20MCA241 DATA SCIENCE LAB

Lab Report SubmittedBy

ELIZABETH ANTONY

Reg. No.:AJC20MCA-2036

In Partial fulfillment for the Award of the Degree Of

MASTER OF COMPUTER APPLICATIONS (2 Year)
(MCA)
APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY

AMAL JYOTHI COLLEGE OF ENGINEERING
KANJIRAPPALLY

[Affiliated to APJ Abdul Kalam Technological University, Kerala. Approved by AICTE,
Accredited by NAAC with ‘A’ grade. Koovappally, Kanjirappally, Kottayam, Kerala – 686518]

2020-2022

DEPARTMENT OF COMPUTER APPLICATIONS

AMAL JYOTHI COLLEGE OF ENGINEERING

KANJIRAPPALLY

CERTIFICATE

This is to certify that the Lab report, "**20MCA241 DATA SCIENCE LAB"** is the bonafide work of **ELIZABETH ANTONY (Reg.No:AJC20MCA-2036)** in partial fulfillment of the requirements for the award of the Degree of Master of Computer Applications under APJ Abdul Kalam Technological University during the year 2021-22.

Ms. Nimmy Francis

Lab In-Charge

CONTENT

PROGRAM NO : 01 **Date:24/11/2021**

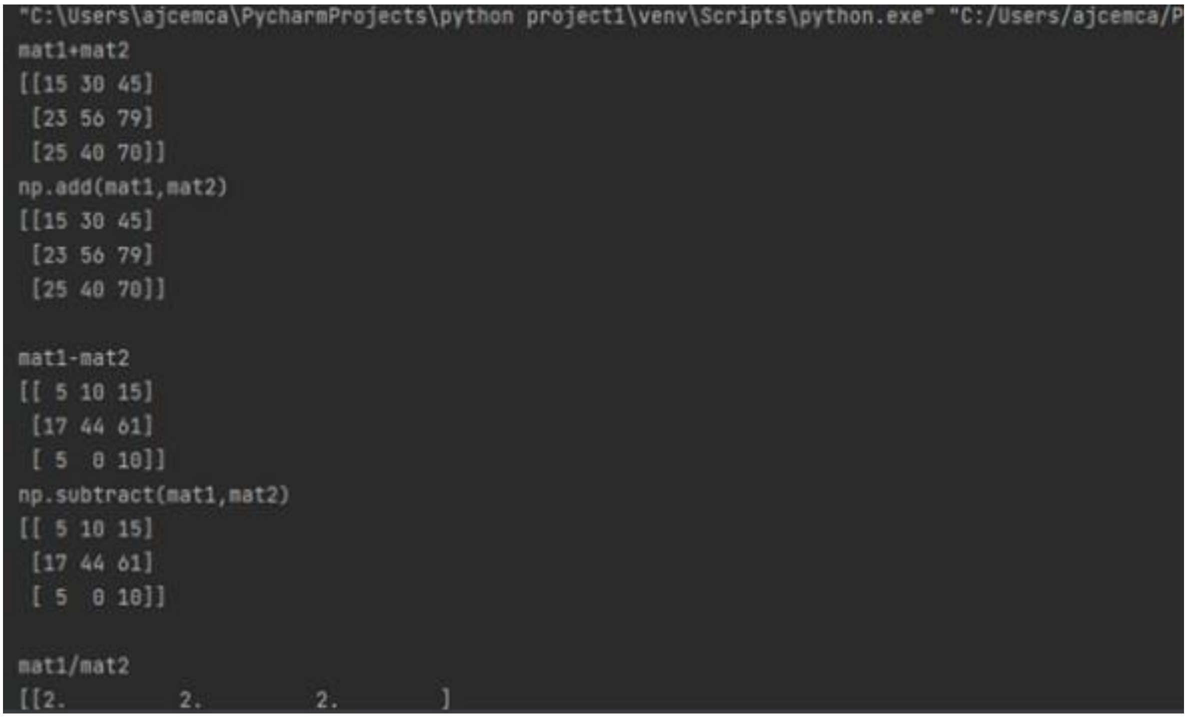
AIM : Perform all matrix operation using python

Program Code :

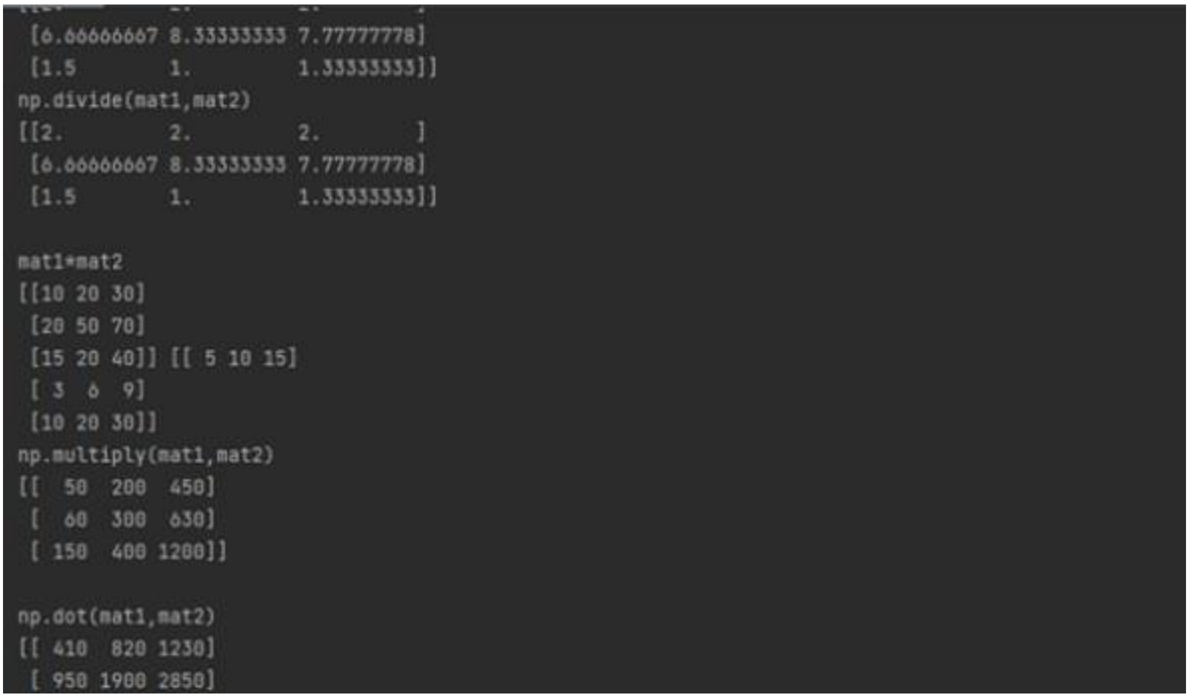
```
import numpy as np
mat1=np.array([[10,20,30],[
20,50,70],[15,20,40]])
mat2=np.array([[5,10,15],[3,
6,9],[10,20,30]])
print("mat1+mat2")
print(mat1+mat2)
print("np.add(mat1,mat2)")
print(np.add(mat1,mat2))
print()
print("mat1-mat2")
print(mat1-mat2)
print("np.subtract(mat1,mat1
)")
print(np.subtract(mat1,mat2)
print()
print("mat1/mat2")
print(mat1/mat2)
print("np.divide(mat1,mat2))
print(np.divide(mat1,mat2))
print()
print("mat1*mat2")
```

```
 print(mat1,mat2)
print("np.multiply(mat1,mat2
)")
 print(np.multiply(mat1,mat2
 ))
 print()
 print("np.dot(mat1,mat2)")
 print(np.dot(mat1,mat2))
 print("np.sqrt(mat1)")
 print(np.sqrt(mat1))
 print("np.sqrt(mat2)")
 print(np.sqrt(mat2))
```

<u>Output</u> :



```
"C:\Users\ajcemca\PycharmProjects\python project1\venv\Scripts\python.exe" "C:/Users/ajcemca/P
mat1+mat2
[[15 30 45]
 [23 56 79]
 [25 40 70]]
np.add(mat1,mat2)
[[15 30 45]
 [23 56 79]
 [25 40 70]]

mat1-mat2
[[ 5 10 15]
 [17 44 61]
 [ 5  0 10]]
np.subtract(mat1,mat2)
[[ 5 10 15]
 [17 44 61]
 [ 5  0 10]]

mat1/mat2
[[2.         2.         2.        ]
```



```
 [6.66666667 8.33333333 7.77777778]
 [1.5        1.         1.33333333]]
np.divide(mat1,mat2)
[[2.         2.         2.        ]
 [6.66666667 8.33333333 7.77777778]
 [1.5        1.         1.33333333]]

mat1*mat2
[[10 20 30]
 [20 50 70]
 [15 20 40]] [[ 5 10 15]
 [ 3  6  9]
 [10 20 30]]
np.multiply(mat1,mat2)
[[  50  200  450]
 [  60  300  630]
 [ 150  400 1200]]

np.dot(mat1,mat2)
[[ 410  820 1230]
 [ 950 1900 2850]
```

