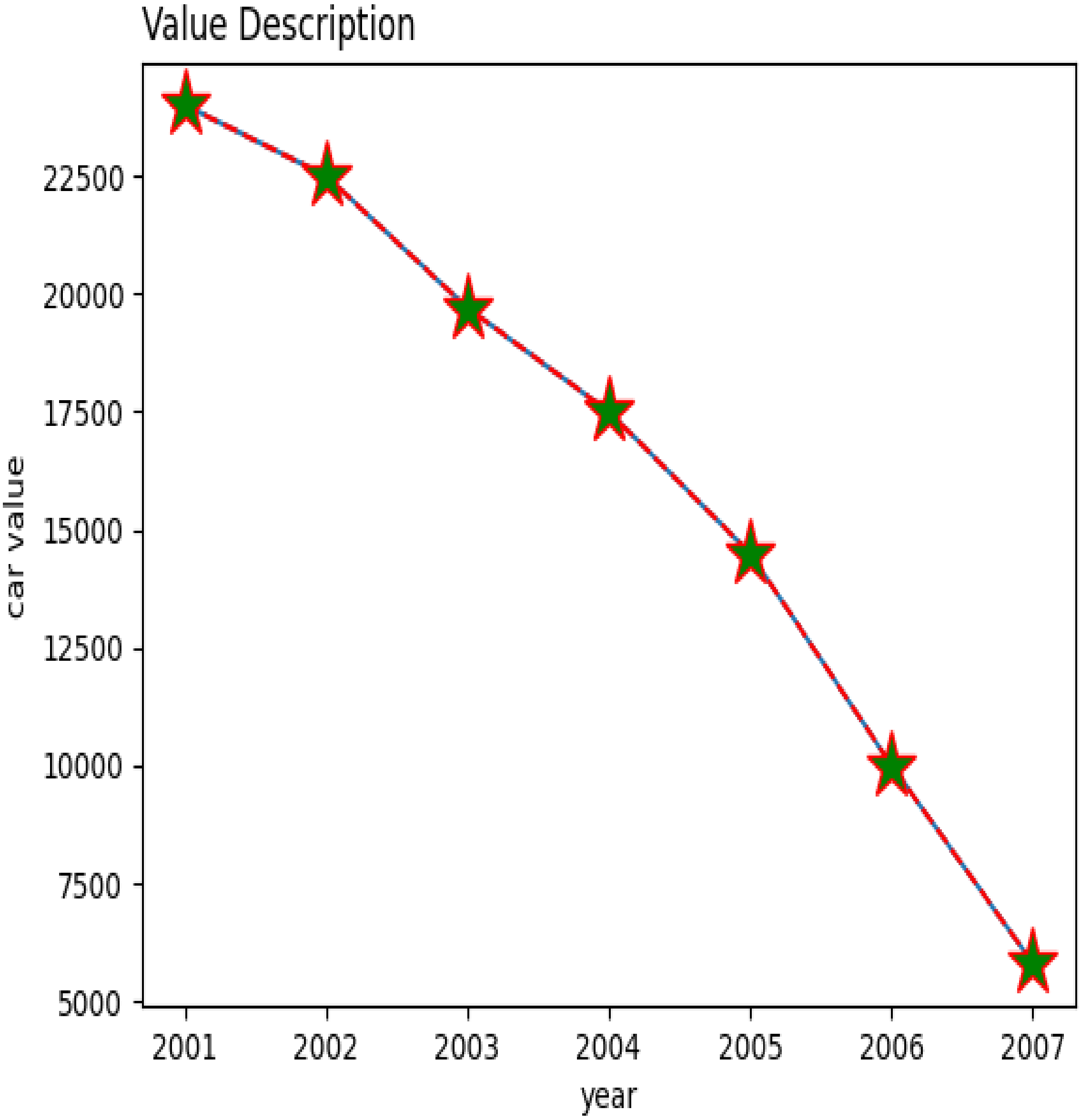
```
ANAKHA THOMAS T
21MCA004
Value of X=A -1 b:
[[  3.]
 [-11.]
 [  4.]]
```

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

CODE:

```
print(“ANAKHA THOMAS T”)
print(“21MCA004”)
from matplotlib import pyplot as plt
import numpy as np
x = np.array([2001,2002,2003,2004,2005,2006,2007])
y = np.array([24000,22500,19700,17500,14500,10000,5800])
plt.plot(x,y)
plt.xlabel("year")
plt.ylabel("car value")
plt.title("Value Description",loc='left')
plt.plot(x,y,linestyle='dashed',color='r',marker='*',markersize='20',markerfacecolor='green')
plt.show()
```

OUTPUT

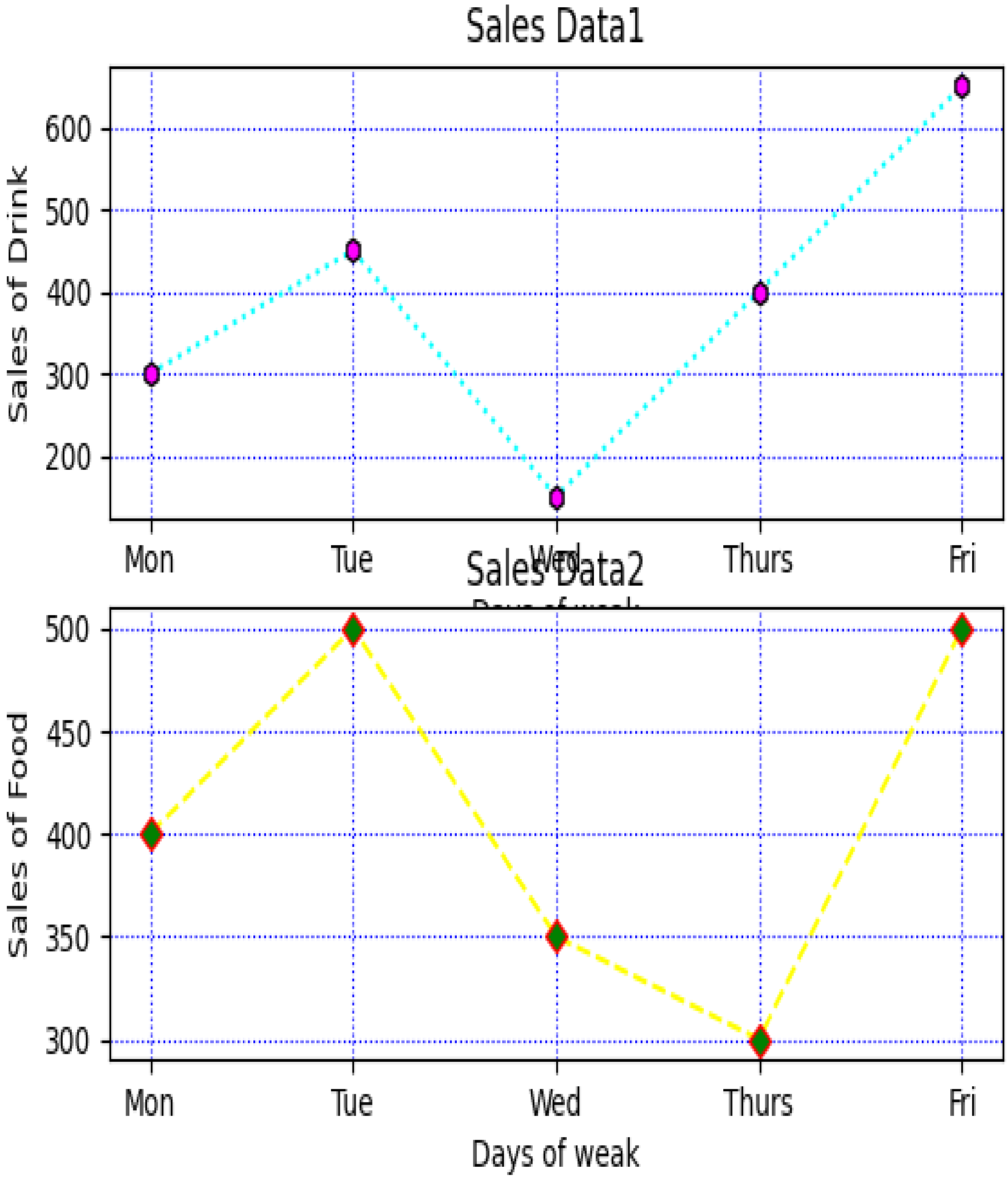

Value Description
car value
22500
20000
17500
15000
12500
10000
7500
5000
2001
2002
2003
2004
2005
2006
2007
year

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

CODE:

```
print(“ANAKHA THOMAS T”)
print(“21MCA004”)
import matplotlib.pyplot as plt
x = ['Mon','Tue','Wed','Thurs','Fri']
y = [300,450,150,400,650]
plt.subplot(2,1,1)
plt.plot(x,y,linestyle='dotted',color='cyan',marker='h',markerfacecolor='magenta',markeredgecolor='black')
plt.xlabel('Days of weak')
plt.ylabel('Sales of Drink')
plt.title("Sales Data1")
plt.grid(color='blue',linestyle=':')

x = ['Mon','Tue','Wed','Thurs','Fri']
y = [400,500,350,300,500]
plt.subplot(2,1,2)
plt.plot(x,y,linestyle='dashed',color='yellow',marker='D',markerfacecolor='green',markeredgecolor='red')
plt.xlabel('Days of weak')
plt.ylabel('Sales of Food')
plt.title("Sales Data2")
plt.grid(color='blue',linestyle=':')
plt.show()
```

OUTPUT

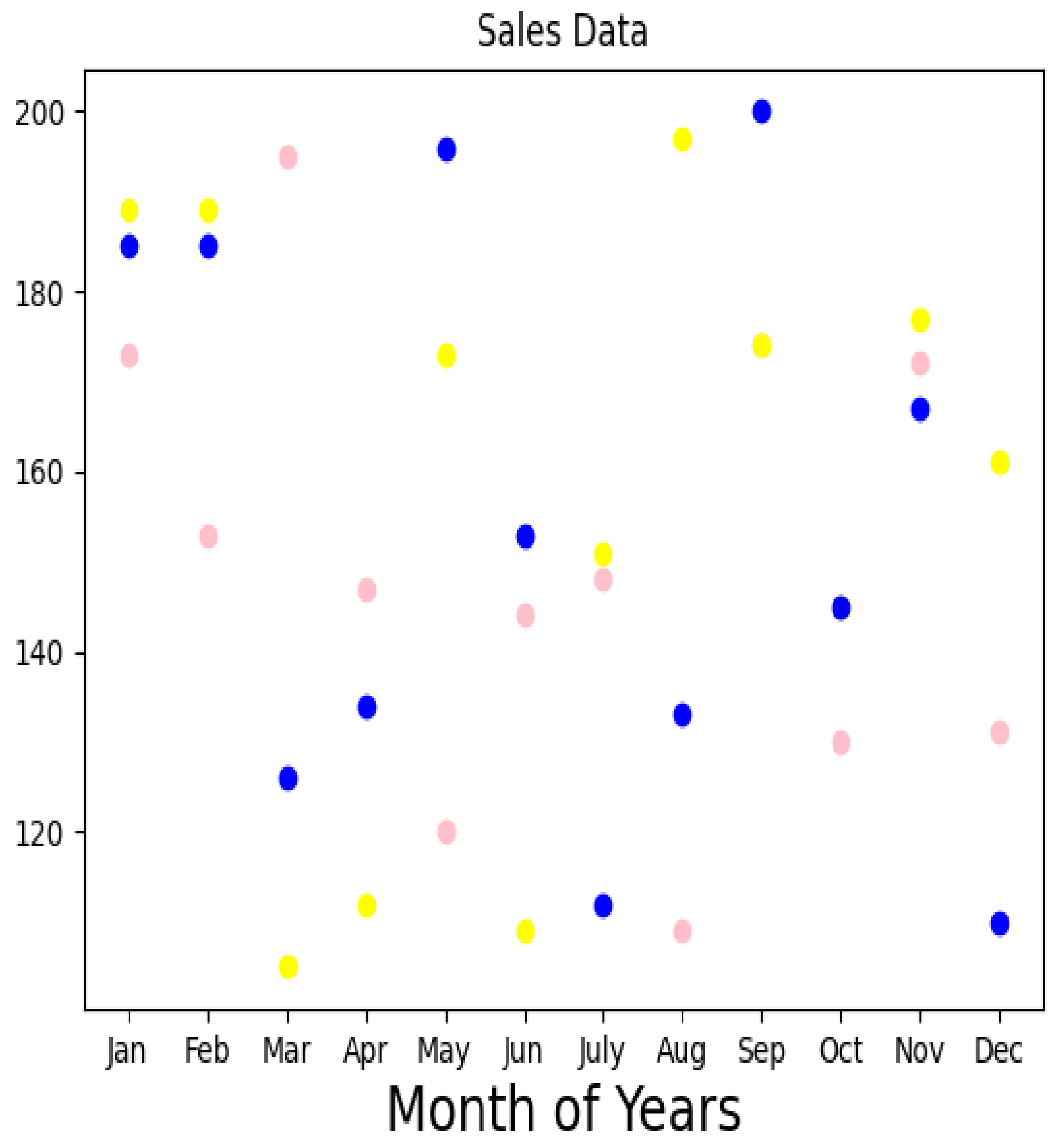

Sales Data1
600
500
400
300
200
Sales of Drink
Mon
Tue
Wed
Thurs
Fri
Days of weak
Sales Data2
500
450
400
350
300
Sales of Food
Mon
Tue
Wed
Thurs
Fri
Days of weak