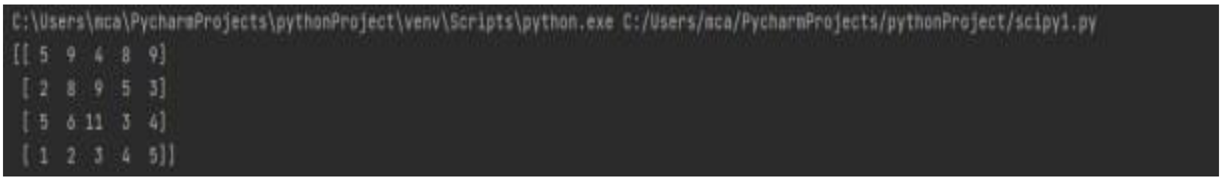
PROGRAM NO : 02 **Date :01/12/2021**

AIM : Program to perform SVD using python

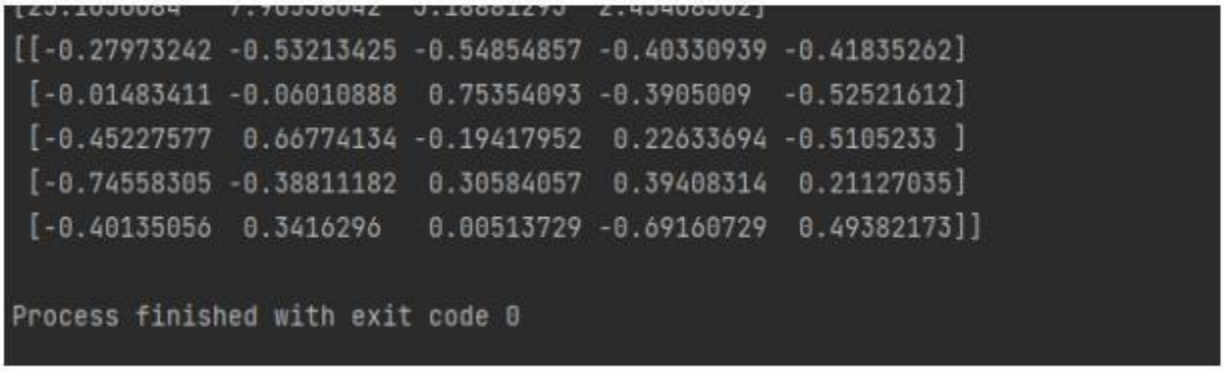
Program Code :

```
from numpy import array

from scipy.linalg import svd

B=array([[5,9,4,8,9],[2,8,9,5,3],[5,6,11,3,4],[1,2,3,4,5]])

print(B)

P,Q,R = svd(B)

print(P)

print(Q)

print(R)
```

Output :



```
C:\Users\mca\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/mca/PycharmProjects/pythonProject/scipy1.py
[[ 5  9  4  8  9]
 [ 2  8  9  5  3]
 [ 5  6 11  3  4]
 [ 1  2  3  4  5]]
```



```
[25.1036084   7.90538042  5.18881293  2.43408302]
[[-0.27973242 -0.53213425 -0.54854857 -0.40330939 -0.41835262]
 [-0.01483411 -0.06010888  0.75354093 -0.3905009  -0.52521612]
 [-0.45227577  0.66774134 -0.19417952  0.22633694 -0.5105233 ]
 [-0.74558305 -0.38811182  0.30584057  0.39408314  0.21127035]
 [-0.40135056  0.3416296   0.00513729 -0.69160729  0.49382173]]

Process finished with exit code 0
```

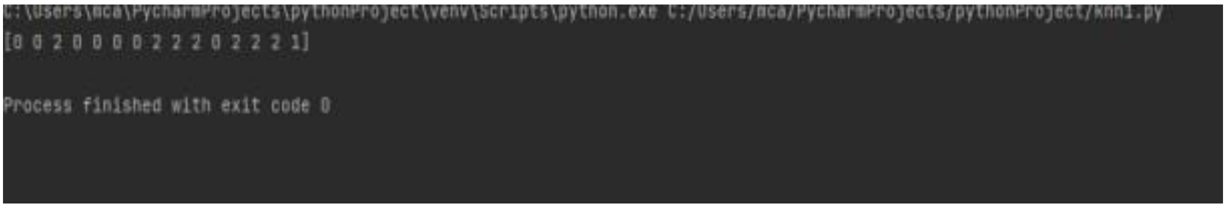
PROGRAM NO : 03 **Date :01/12/2021**

AIM :Program to implement k-NN Classification using any standard dataset available in the public domain and find the accuracy of the algorithm using in build function

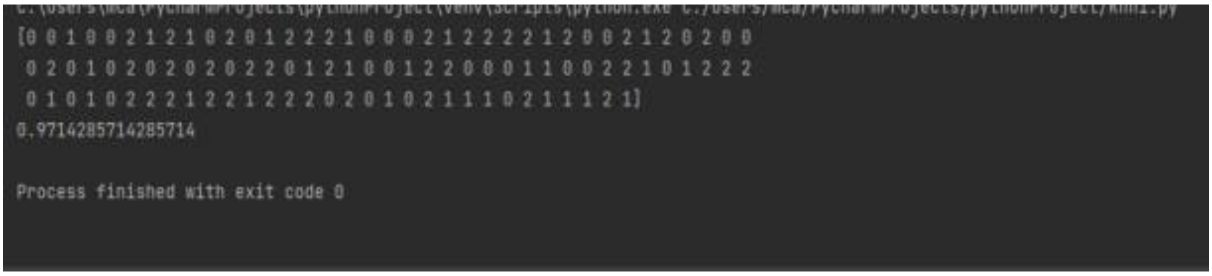
Program Code :

```
from sklearn.neighbors import

KNeighborsClassifier

from sklearn.model_selection import

train_test_split

from sklearn.datasets import load_iris

irisData=load_iris()

x=irisData.data

y=irisData.target

x_train,x_test,y_train,y_test=train_test_split(

x,y,test_size=0.1,random_state=45)

Knn=KNeighborsClassifier(n_neighbors=2)

Knn.fit(x_train,y_train)

print(Knn.predict(x_test))

w=Knn.predict(x_test)

z=accuracy_score(y_test,w)

print(z)
```

Output :



```
C:\Users\mca\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/mca/PycharmProjects/pythonProject/knn1.py
[0 0 2 0 0 0 0 2 2 2 0 2 2 2 1]

Process finished with exit code 0
```



```
C:\Users\mca\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/mca/PycharmProjects/pythonProject/knn1.py
[0 0 1 0 0 2 1 2 1 0 2 0 1 2 2 2 1 0 0 0 2 1 2 2 2 2 1 2 0 0 2 1 2 0 2 0 0
 0 2 0 1 0 2 0 2 0 2 0 2 2 0 1 2 1 0 0 1 2 2 0 0 0 1 1 0 0 2 2 1 0 1 2 2 2
 0 1 0 1 0 2 2 2 1 2 2 1 2 2 2 0 2 0 1 0 2 1 1 1 0 2 1 1 1 2 1]
0.9714285714285714

Process finished with exit code 0
```

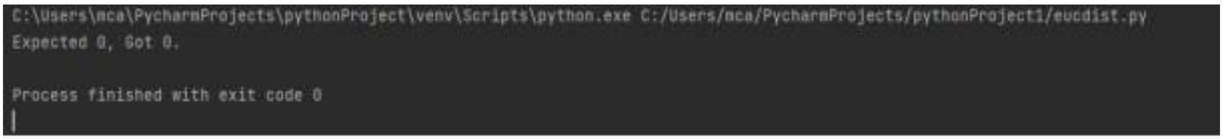
PROGRAM NO : 04 **Date :01/12/2021**

AIM :Program to implement k-NN Classification using any random dataset without using in-build functions.