9. Program to create a scatter plot for the product details.

CODE:

```
print(“ANAKHA THOMAS T”)
print(“21MCA004”)

import matplotlib.pyplot as plt
x = ['Jan','Feb','Mar','Apr','May','Jun','July','Aug','Sep','Oct','Nov','Dec']
y = [173,153,195,147,120,144,148,109,174,130,172,131]
plt.title('Sales Data')
plt.xlabel('Month of Years',fontsize=18)
plt.scatter(x,y,color='pink')

x = ['Jan','Feb','Mar','Apr','May','Jun','July','Aug','Sep','Oct','Nov','Dec']
y = [189,189,105,112,173,109,151,197,174,145,177,161]
plt.scatter(x,y,color='yellow')

x = ['Jan','Feb','Mar','Apr','May','Jun','July','Aug','Sep','Oct','Nov','Dec']
y = [185,185,126,134,196,153,112,133,200,145,167,110]
plt.scatter(x,y,color='blue')
plt.show()
```

OUTPUT

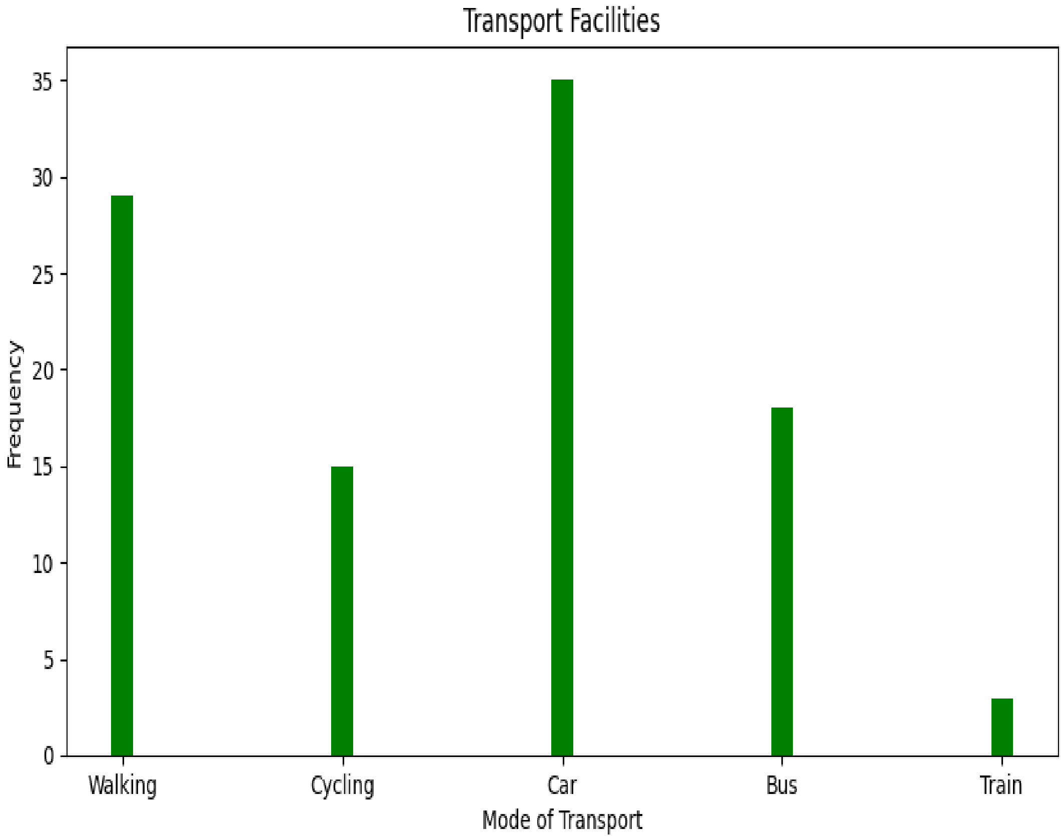

Sales Data
200
180
160
140
120
Jan
Feb
Mar
Apr
May
Jun
July
Aug
Sep
Oct
Nov
Dec
Month of Years

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

CODE:

```
print(“ANAKHA THOMAS T”)
print(“21MCA004”)
import matplotlib.pyplot as plt
import numpy as np
data={'Walking': 29,'Cycling': 15,'Car':35,'Bus':18,'Train':3}
transport=list(data.keys())
frequency = list(data.values())

fig = plt.figure(figsize = (10, 5))
plt.bar(transport, frequency, color ='green', width = 0.1)
plt.xlabel("Mode of Transport")
plt.ylabel("Frequency")
plt.title("Transport Facilities")
plt.show()
```

OUTPUT

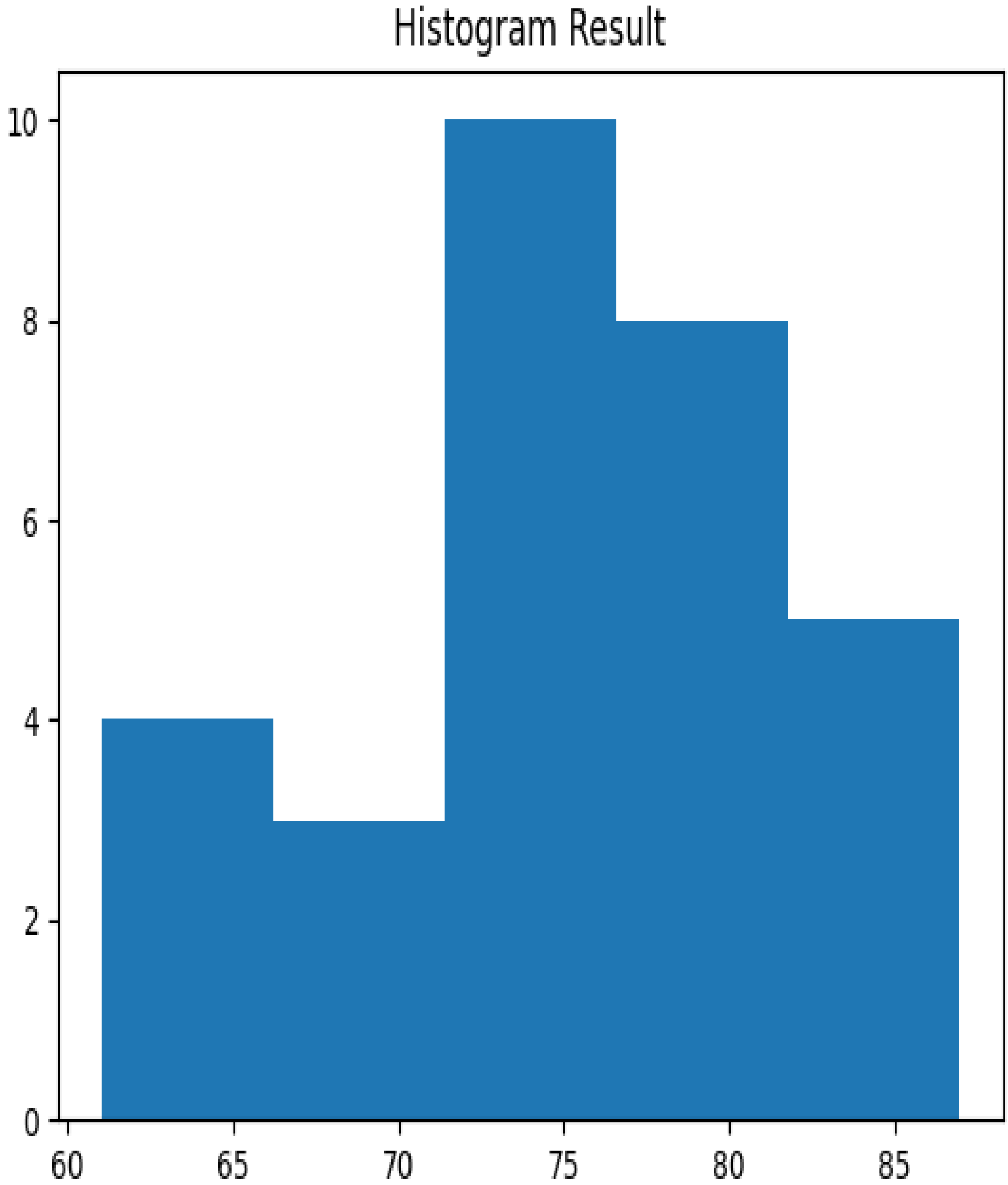

Transport Facilities
Frequency
35
30
25
20
15
10
5
0
Walking
Cycling
Car
Bus
Train
Mode of Transport

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

CODE:

```
print(“ANAKHA THOMAS T”)
print(“21MCA004”)
import matplotlib.pyplot as plt
import numpy as np
fig,ax = plt.subplots(1,1)
a=np.array([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(a, bins =5)
plt.title("Histogram Result")
plt.show()
```

OUTPUT

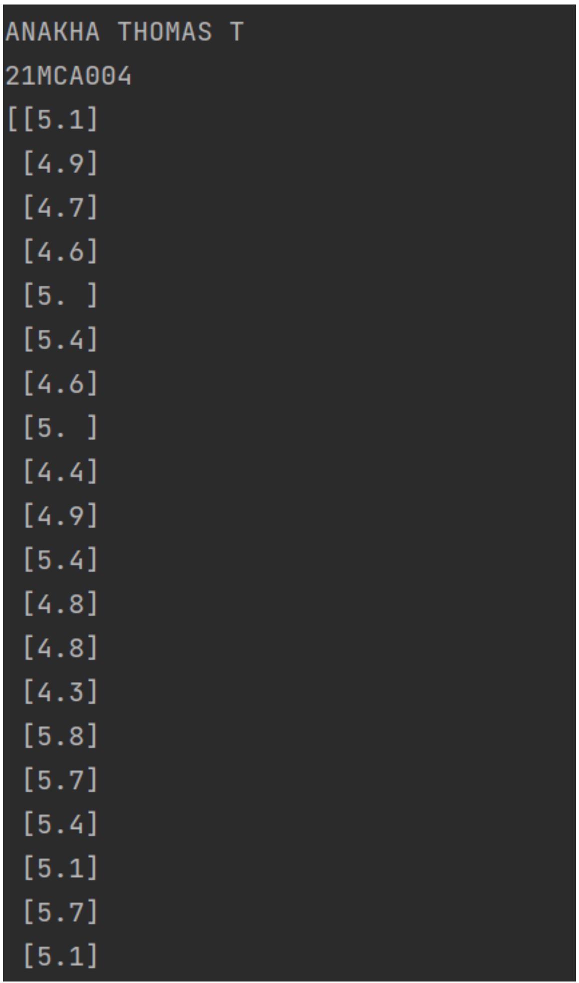


12. Write a program to implement KNN algorithm using iris data Set. Use different values for K and different values for text and training data.

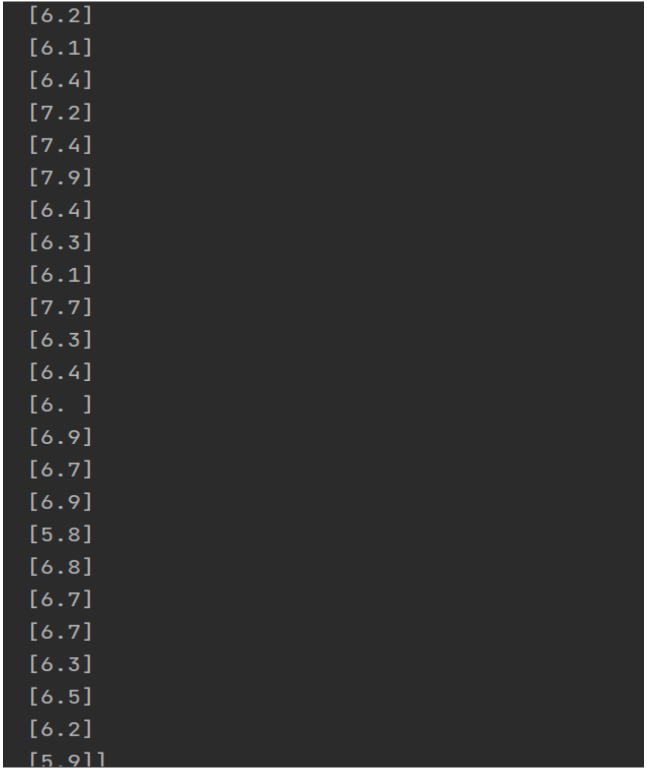
CODE:

```
print(“ANAKHA THOMAS T”)
print(“21MCA004”)
import pandas as pd
dataset=pd.read_csv("iris.csv")
X = dataset.iloc[:,:1].values
y = dataset.iloc[:,4].values
print(X)
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=0.20)
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors=5)
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))
```

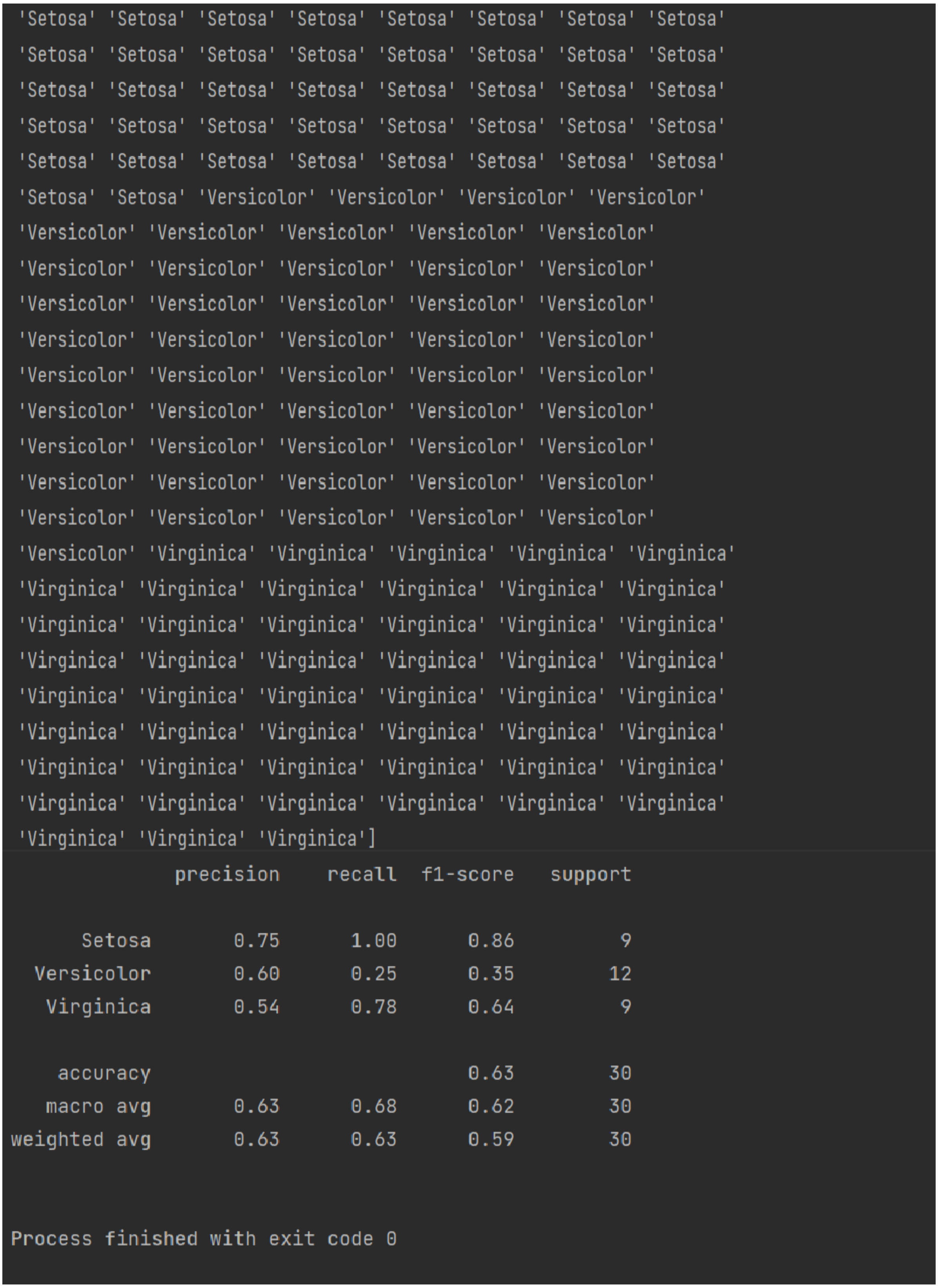
OUTPUT



```
ANAKHA THOMAS T
21MCA004
[[5.1]
 [4.9]
 [4.7]
 [4.6]
 [5. ]
 [5.4]
 [4.6]
 [5. ]
 [4.4]
 [4.9]
 [5.4]
 [4.8]
 [4.8]
 [4.3]
 [5.8]
 [5.7]
 [5.4]
 [5.1]
 [5.7]
 [5.1]
```



```
 [6.2]
 [6.1]
 [6.4]
 [7.2]
 [7.4]
 [7.9]
 [6.4]
 [6.3]
 [6.1]
 [7.7]
 [6.3]
 [6.4]
 [6. ]
 [6.9]
 [6.7]
 [6.9]
 [5.8]
 [6.8]
 [6.7]
 [6.7]
 [6.3]
 [6.5]
 [6.2]
 [5.9]]
```



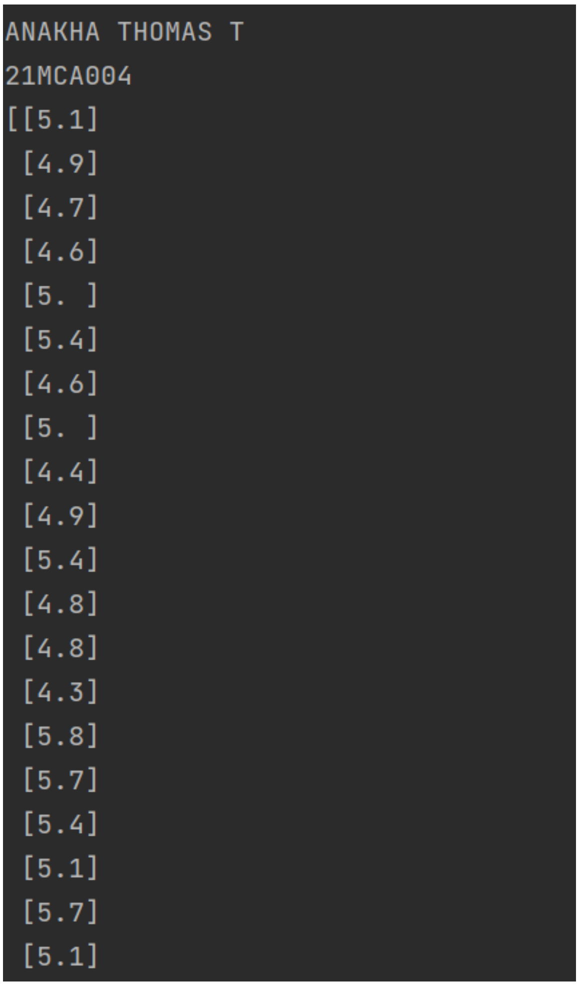
```
 'Setosa' 'Setosa' 'Setosa' 'Setosa' 'Setosa' 'Setosa' 'Setosa' 'Setosa'
 'Setosa' 'Setosa' 'Setosa' 'Setosa' 'Setosa' 'Setosa' 'Setosa' 'Setosa'
 'Setosa' 'Setosa' 'Setosa' 'Setosa' 'Setosa' 'Setosa' 'Setosa' 'Setosa'
 'Setosa' 'Setosa' 'Setosa' 'Setosa' 'Setosa' 'Setosa' 'Setosa' 'Setosa'
 'Setosa' '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' '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       0.75      1.00      0.86         9
  Versicolor       0.60      0.25      0.35        12
   Virginica       0.54      0.78      0.64         9

    accuracy                           0.63        30
   macro avg       0.63      0.68      0.62        30
weighted avg       0.63      0.63      0.59        30



Process finished with exit code 0
```

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

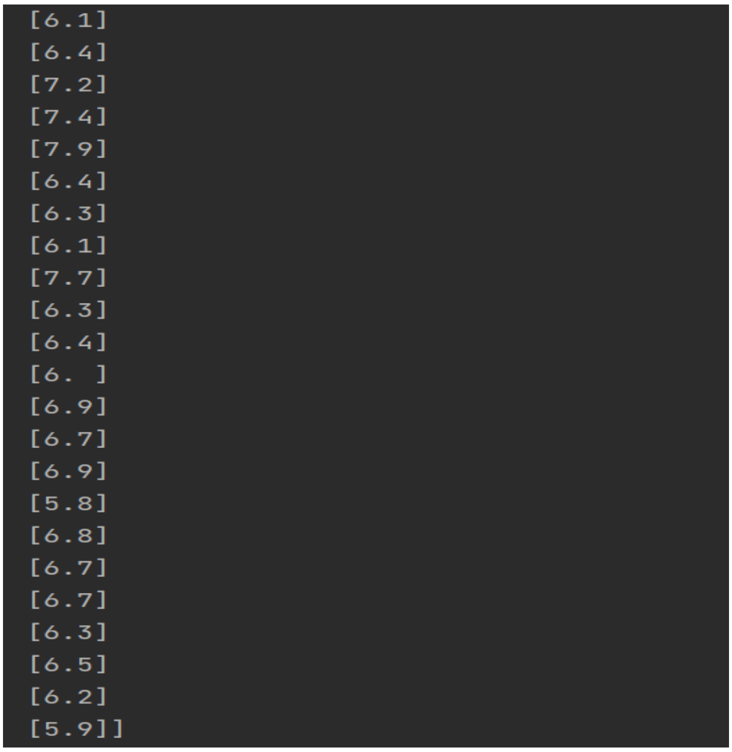
CODE:

```
print(“ANAKHA THOMAS T”)
print(“21MCA004”)
import pandas as pd
dataset = pd.read_csv('iris.csv')
X = dataset.iloc[:, :1].values
y = dataset.iloc[:, 4].values
print(X)
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=0.20)

from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
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))
df = pd.DataFrame({'Real Values':y_test, 'Predicted Values':y_pred})
print(df)
```