Program Code :

```
from math import sqrt

def euclidean_distance(row1, row2):

distance = 0.0

for i in range(len(row1) -1):

distance += (row1[i] - row2[i]) **2

return sqrt(distance)

def get_neighbors(train, test_row,
num_neighbors):

distances = list()

for train_row in train:

dist = euclidean_distance(test_row, train_row)

distances.append((train_row, dist))

distances.sort(key=lambda tup: tup[1])

neighbors = list()

for i in range(num_neighbors):

neighbors.append(distances[i][0])

return neighbors

def predict_classification(train, test_row,
num_neighbors):

        neighbors = get_neighbors(train,

test_row, num_neighbors)

output_values = [row[-1] for row in neighbors]

prediction = max(set(output_values),

key=output_values.count)

return prediction

dataset = [[2.7810836, 2.550537003, 0],

[1.4645489372, 2.362125076, 0],

[3.396561688, 4.400293529, 0],

[1.38807019, 1.850220317, 0],

[3.06407232, 3.005305973, 1],

[7.627531214, 2.759262635, 1],

[5.332441248, 2.088626775, 1],

[6.922596716, 1.77106367, 1],

[8.675418651, -0.242068655, 1],

[7.673756466, 3.508563011, 1]]

prediction = predict_classification(dataset,

dataset[0], 5)

print('Expected %d, Got %d.' % (dataset[0][-1],

prediction))
```

Output :



```
C:\Users\mca\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/mca/PycharmProjects/pythonProject1/eucdist.py
Expected 0, Got 0.

Process finished with exit code 0
```

<u>PROGRAM NO</u> : 05 **Date :08/12/2021**

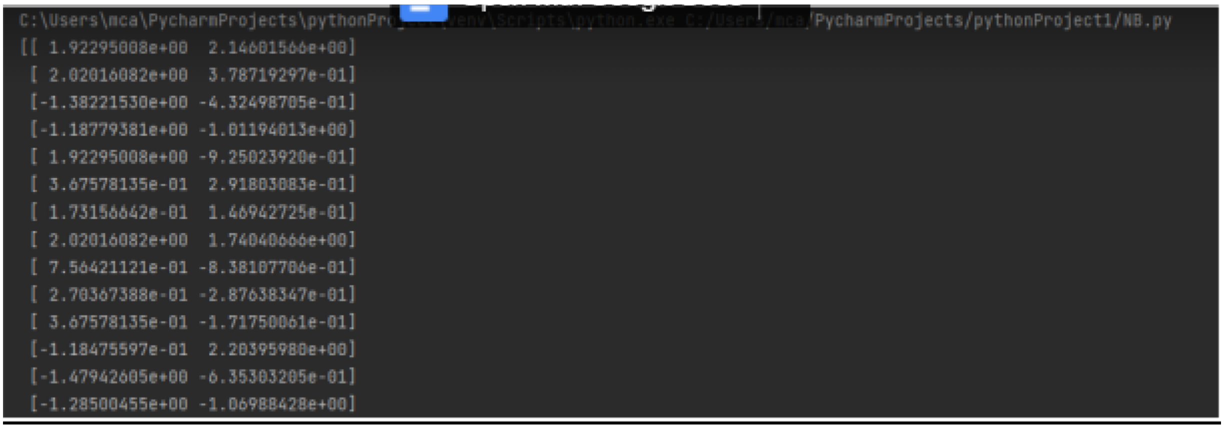
AIM : Program to implement Naïve Bayes Algorithm using any standard dataset available in the public domain and find the accuracy of the algorithm.

<u>Program Code</u> :

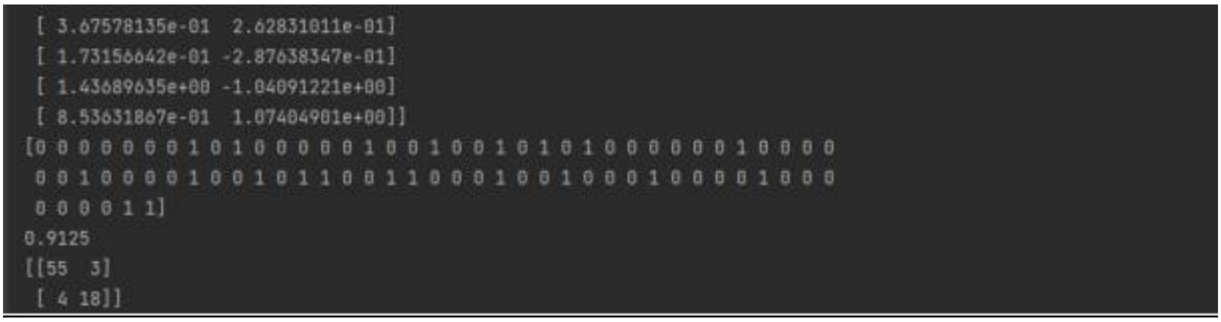
```
import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

dataset = pd.read_csv('csv.txt')

X = dataset.iloc[:, [2, 3]].values

Y = dataset.iloc[:, -1].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, random_state = 0)

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

X_train = sc.fit_transform(X_train)

X_test = sc.transform(X_test)

print(X_train)

print(X_test)

from sklearn.naive_bayes import GaussianNB
```

```
classifier = GaussianNB()
classifier.fit(X_train, Y_train)

Y_pred = classifier.predict(X_test)

print(Y_pred)

from sklearn.metrics import confusion_matrix,

accuracy_score

ac = accuracy_score(Y_test, Y_pred)

cm = confusion_matrix(Y_test, Y_pred)

print(ac)

print(cm)
```

<u>Output</u> :



```
C:\Users\mca\PycharmProjects\pythonProject1\venv\Scripts\python.exe C:/Users/mca/PycharmProjects/pythonProject1/NB.py
[[ 1.92295008e+00  2.14601566e+00]
 [ 2.02016082e+00  3.78719297e-01]
 [-1.38221530e+00 -4.32498705e-01]
 [-1.18779381e+00 -1.01194013e+00]
 [ 1.92295008e+00 -9.25023920e-01]
 [ 3.67578135e-01  2.91803083e-01]
 [ 1.73156642e-01  1.46942725e-01]
 [ 2.02016082e+00  1.74040666e+00]
 [ 7.56421121e-01 -8.38107706e-01]
 [ 2.70367388e-01 -2.87638347e-01]
 [ 3.67578135e-01 -1.71750061e-01]
 [-1.18475597e-01  2.20395980e+00]
 [-1.47942605e+00 -6.35303205e-01]
 [-1.28500455e+00 -1.06988428e+00]
```



```
 [ 3.67578135e-01  2.62831011e-01]
 [ 1.73156642e-01 -2.87638347e-01]
 [ 1.43689635e+00 -1.04091221e+00]
 [ 8.53631867e-01  1.07404901e+00]]
[0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 0 1 0 0 1 0 1 0 1 0 0 0 0 0 0 1 0 0 0 0
 0 0 1 0 0 0 0 1 0 0 1 0 1 1 0 0 1 1 0 0 0 1 0 0 1 0 0 0 1 0 0 0 0 1 0 0 0
 0 0 0 0 1 1]
0.9125
[[55  3]
 [ 4 18]]
```

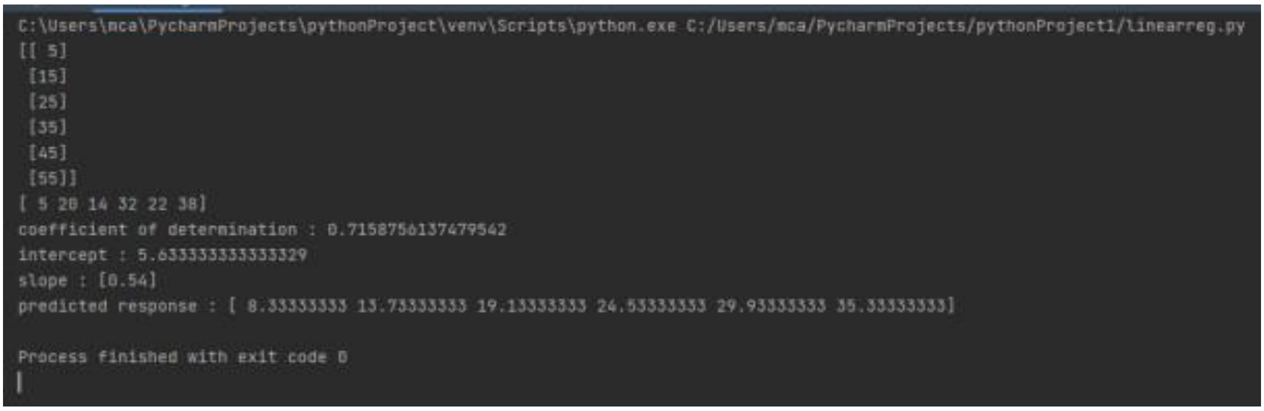
PROGRAM NO : 06 **Date : 08/01/2022**

AIM : Program to implement linear and multiple regression techniques using any standard dataset available in the public domain.