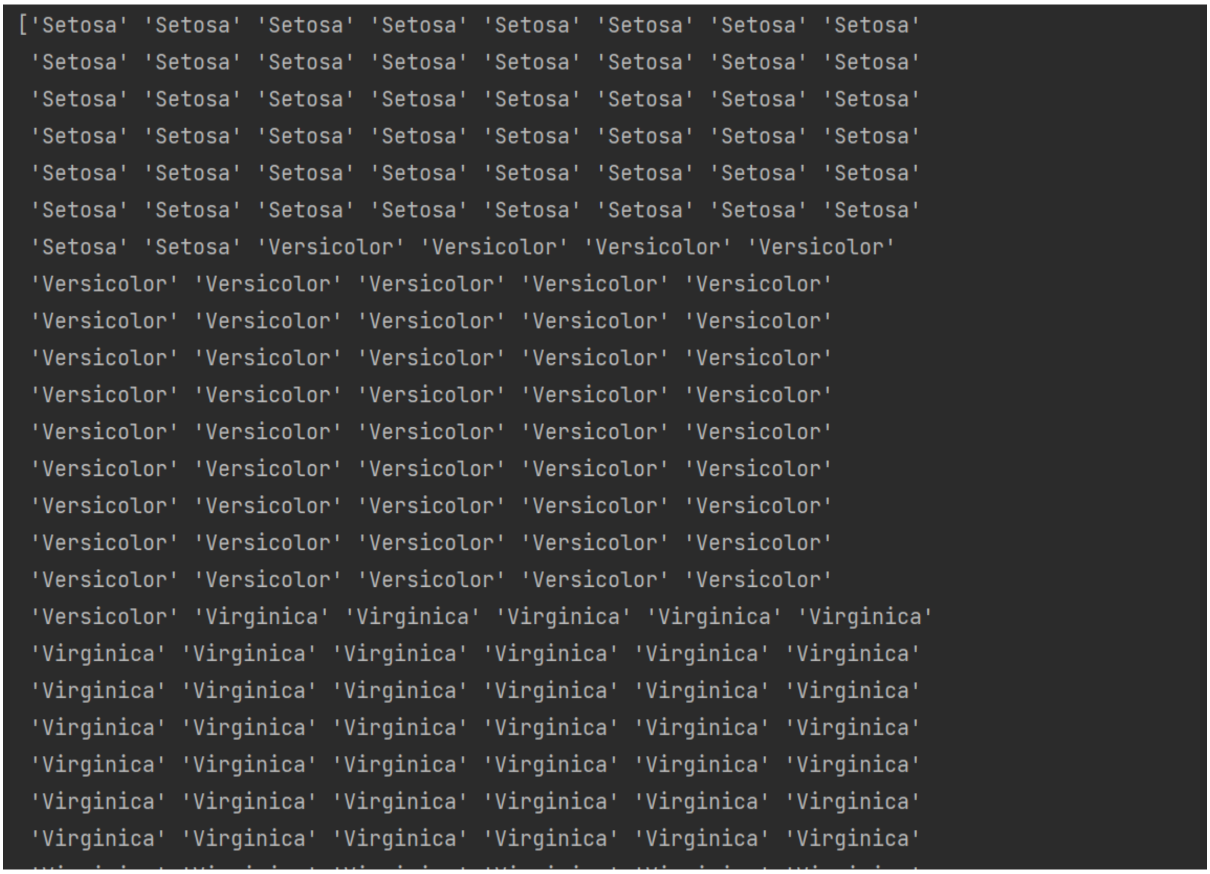
OUTPUT



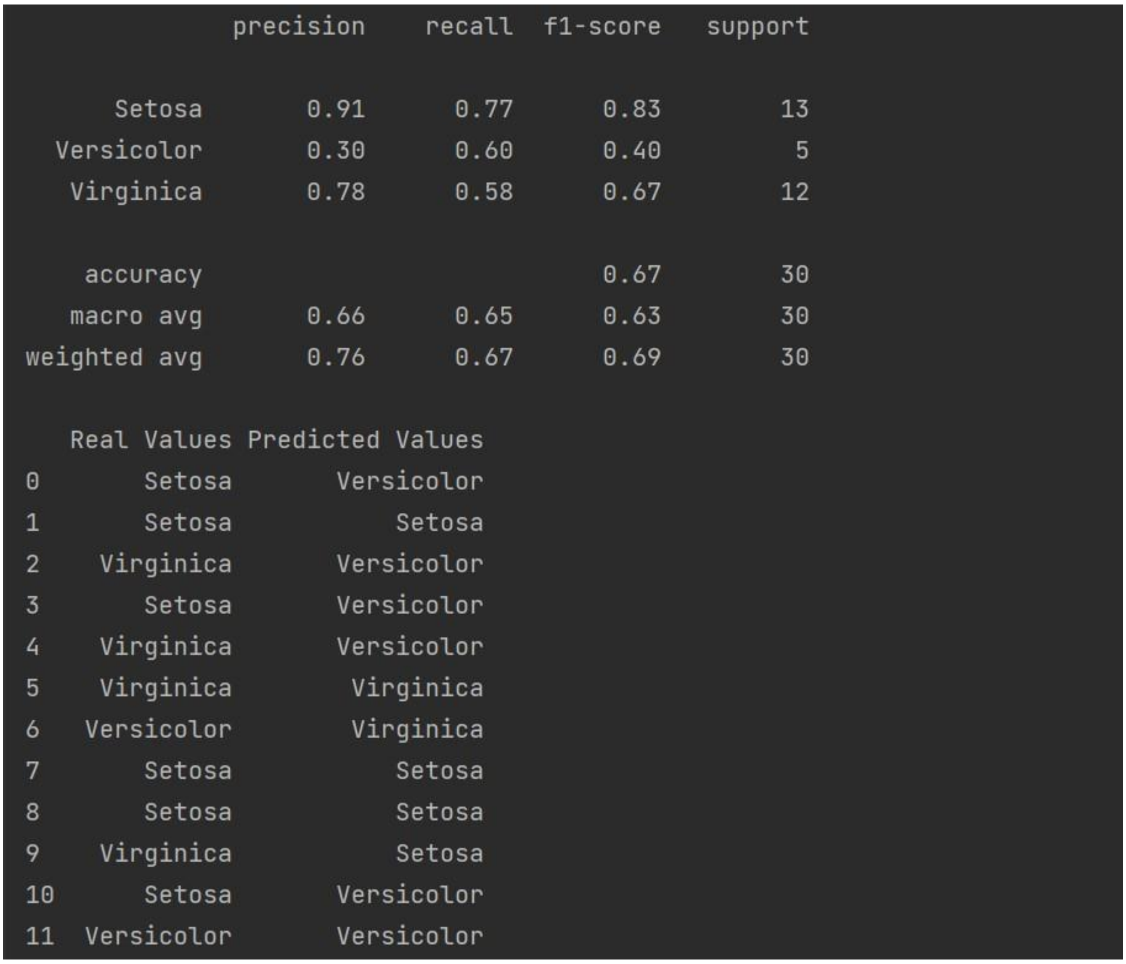
```
ANAKHA THOMAS T
21MCA004
[[5.1]
 [4.9]
 [4.7]
 [4.6]
 [5. ]
 [5.4]
 [4.6]
 [5. ]
 [4.4]
 [4.9]
 [5.4]
 [4.8]
 [4.8]
 [4.3]
 [5.8]
 [5.7]
 [5.4]
 [5.1]
 [5.7]
 [5.1]
```



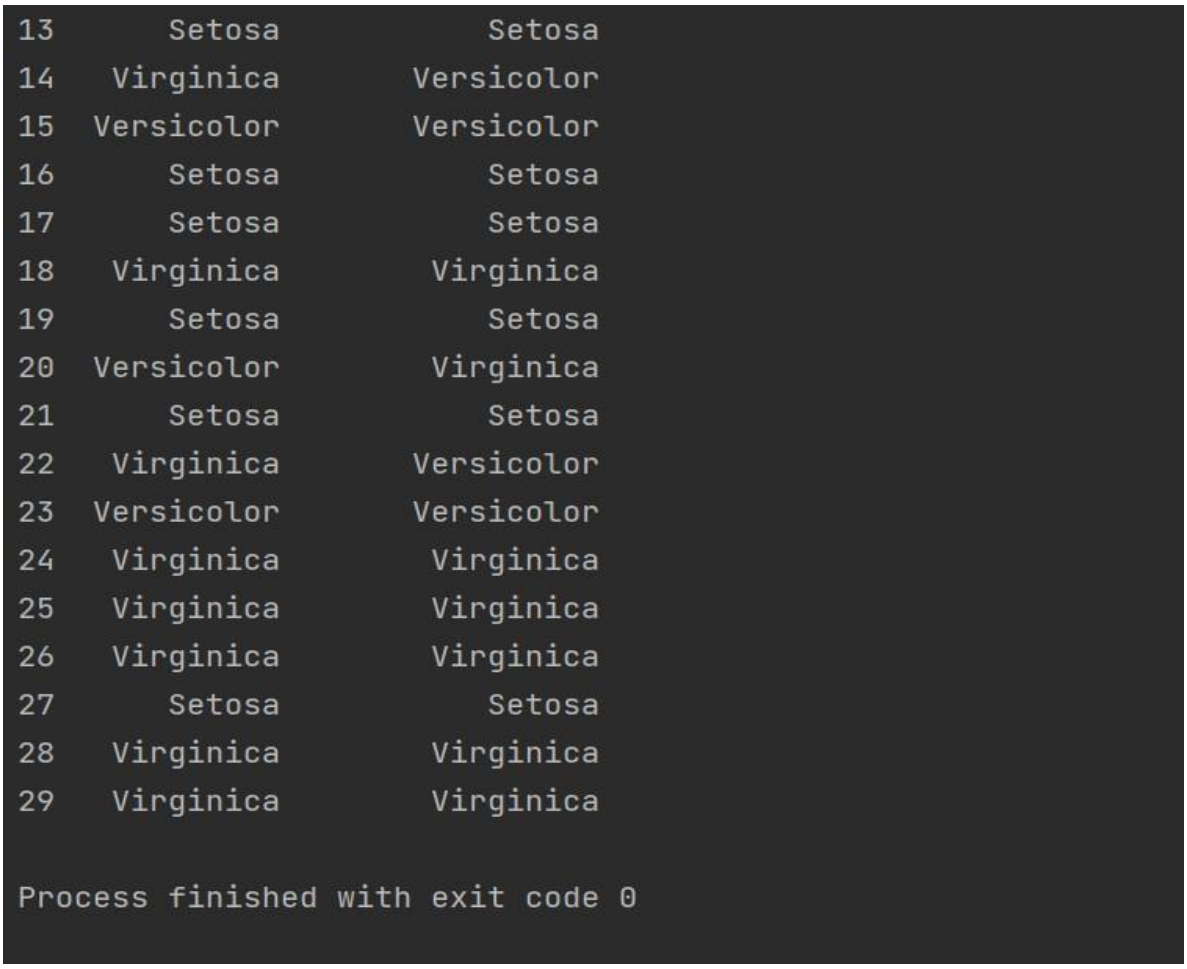
```
 [6.1]
 [6.4]
 [7.2]
 [7.4]
 [7.9]
 [6.4]
 [6.3]
 [6.1]
 [7.7]
 [6.3]
 [6.4]
 [6. ]
 [6.9]
 [6.7]
 [6.9]
 [5.8]
 [6.8]
 [6.7]
 [6.7]
 [6.3]
 [6.5]
 [6.2]
 [5.9]]
```



```
['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' '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'
```



```
              precision    recall  f1-score   support

      Setosa       0.91      0.77      0.83        13
  Versicolor       0.30      0.60      0.40         5
   Virginica       0.78      0.58      0.67        12

    accuracy                           0.67        30
   macro avg       0.66      0.65      0.63        30
weighted avg       0.76      0.67      0.69        30

   Real Values Predicted Values
0       Setosa       Versicolor
1       Setosa           Setosa
2    Virginica       Versicolor
3       Setosa       Versicolor
4    Virginica       Versicolor
5    Virginica        Virginica
6   Versicolor        Virginica
7       Setosa           Setosa
8       Setosa           Setosa
9    Virginica           Setosa
10      Setosa       Versicolor
11  Versicolor       Versicolor
```



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

Process finished with exit code 0
```

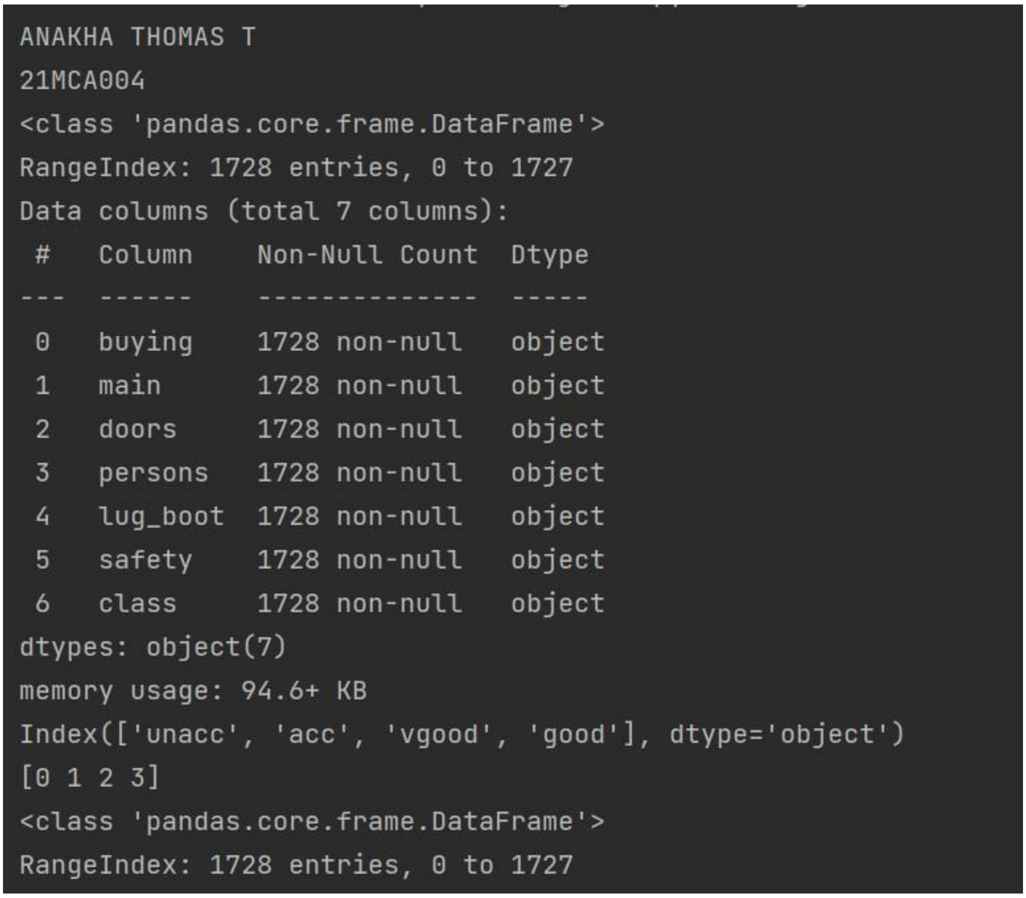
14. Write a program to implement decision tree algorithm using the given data set

CODE:

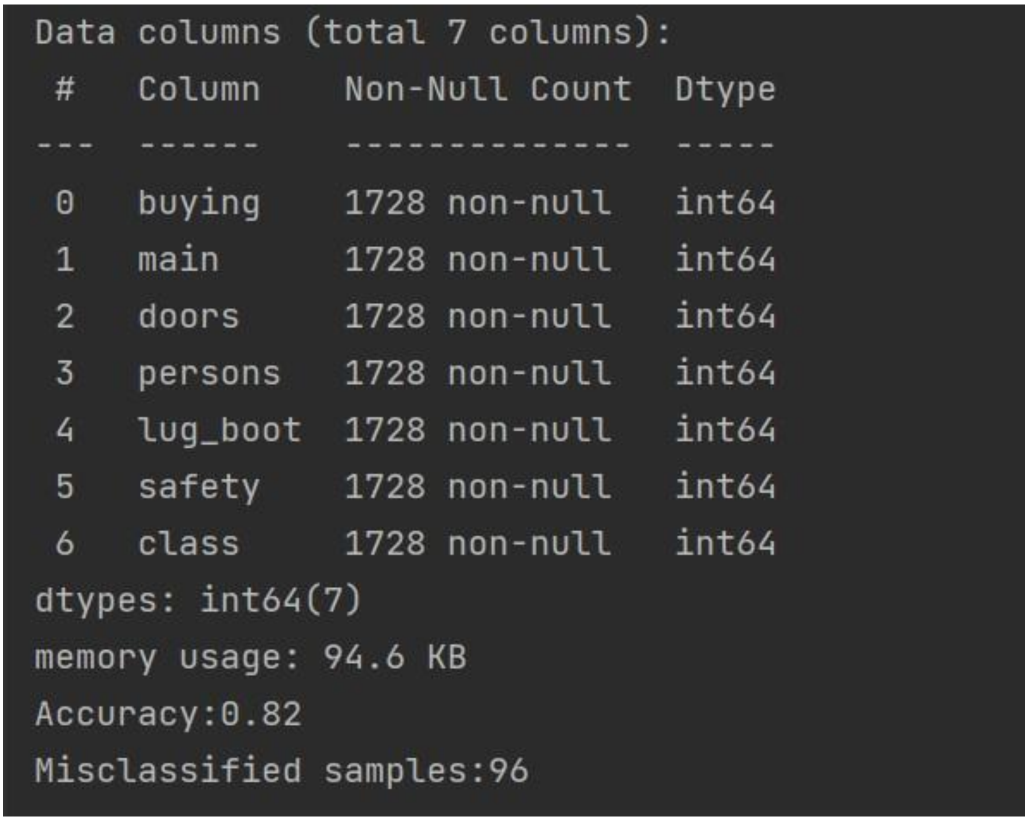
```
print("ANAKHA THOMAS T")
print("21MCA004")
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import tree,metrics,model_selection
data=pd.read_csv('car.csv',names=['buying','main','doors','persons','lug_boot','safety','class'])
data.head()
data.info()

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

print(data['class'].unique())
data['buying'],_ = pd.factorize(data['buying'])
data['main'],_ = pd.factorize(data['main'])
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'])
data.head()
data.info()
x=data.iloc[:,:-1]
```

```
y=data.iloc[:,-1]
x_train,x_test,y_train,y_test=model_selection.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)
accuracy = metrics.accuracy_score(y_test,y_pred)
print('Accuracy:{:.2f}'.format(accuracy))
count_misclassified = (y_test != y_pred).sum()
print('Misclassified samples:{}'.format(count_misclassified))
```

OUTPUT



```
ANAKHA THOMAS T
21MCA004
<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   main      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]
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1728 entries, 0 to 1727
```



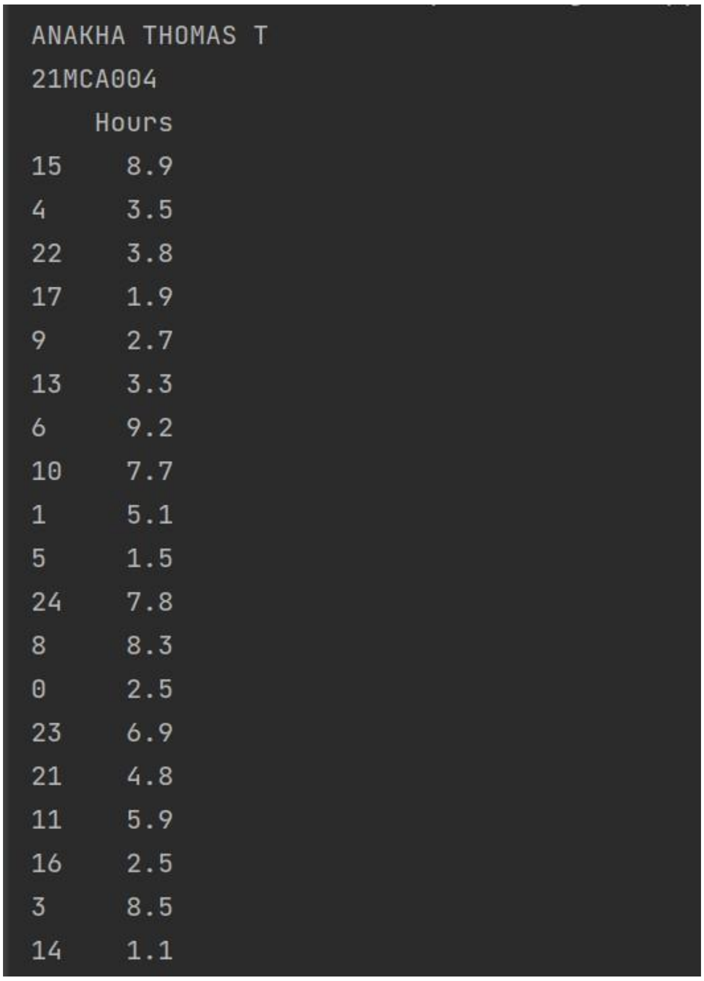
```
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   buying    1728 non-null   int64
 1   main      1728 non-null   int64
 2   doors     1728 non-null   int64
 3   persons   1728 non-null   int64
 4   lug_boot  1728 non-null   int64
 5   safety    1728 non-null   int64
 6   class     1728 non-null   int64
dtypes: int64(7)
memory usage: 94.6 KB
Accuracy:0.82
Misclassified samples:96
```

15. Write a program to demonstrate Simple Linear Regression using given data set

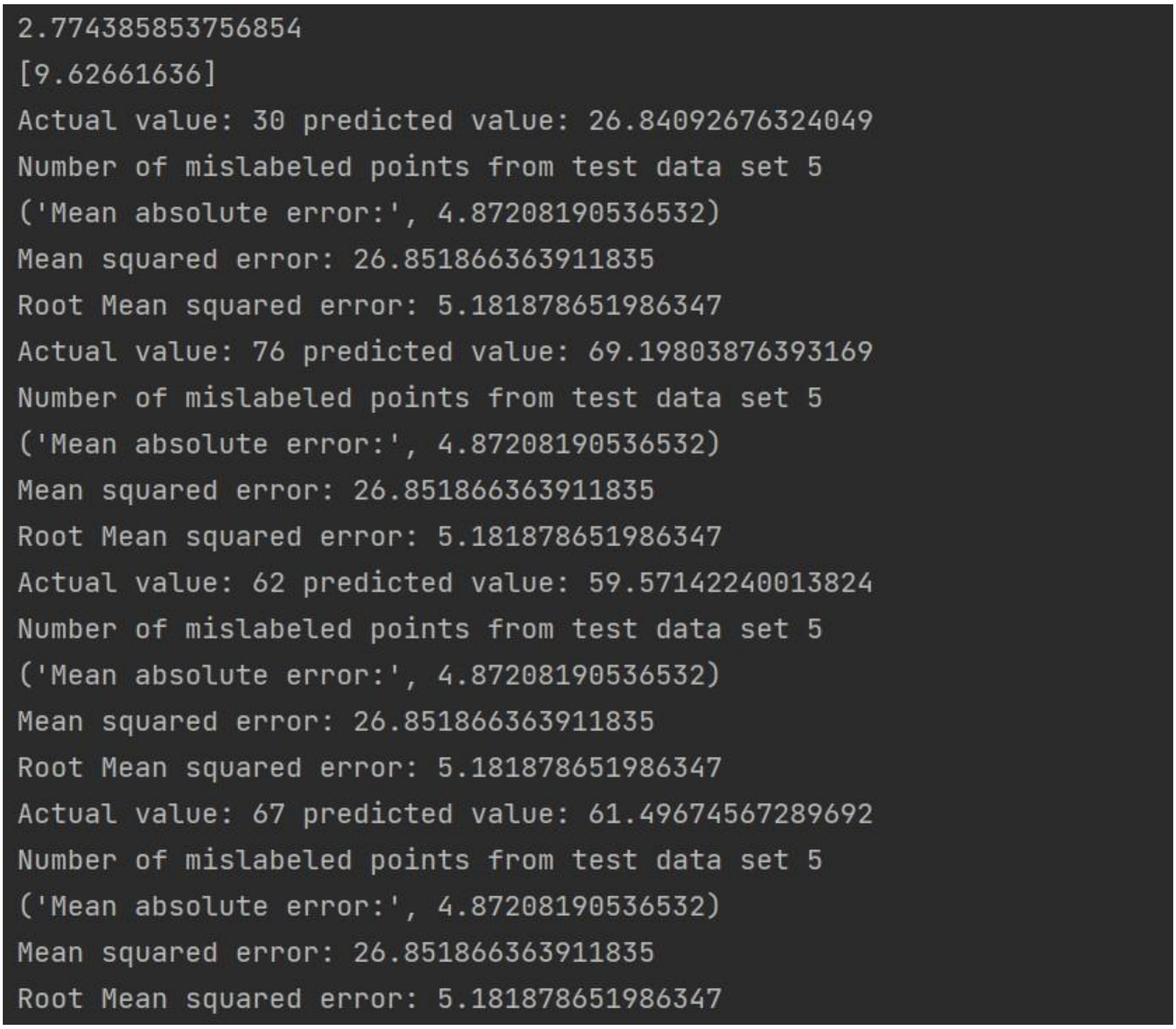
CODE:

```
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 mislabeled points from test data set", (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("RootMeansquarederror:",np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
```

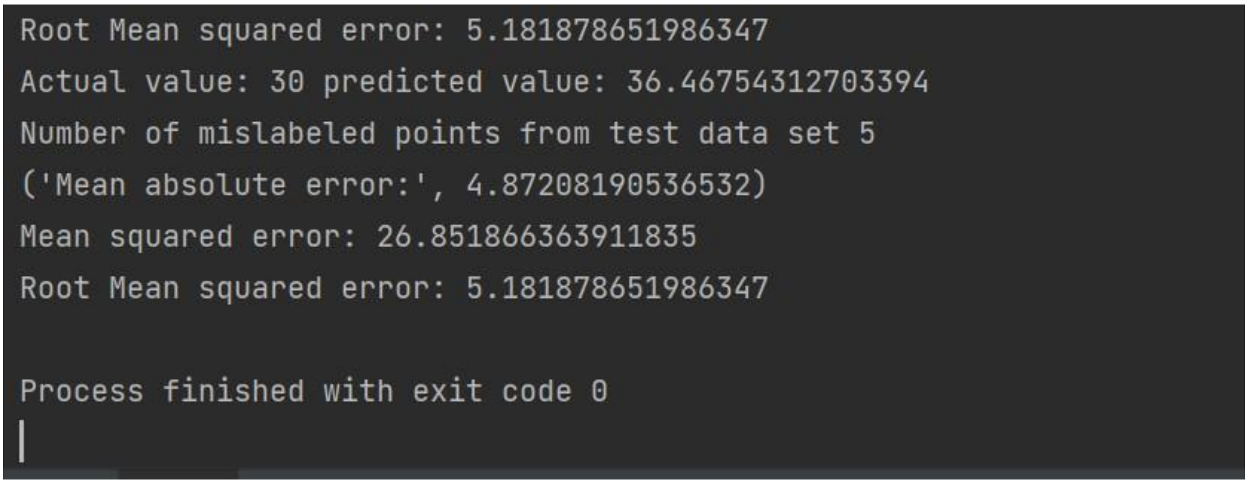
OUTPUT



```
ANAKHA THOMAS T
21MCA004
    Hours
15    8.9
4     3.5
22    3.8
17    1.9
9     2.7
13    3.3
6     9.2
10    7.7
1     5.1
5     1.5
24    7.8
8     8.3
0     2.5
23    6.9
21    4.8
11    5.9
16    2.5
3     8.5
14    1.1
```



```
2.774385853756854
[9.62661636]
Actual value: 30 predicted value: 26.84092676324049
Number of mislabeled points from test data set 5
('Mean absolute error:', 4.87208190536532)
Mean squared error: 26.851866363911835
Root Mean squared error: 5.181878651986347
Actual value: 76 predicted value: 69.19803876393169
Number of mislabeled points from test data set 5
('Mean absolute error:', 4.87208190536532)
Mean squared error: 26.851866363911835
Root Mean squared error: 5.181878651986347
Actual value: 62 predicted value: 59.57142240013824
Number of mislabeled points from test data set 5
('Mean absolute error:', 4.87208190536532)
Mean squared error: 26.851866363911835
Root Mean squared error: 5.181878651986347
Actual value: 67 predicted value: 61.49674567289692
Number of mislabeled points from test data set 5
('Mean absolute error:', 4.87208190536532)
Mean squared error: 26.851866363911835
Root Mean squared error: 5.181878651986347
```



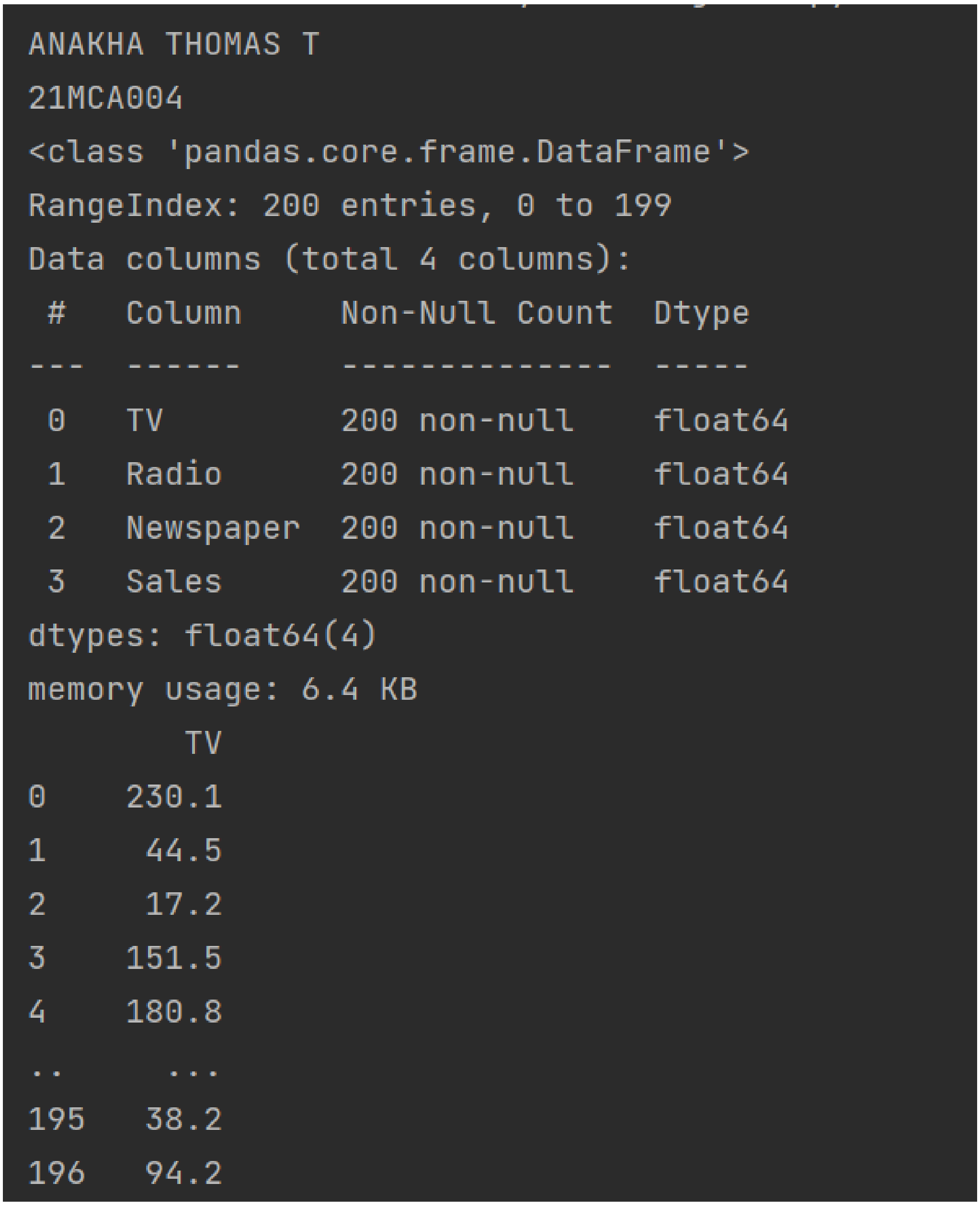
```
Root Mean squared error: 5.181878651986347
Actual value: 30 predicted value: 36.46754312703394
Number of mislabeled points from test data set 5
('Mean absolute error:', 4.87208190536532)
Mean squared error: 26.851866363911835
Root Mean squared error: 5.181878651986347