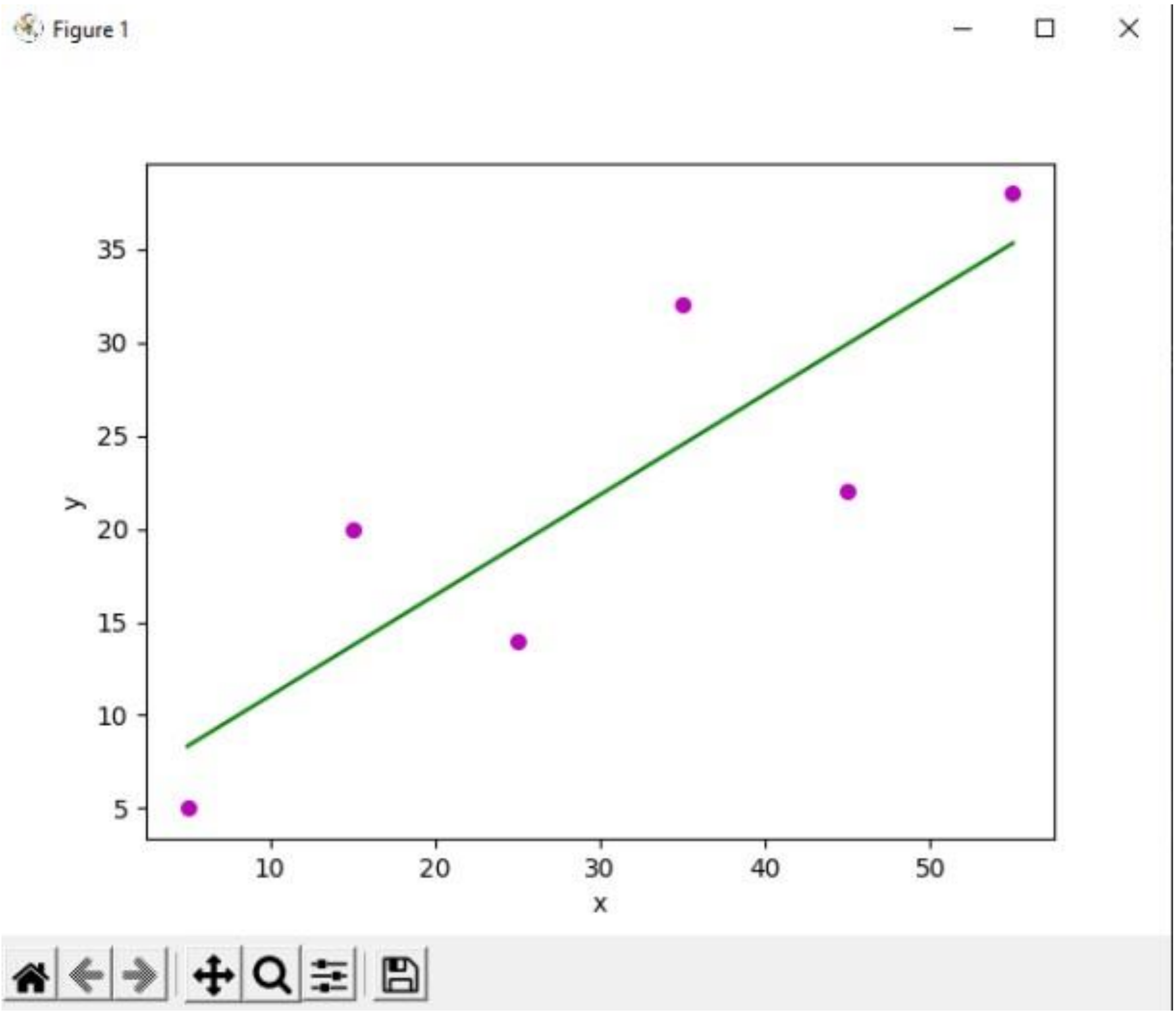
Program Code :

```
import numpy as np

from sklearn.linear_model import LinearRegression
x = np.array([5,15,25,35,45,55]).reshape((-1, 1))
y= np.array([5,20,14,32,22,38])
print(x)
print(y)
model = LinearRegression()
model.fit(x, y)
r_sq = model.score(x, y)
print('coefficient of determination :', r_sq)
print('intercept :' ,model.intercept_)
print('slope :' ,model.coef_)
y_pred = model.predict(x)
print('predicted response :', y_pred )
plt.scatter(x, y, color="m",
marker="o", s=30)
plt.plot(x, y_pred, color="g")
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```

<u>Output</u> :



```
C:\Users\mca\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/mca/PycharmProjects/pythonProject1/linearreg.py
[[ 5]
 [15]
 [25]
 [35]
 [45]
 [55]]
[ 5 20 14 32 22 38]
coefficient of determination : 0.7158756137479542
intercept : 5.633333333333329
slope : [0.54]
predicted response : [ 8.33333333 13.73333333 19.13333333 24.53333333 29.93333333 35.33333333]

Process finished with exit code 0
```



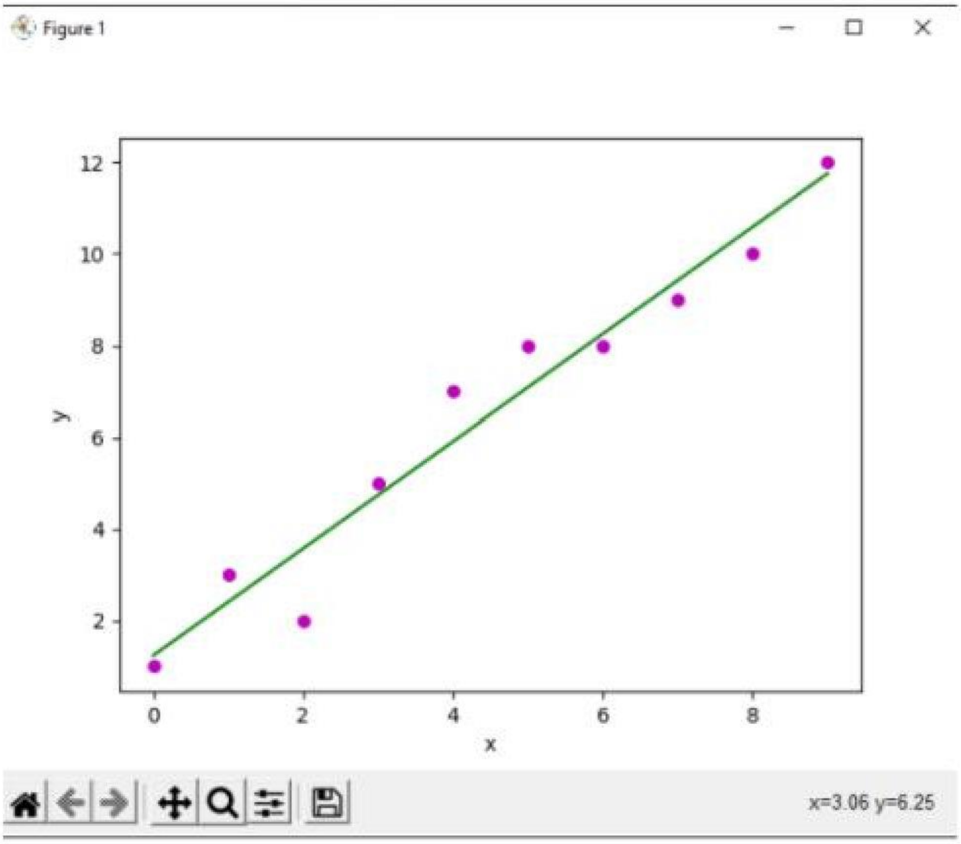
PROGRAM NO : 07 **Date :15/01/2022**

AIM : Program to implement Linear and Multiple regression techniques using any standard dataset available in public domain and evaluate its performance.

Program Code :