Process finished with exit code 0
```

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

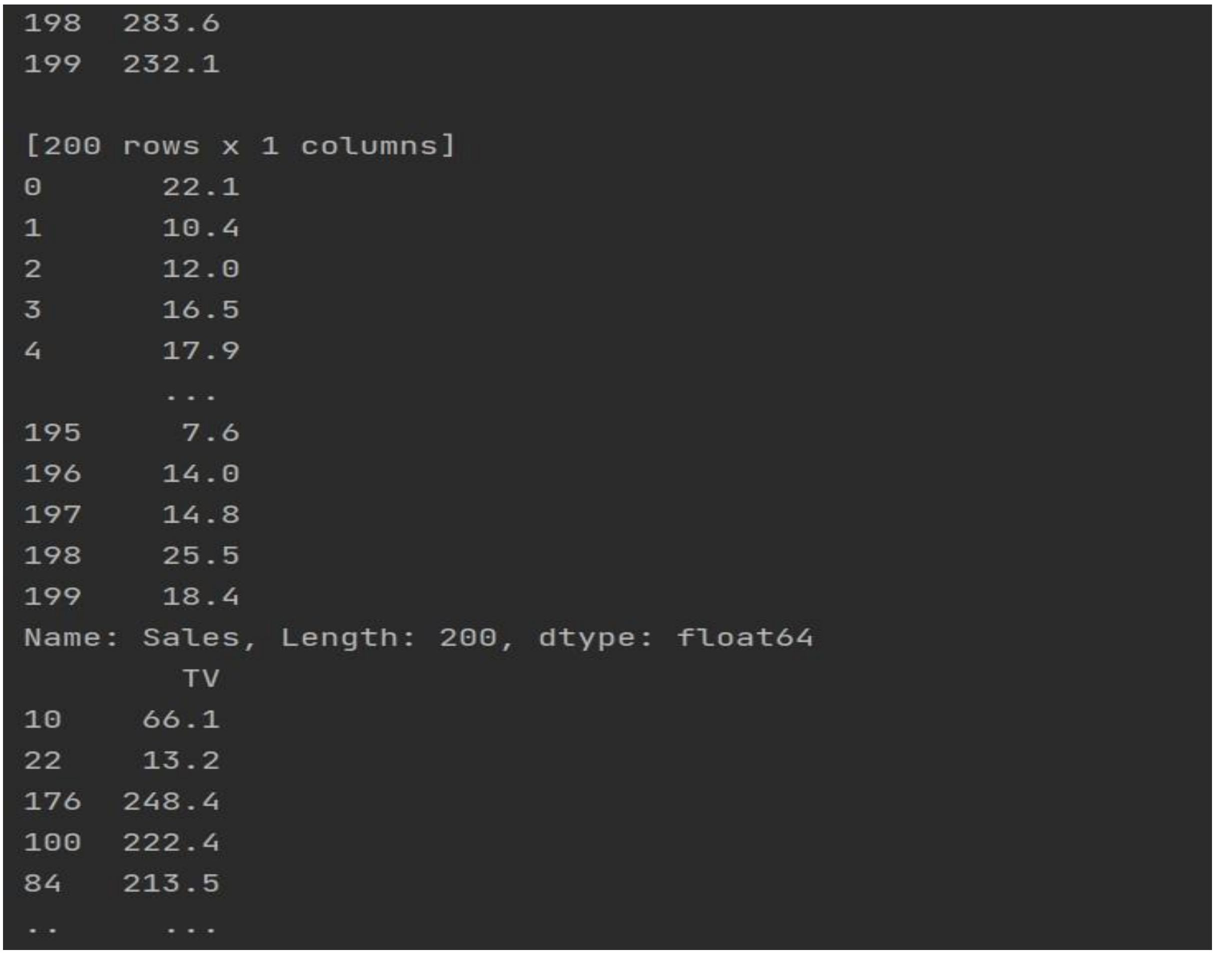
CODE:

```
import numpy as np
import pandas as pd
import matplotlib as plt
#import pd as pd
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("Numberofmislabeledpointsfromtestdata set",(y_test!=y_pred).sum())
```

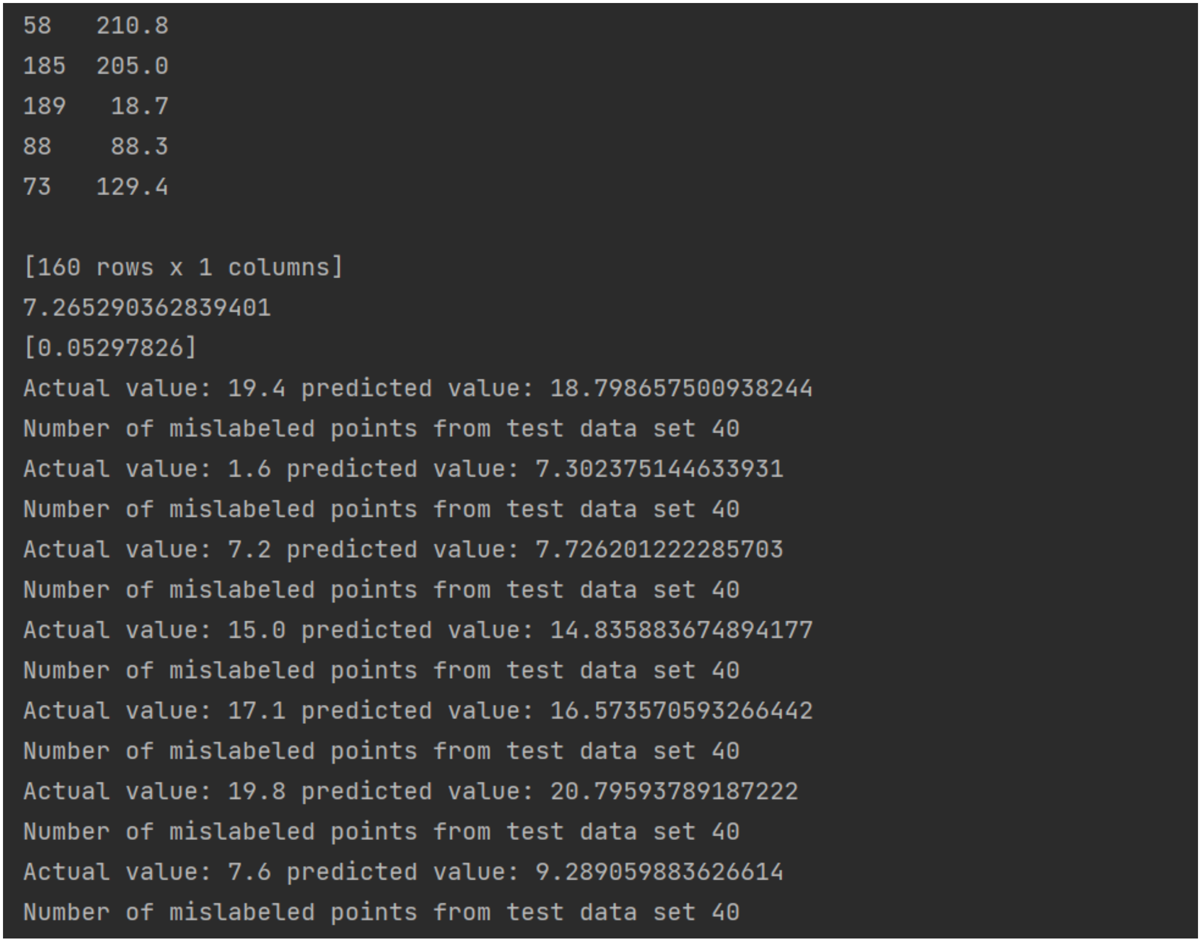
OUTPUT



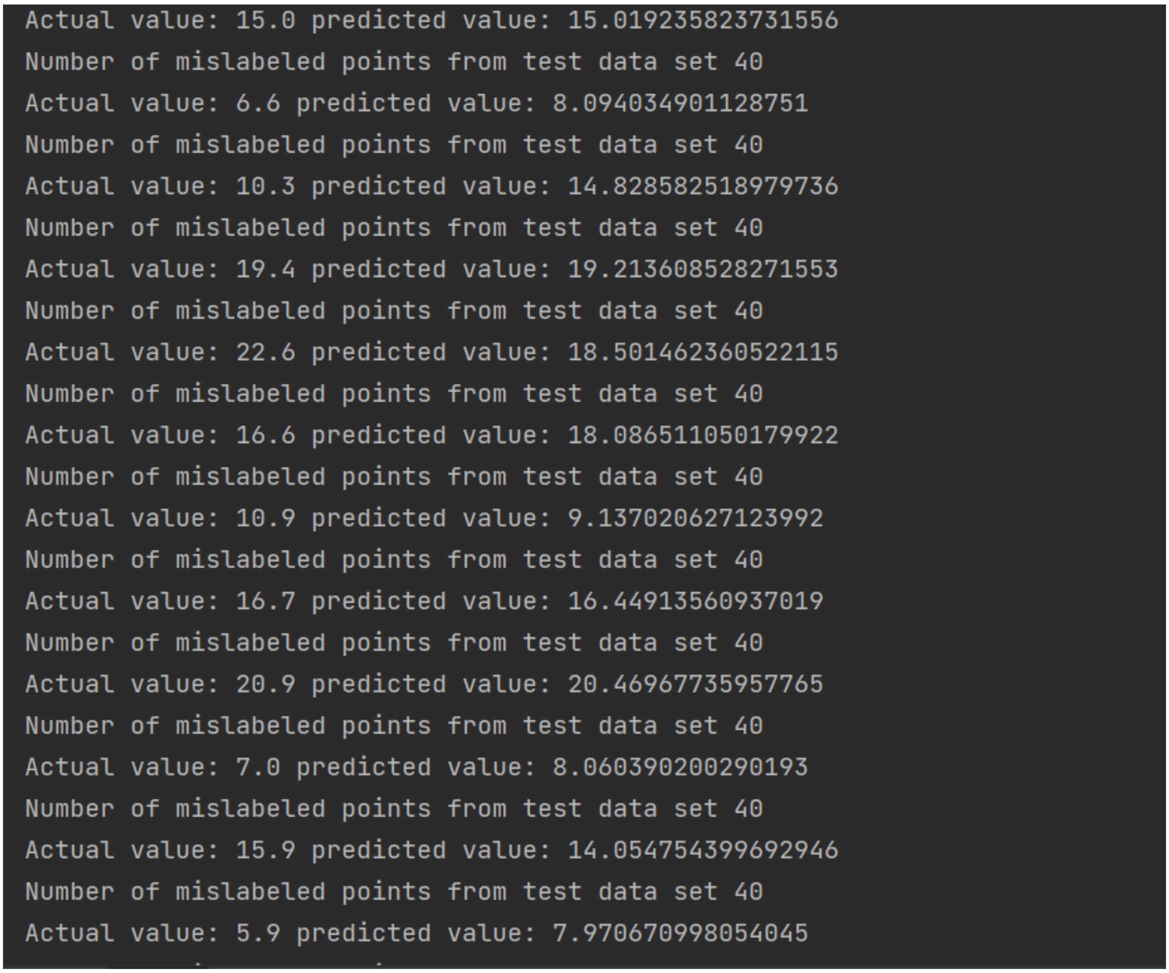
```
ANAKHA THOMAS T
21MCA004
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   TV         200 non-null    float64
 1   Radio      200 non-null    float64
 2   Newspaper  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
```



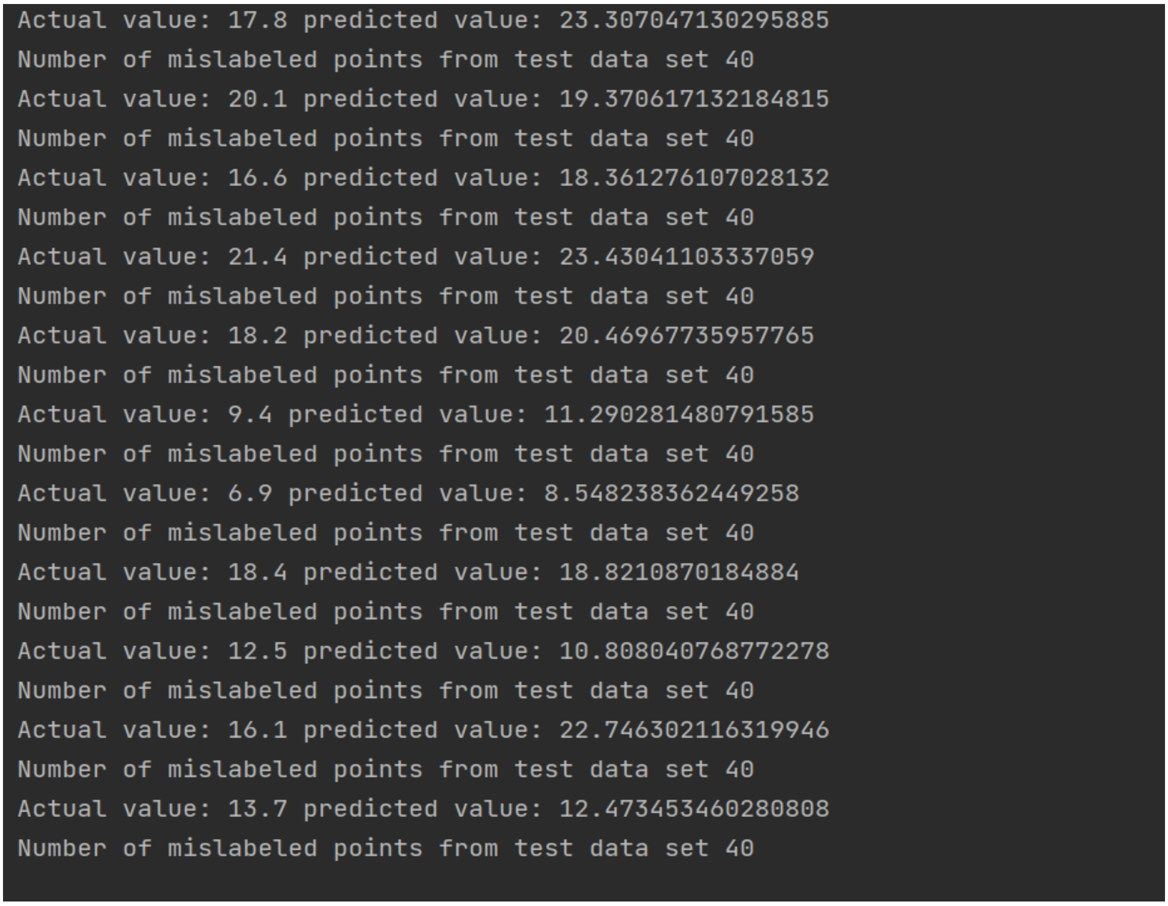
```
198  283.6
199  232.1