```
import numpy as np
import matplotlib.pyplot as plt
def estimate_coef(x, y):
n = np.size(x)
m_x = np.mean(x)
m_y = np.mean(y)
SS_xy = np.sum(y * x) - n * m_y * m_x
SS_xx = np.sum(x * x) - n * m_x * m_x
b_1 = SS_xy / SS_xx
b_0 = m_y - b_1 * m_x
b_1 = SS_xy / SS_xx
b_0 = m_y - b_1 * m_x
return(b_0, b_1)
def plot_regression_line(x, y, b):
plt.scatter(x, y, color="m",
marker="o", s=30)
y_pred = b[0] + b[1] * x
plt.plot(x, y_pred, color="g")
plt.xlabel('x')
plt.ylabel('y')
plt.show()
def main():
x = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
y = np.array([1, 3, 2, 5, 7, 8, 9, 10, 12])
b = estimate_coef(x, y)
print("Estimated coefficients:\nb_0 = {} \
\nb_1 = {}".format(b[0], b[1]))
plot_regression_line(x, y, b)

if__name__== "_main_":
main()
```

Output :



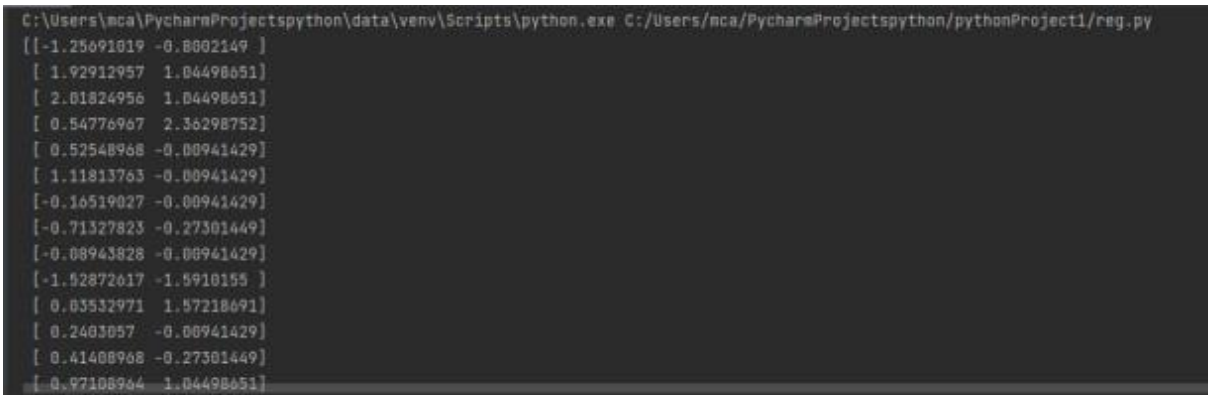
PROGRAM NO : 08 **Date :15/01/2022**

AIM : Program to implement Linear and Multiple regression techniques using cars dataset available in public domain and evaluate its performance.

Program Code :

```
import pandas
df = pandas.read_csv("cars.csv")
x = df[['Weight', 'Volume']]
y = df['CO2']
#splitting
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,
random_state=0)
# feature scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)
print(x_train)
print(x_test)
from sklearn import linear_model
regr = linear_model.LinearRegression()
regr.fit(x ,y)
predictedCO2 = regr.predict([[2300, 1300]])

print(predictedCO2)
from sklearn.metrics import accuracy_score
ac=accuracy_score(y_test,y_test)
print(ac)
```

Output:



```
C:\Users\mca\PycharmProjectspython\data\venv\Scripts\python.exe C:/Users/mca/PycharmProjectspython/pythonProject1/reg.py
[[-1.25691019 -0.8002149 ]
 [ 1.92912957  1.04498651]
 [ 2.01824956  1.04498651]
 [ 0.54776967  2.36298752]
 [ 0.52548968 -0.00941429]
 [ 1.11813763 -0.00941429]
 [-0.16519027 -0.00941429]
 [-0.71327823 -0.27301449]
 [-0.08943828 -0.00941429]
 [-1.52872617 -1.5910155 ]
 [ 0.03532971  1.57218691]
 [ 0.2403057  -0.00941429]
 [ 0.41408968 -0.27301449]
 [ 0.97108964  1.04498651]
```

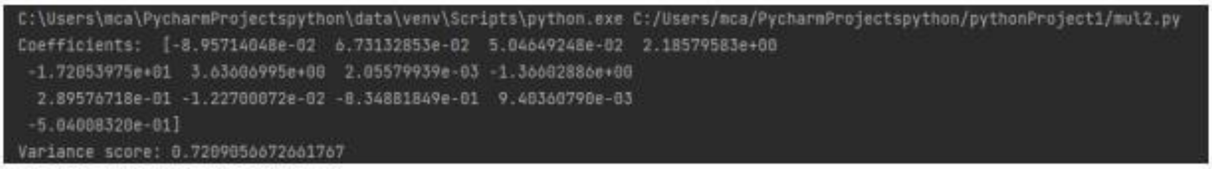
PROGRAM NO : 09 **Date :15/01/2022**

AIM : Program to implement multiple linear regression techniques using Boston dataset available in the public domain and evaluate its performance and plotting graph.

Program Code :

```
import matplotlib.pyplot as plt
#import numpy as np
from sklearn import datasets, linear_model, metrics
boston = datasets.load_boston(return_X_y=False)
X = boston.data
y = boston.target
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4,
random_state=1)
reg = linear_model.LinearRegression()
reg.fit(X_train, y_train)
print('Coefficients: ', reg.coef_)
print('Variance score: {}'.format(reg.score(X_test, y_test)))
```

Output:



```
C:\Users\mca\PycharmProjectspython\data\venv\Scripts\python.exe C:/Users/mca/PycharmProjectspython/pythonProject1/mul2.py
Coefficients:  [-8.95714048e-02  6.73132853e-02  5.04649248e-02  2.18579583e+00
 -1.72053975e+01  3.63606995e+00  2.05579939e-03 -1.36602886e+00
  2.89576718e-01 -1.22700072e-02 -8.34881849e-01  9.40360790e-03
 -5.04008320e-01]
Variance score: 0.7209056672661767
```

PROGRAM NO : 10 **Date : 22/12/2021**

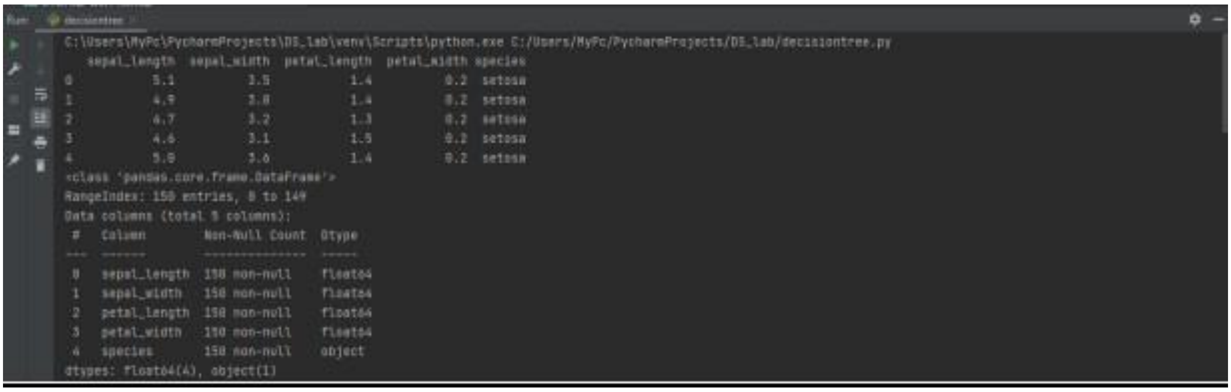
AIM : Program to implement decision tree using any standard dataset available in the public domain and find the accuracy of the algorithm.

Program Code :

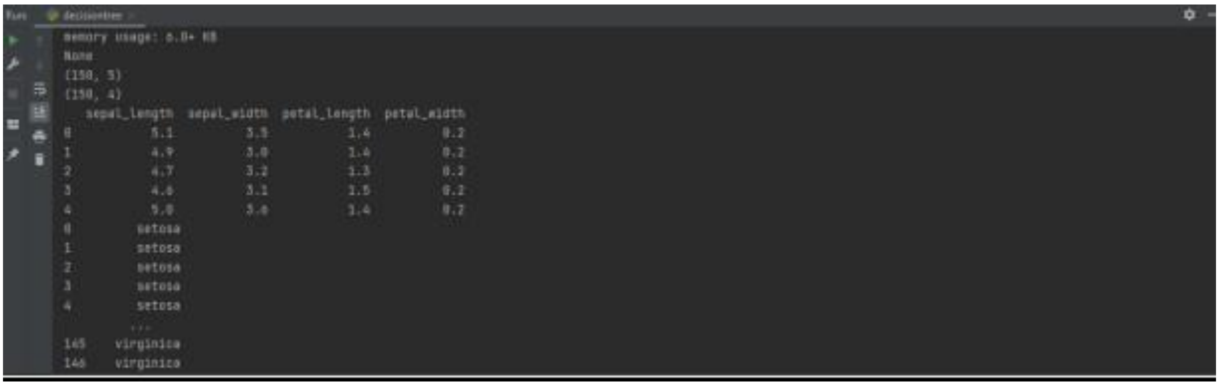
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report,
confusion_matrix
from sklearn.tree import plot_tree
df = sns.load_dataset('iris')
print(df.head())
print(df.info())
df.isnull().any()
print(df.shape)
sns.pairplot(data=df, hue = 'species')
plt.savefig("pne.png")
sns.heatmap(df.corr())
plt.savefig("one.png")
target = df['species']
df1 = df.copy()
df1 = df1.drop('species', axis =1)
print(df1.shape)
print(df1.head())
X = df1
print(target)
le = LabelEncoder()
target = le.fit_transform(target)
print(target)
y = target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size
= 0.2, random_state = 42)
print("Training split input- ", X_train.shape)
print("Testing split input- ", X_test.shape)

dtree = DecisionTreeClassifier()
dtree.fit(X_train,y_train)
print('Decision Tree Classifier Created')
y_pred = dtree.predict(X_test)
```

```
print("Classification report - \n",)
classification_report(y_test,y_pred))
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(5,5))
sns.heatmap(data=cm,linewidths=.5, annot=True,square = True,
cmap = 'Blues')
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
all_sample_title = 'Accuracy Score:
{0}'.format(dtree.score(X_test, y_test))
plt.title(all_sample_title, size = 15)
plt.savefig("two.png")
plt.figure(figsize = (20,20))
dec_tree = plot_tree(decision_tree=dtree,
feature_names=df1.columns,
class_names=["setosa", "vercicolor", "verginica"] ,
filled = True , precision = 4, rounded = True)
plt.savefig("three.png")
```

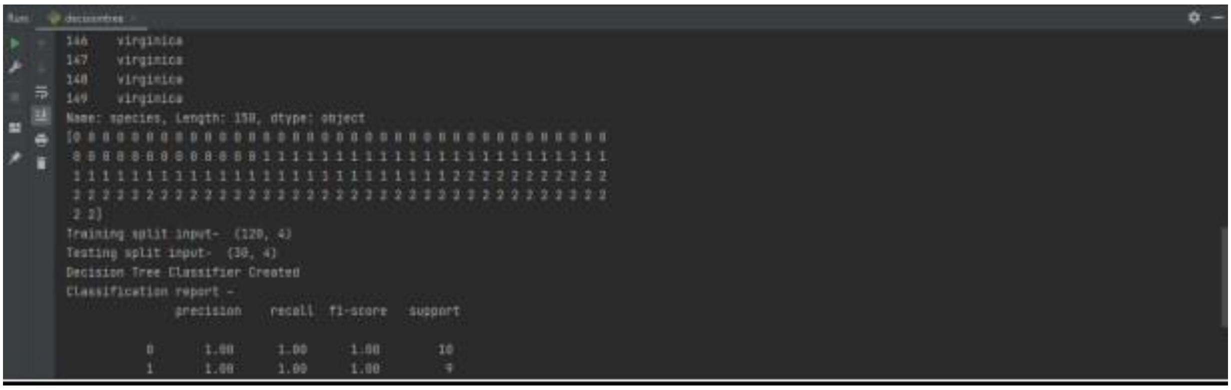
Output:



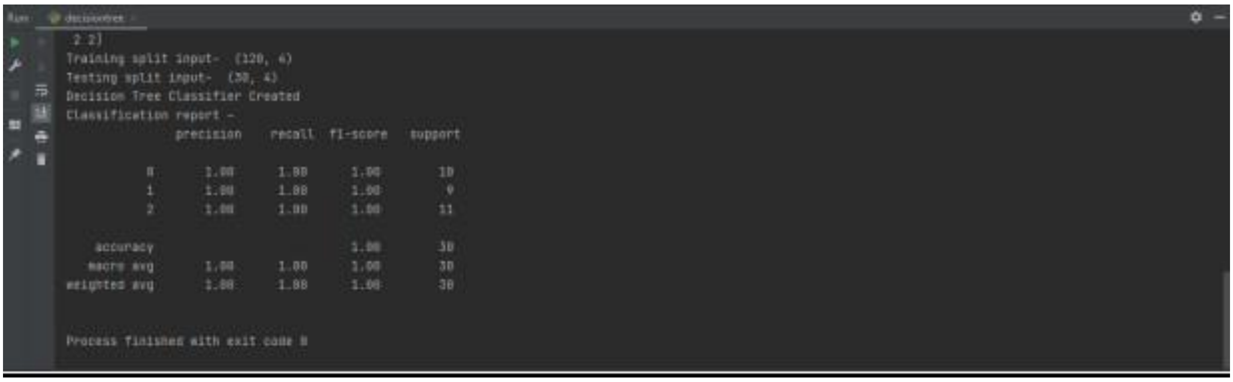
```
C:\Users\MyPc\PycharmProjects\DS_lab\venv\Scripts\python.exe C:/Users/MyPc/PycharmProjects/DS_lab/decisiontree.py
   sepal_length  sepal_width  petal_length  petal_width species
0           5.1          3.5           1.4          0.2  setosa
1           4.9          3.0           1.4          0.2  setosa
2           4.7          3.2           1.3          0.2  setosa
3           4.6          3.1           1.5          0.2  setosa
4           5.0          3.6           1.4          0.2  setosa
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    object
dtypes: float64(4), object(1)
```



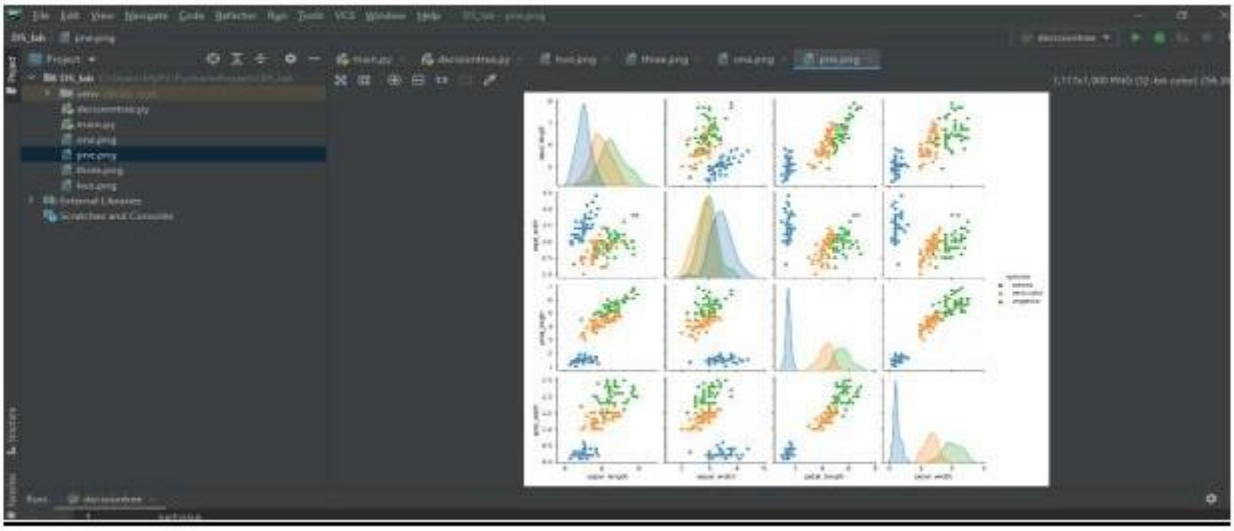
```
memory usage: 6.0+ KB
None
(150, 5)
(150, 4)
   sepal_length  sepal_width  petal_length  petal_width
0           5.1          3.5           1.4          0.2
1           4.9          3.0           1.4          0.2
2           4.7          3.2           1.3          0.2
3           4.6          3.1           1.5          0.2
4           5.0          3.6           1.4          0.2
0         setosa
1         setosa
2         setosa
3         setosa
4         setosa
         ...
145    virginica
146    virginica
```