[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
10    66.1
22    13.2
176  248.4
100  222.4
84   213.5
..     ...
```



```
58   210.8
185  205.0
189   18.7
88    88.3
73   129.4

[160 rows x 1 columns]
7.265290362839401
[0.05297826]
Actual value: 19.4 predicted value: 18.798657500938244
Number of mislabeled points from test data set 40
Actual value: 1.6 predicted value: 7.302375144633931
Number of mislabeled points from test data set 40
Actual value: 7.2 predicted value: 7.726201222285703
Number of mislabeled points from test data set 40
Actual value: 15.0 predicted value: 14.835883674894177
Number of mislabeled points from test data set 40
Actual value: 17.1 predicted value: 16.573570593266442
Number of mislabeled points from test data set 40
Actual value: 19.8 predicted value: 20.79593789187222
Number of mislabeled points from test data set 40
Actual value: 7.6 predicted value: 9.289059883626614
Number of mislabeled points from test data set 40
```



```
Actual value: 15.0 predicted value: 15.019235823731556
Number of mislabeled points from test data set 40
Actual value: 6.6 predicted value: 8.094034901128751
Number of mislabeled points from test data set 40
Actual value: 10.3 predicted value: 14.828582518979736
Number of mislabeled points from test data set 40
Actual value: 19.4 predicted value: 19.213608528271553
Number of mislabeled points from test data set 40
Actual value: 22.6 predicted value: 18.501462360522115
Number of mislabeled points from test data set 40
Actual value: 16.6 predicted value: 18.086511050179922
Number of mislabeled points from test data set 40
Actual value: 10.9 predicted value: 9.137020627123992
Number of mislabeled points from test data set 40
Actual value: 16.7 predicted value: 16.44913560937019
Number of mislabeled points from test data set 40
Actual value: 20.9 predicted value: 20.46967735957765
Number of mislabeled points from test data set 40
Actual value: 7.0 predicted value: 8.060390200290193
Number of mislabeled points from test data set 40
Actual value: 15.9 predicted value: 14.054754399692946
Number of mislabeled points from test data set 40
Actual value: 5.9 predicted value: 7.970670998054045
```



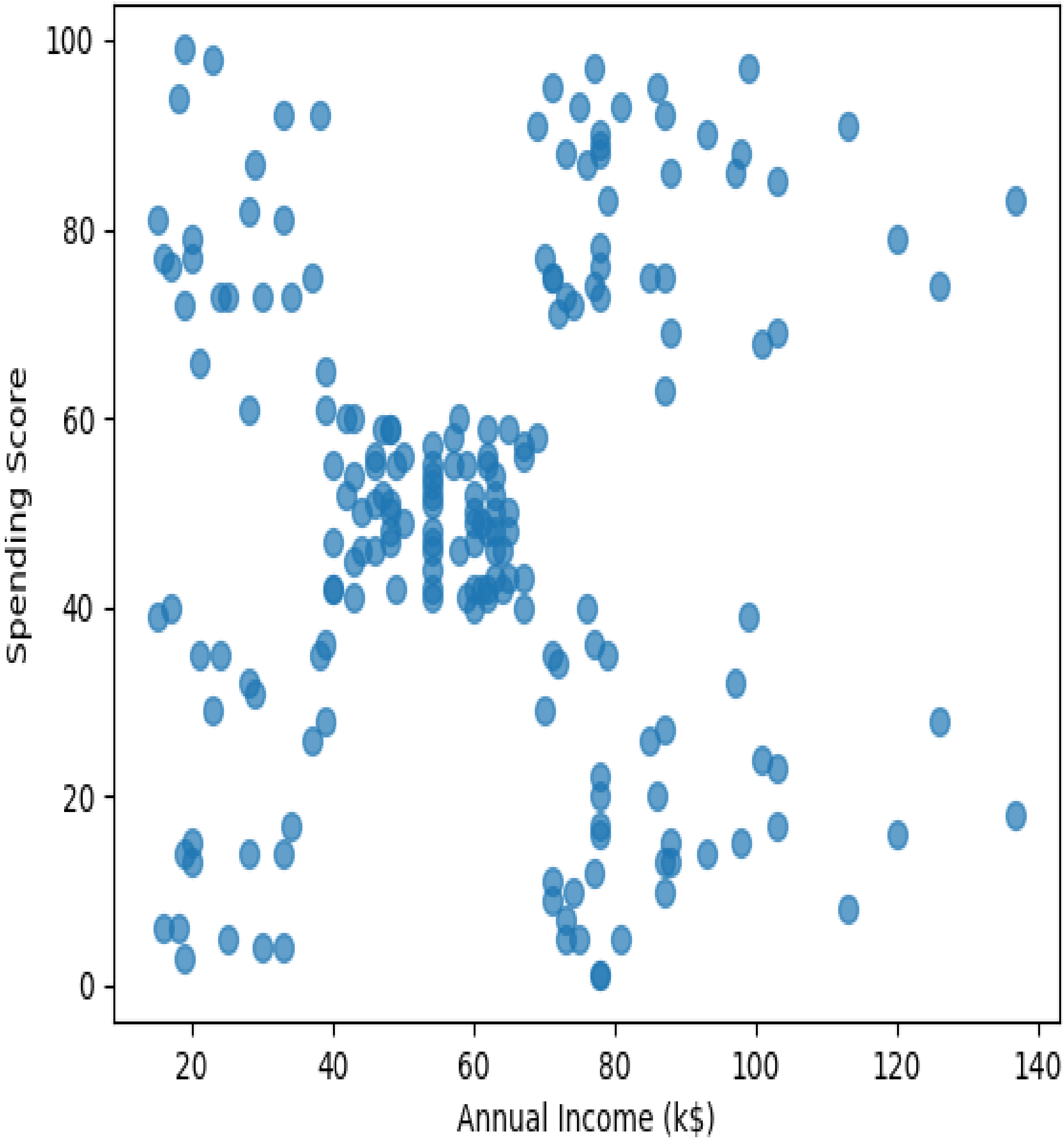
```
Actual value: 17.8 predicted value: 23.307047130295885
Number of mislabeled points from test data set 40
Actual value: 20.1 predicted value: 19.370617132184815
Number of mislabeled points from test data set 40
Actual value: 16.6 predicted value: 18.361276107028132
Number of mislabeled points from test data set 40
Actual value: 21.4 predicted value: 23.43041103337059
Number of mislabeled points from test data set 40
Actual value: 18.2 predicted value: 20.46967735957765
Number of mislabeled points from test data set 40
Actual value: 9.4 predicted value: 11.290281480791585
Number of mislabeled points from test data set 40
Actual value: 6.9 predicted value: 8.548238362449258
Number of mislabeled points from test data set 40
Actual value: 18.4 predicted value: 18.8210870184884
Number of mislabeled points from test data set 40
Actual value: 12.5 predicted value: 10.808040768772278
Number of mislabeled points from test data set 40
Actual value: 16.1 predicted value: 22.746302116319946
Number of mislabeled points from test data set 40
Actual value: 13.7 predicted value: 12.473453460280808
Number of mislabeled points from test data set 40
```

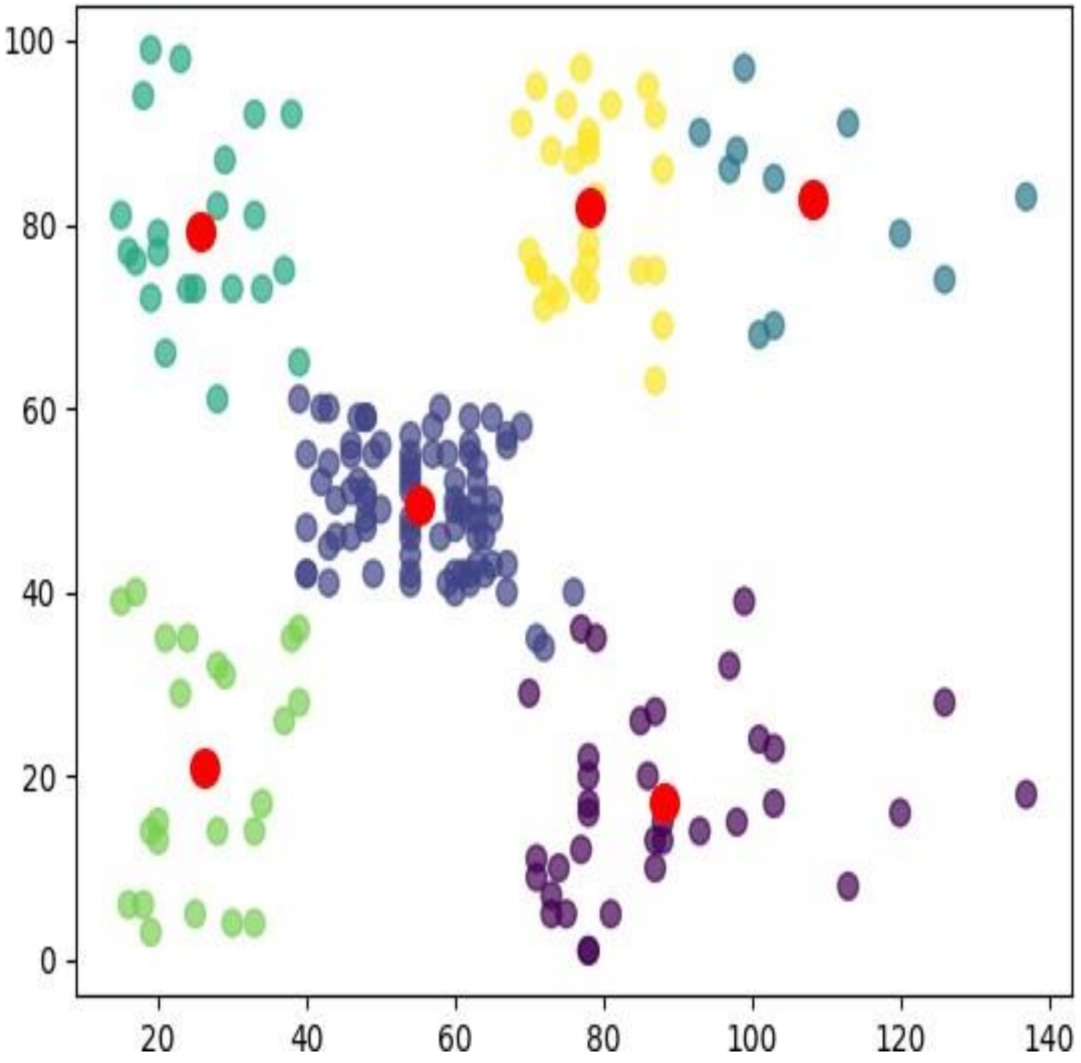
17. Write a program to implement K –Means Clustering Algorithm with k=6. Create a scatter plot to visualize the same.

CODE:

```
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

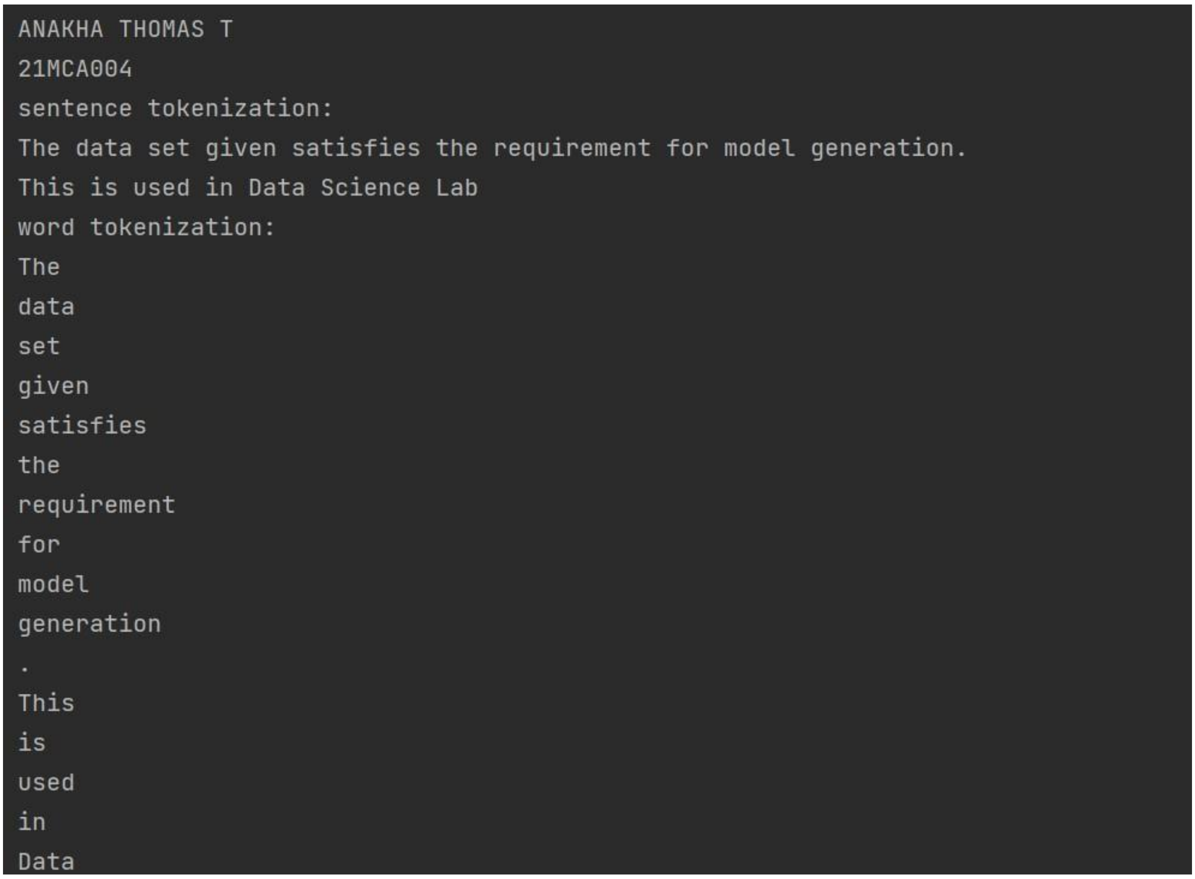
4) create n-grams for different values of n=2,4.

CODE:

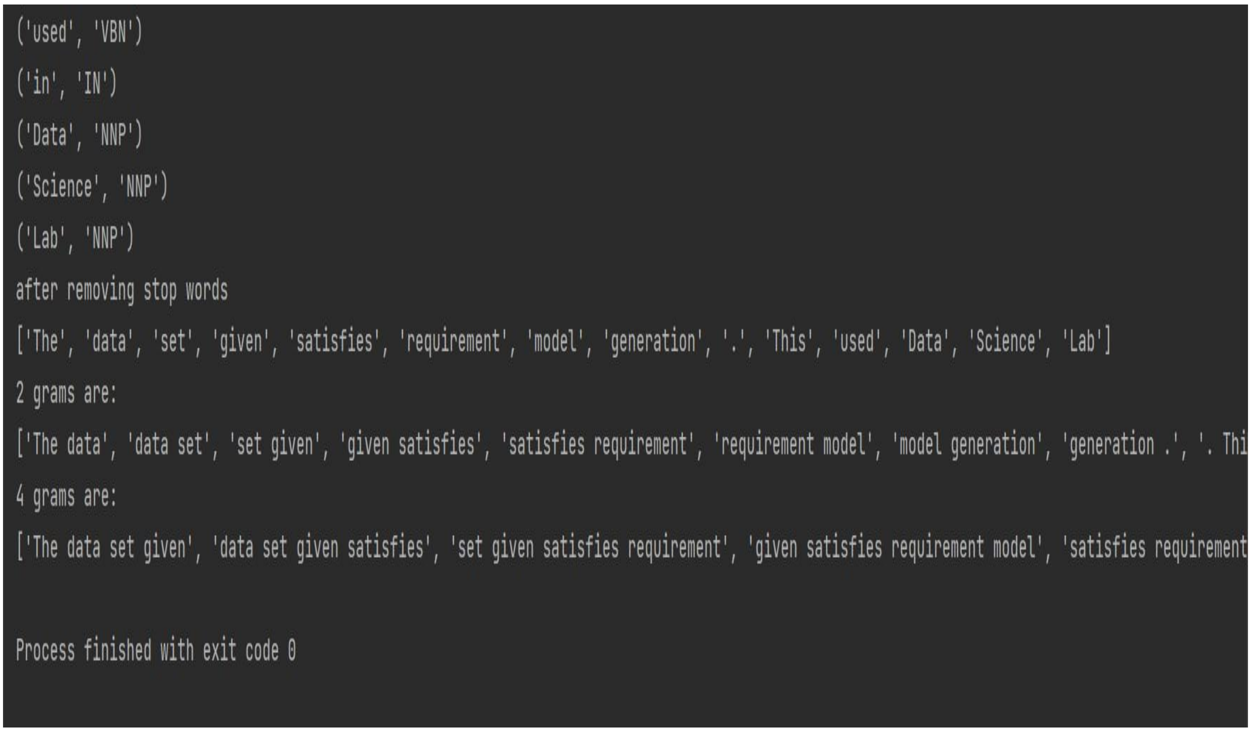
```
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import sent_tokenize, word_tokenize

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



```
ANAKHA THOMAS T
21MCA004
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
```



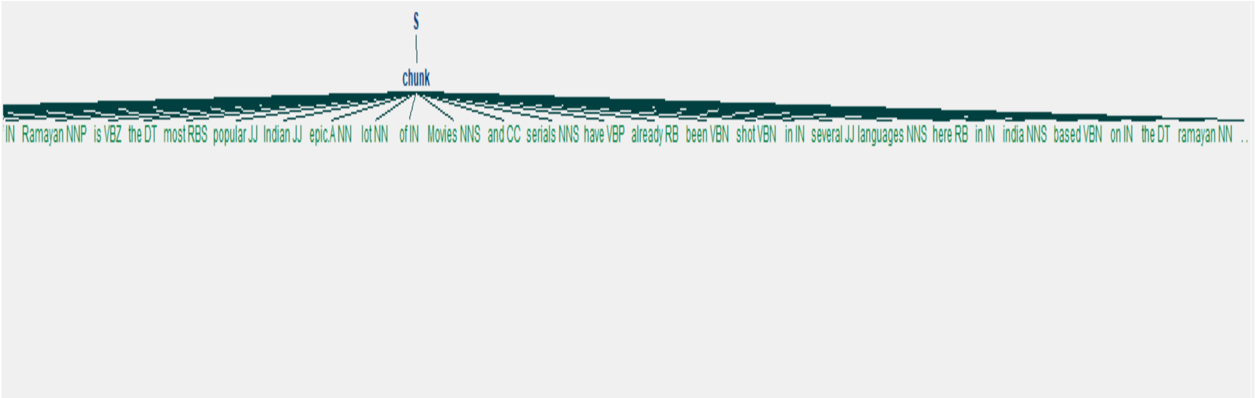
```
('used', 'VBN')
('in', 'IN')
('Data', 'NNP')
('Science', 'NNP')
('Lab', 'NNP')
after removing stop words
['The', 'data', 'set', 'given', 'satisfies', 'requirement', 'model', 'generation', '.', 'This', 'used', 'Data', 'Science', 'Lab']
2 grams are:
['The data', 'data set', 'set given', 'given satisfies',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

Process finished with exit code 0
```

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

CODE:

```
import nltk
nltk.download('averaged_perceptron_tagger')
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: {<.*>+ }"""
chunkParser=nltk.RegexpParser(chunkGram)
chunked=chunkParser.parse(tagged_words)
print(chunked)
chunked.draw()
```

OUTPUT

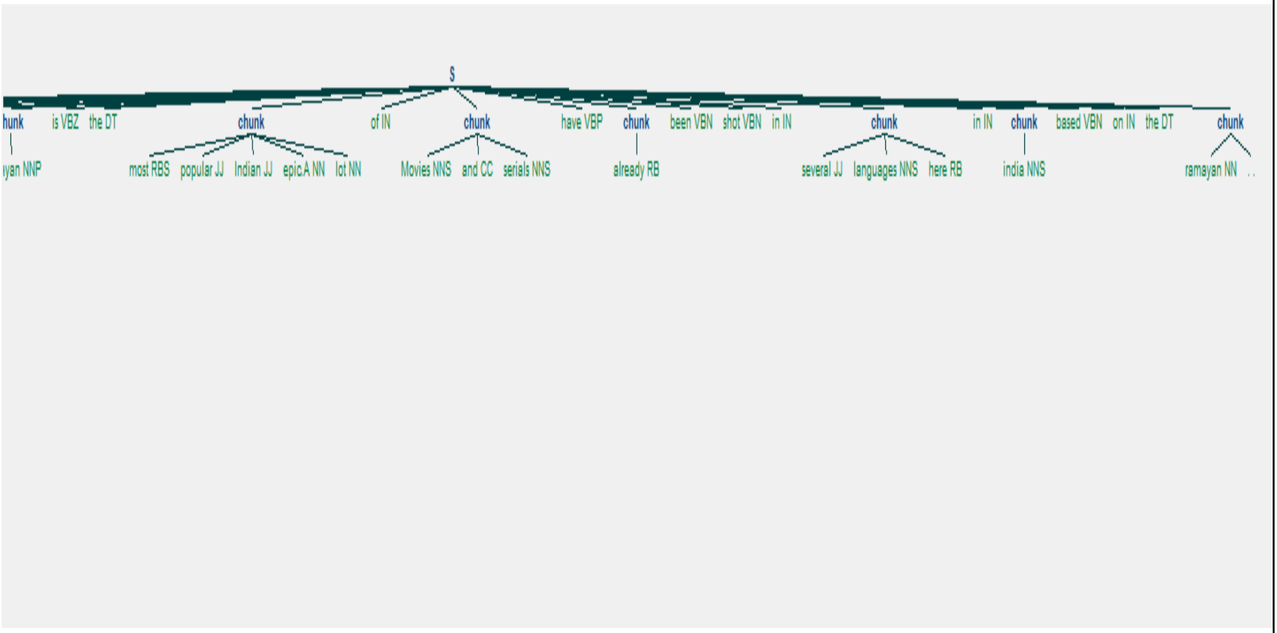

S
chunk
IN
Ramayan NNP
is VBZ
the DT
most RBS
popular JJ
Indian JJ
epic.A NN
lot NN
of IN
Movies NNS
and CC
serials NNS
have VBP
already RB
been VBN
shot VBN
in IN
several JJ
languages NNS
here RB
in IN
india NNS
based VBN
on IN
the DT
ramayan NN
. .

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

CODE:

```
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
hunk
is VBZ
the DT
chunk
of IN
chunk
have VBP
chunk
been VBN
shot VBN
in IN
chunk
in IN
chunk
based VBN
on IN
the DT
chunk
ıyan NNP
most RBS
popular JJ
Indian JJ
epic.A NN
lot NN
Movies NNS
and CC
serials NNS
already RB
several JJ
languages NNS
here RB
india NNS
ramayan NN
. .