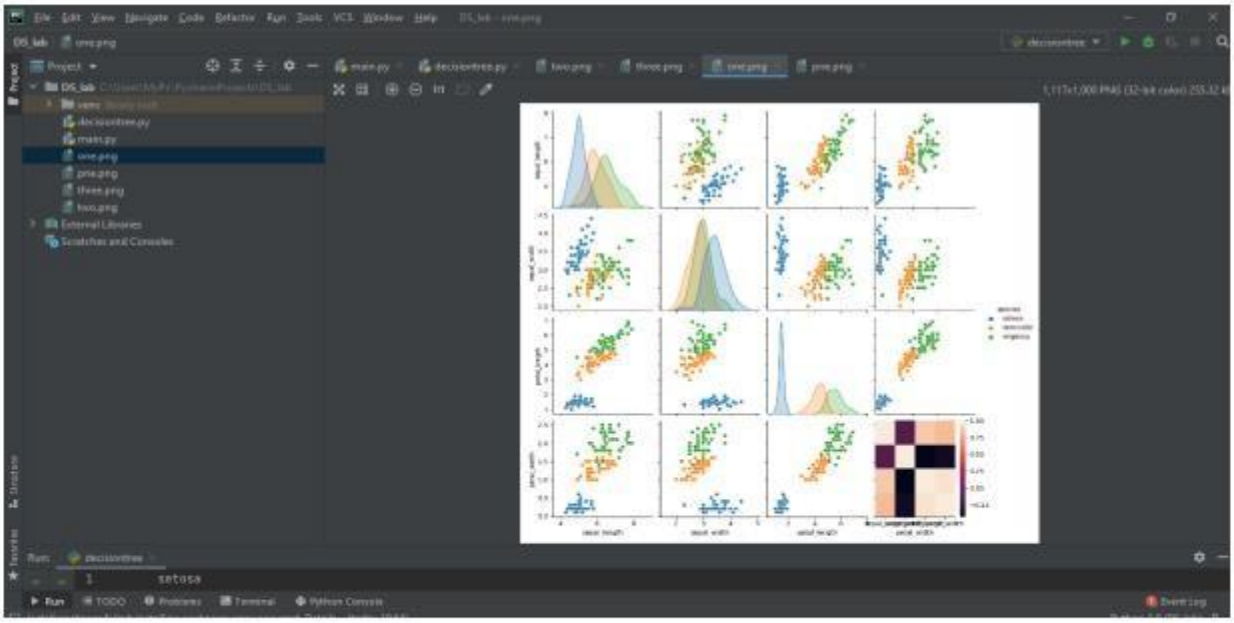


```
146    virginica
147    virginica
148    virginica
149    virginica
Name: species, Length: 150, dtype: object
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2]
Training split input-  (120, 4)
Testing split input-  (30, 4)
Decision Tree Classifier Created
Classification report -
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      1.00      1.00         9
```

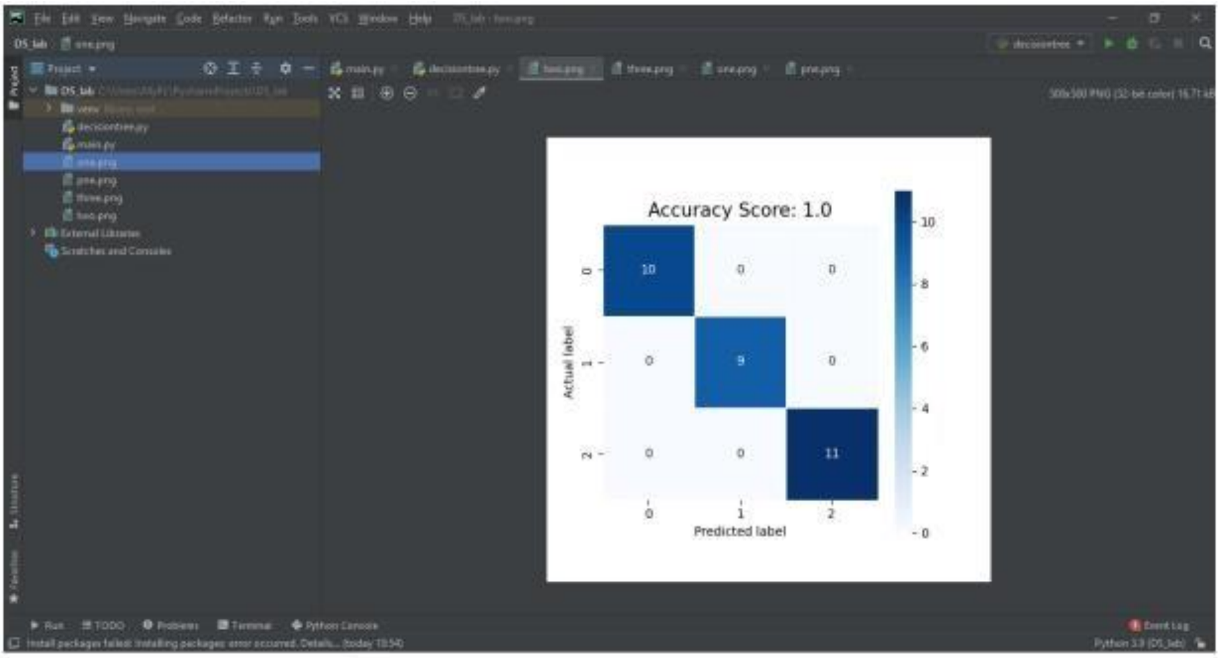


```
 2 2]
Training split input-  (120, 4)
Testing split input-  (30, 4)
Decision Tree Classifier Created
Classification report -
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      1.00      1.00         9
           2       1.00      1.00      1.00        11

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30


Process finished with exit code 0
```



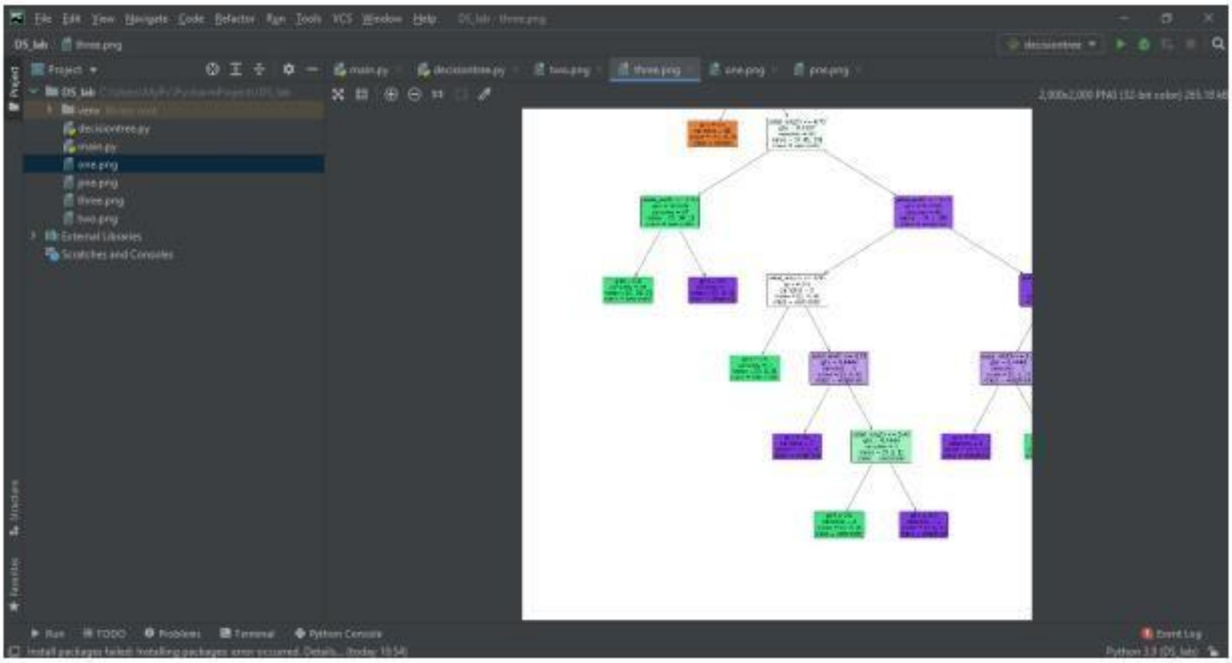


```
1        setosa
```



	0	1	2
0	10	0	0
1	0	9	0
2	0	0	11

```
Install packages failed: Installing packages: error occurred. Details... (today 10:54)
```



Install packages failed: Installing packages: error occurred. Details... (today 10:54)

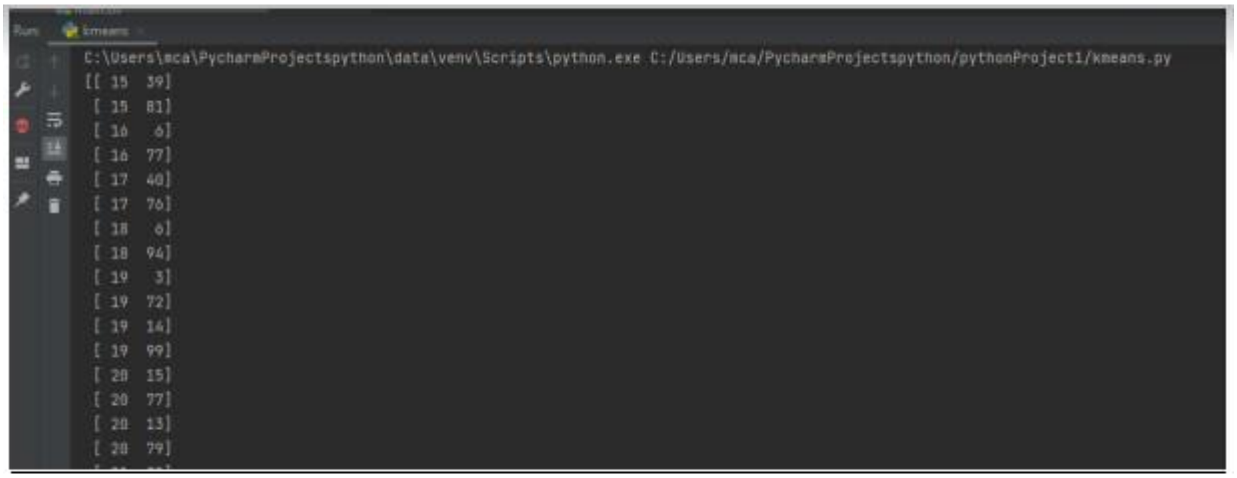
PROGRAM NO : 11 **Date : 05/01/2022**

AIM : Program to implement K-Means clustering technique using any standard dataset available in the public domain.

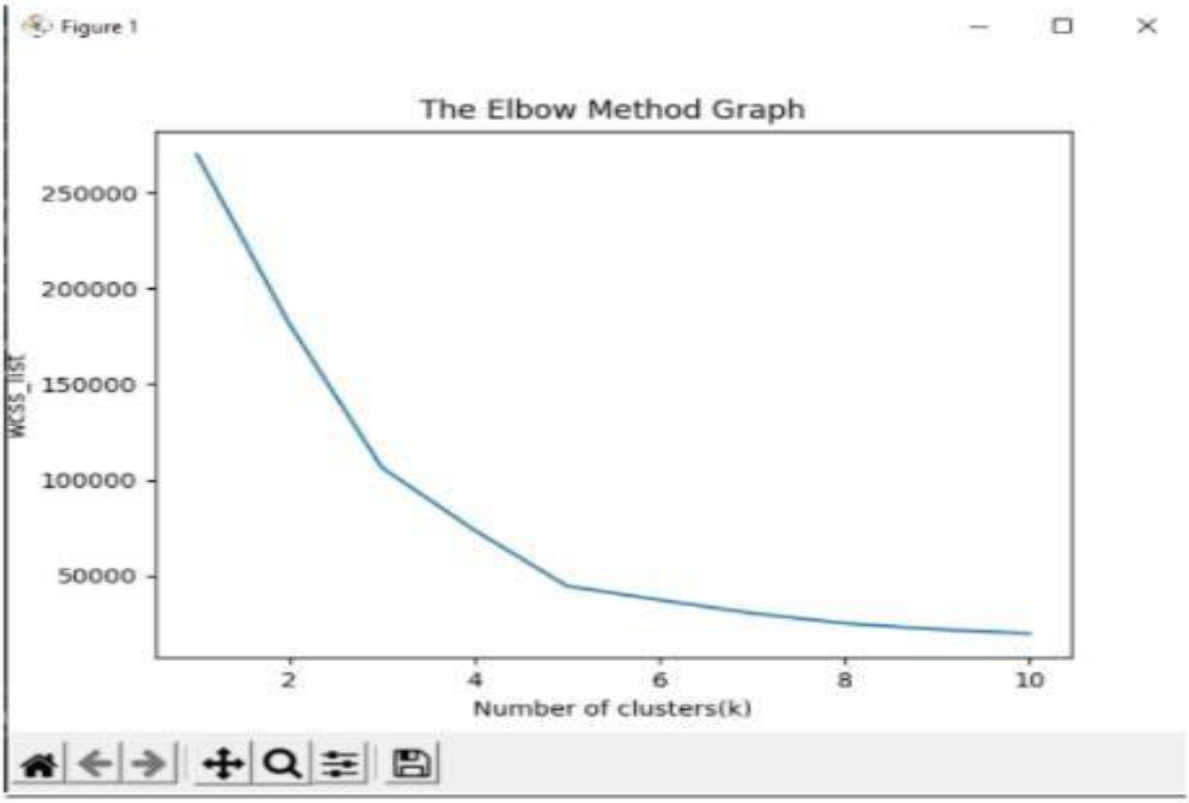
Program Code :

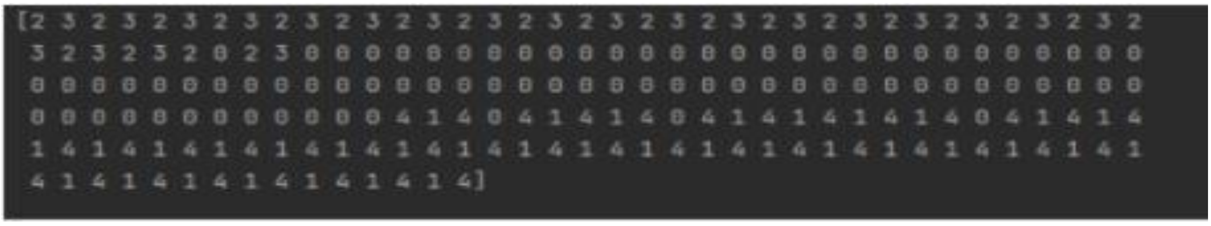
```
import numpy as nm
import matplotlib.pyplot as mtp
import pandas as pd
dataset = pd.read_csv('Mall_Customers.csv')
x = dataset.iloc[:, [3, 4]].values
print(x)
from sklearn.cluster import KMeans
wcss_list = []
for i in range(1, 11):
kmeans = KMeans(n_clusters=i, init='k-means++',
random_state=42)
kmeans.fit(x)
wcss_list.append(kmeans.inertia_)
mtp.plot(range(1, 11), wcss_list)
mtp.title('The Elbow Method Graph')
mtp.xlabel('Number of clusters(k)')
mtp.ylabel('wcss_list')
mtp.show()
kmeans = KMeans(n_clusters=5, init='k-means++',
random_state=42)
y_predict = kmeans.fit_predict(x)
print(y_predict)
mtp.scatter(x[y_predict == 0, 0], x[y_predict == 0, 1], s =
100, c = 'blue', label = 'Cluster0')
mtp.scatter(x[y_predict == 1, 0], x[y_predict == 1, 1], s =
100, c = 'green', label = 'Cluster1')
mtp.scatter(x[y_predict == 2, 0], x[y_predict == 2, 1], s =
100, c = 'red', label = 'Cluster2')
mtp.scatter(x[y_predict == 3, 0], x[y_predict == 3, 1], s =
100, c = 'cyan', label = 'Cluster3')
mtp.scatter(x[y_predict == 4, 0], x[y_predict == 4, 1], s =
100, c = 'magenta', label = 'Cluster4')
mtp.scatter(kmeans.cluster_centers_[:, 0],
kmeans.cluster_centers_[:, 1], s = 300, c = 'yellow')
mtp.title('Clusters of customers')
mtp.xlabel('Annual Income (k$)')
mtp.ylabel('Spending Score (1-100)')
mtp.legend()
mtp.show()
```

Output:



```
C:\Users\mca\PycharmProjectspython\data\venv\Scripts\python.exe C:/Users/mca/PycharmProjectspython/pythonProject1/kmeans.py
[[ 15  39]
 [ 15  81]
 [ 16   6]
 [ 16  77]
 [ 17  40]
 [ 17  76]
 [ 18   6]
 [ 18  94]
 [ 19   3]
 [ 19  72]
 [ 19  14]
 [ 19  99]
 [ 20  15]
 [ 20  77]
 [ 20  13]
 [ 20  79]
```





```
[2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2
 3 2 3 2 3 2 0 2 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 4 1 4 0 4 1 4 1 4 0 4 1 4 1 4 1 4 1 4 0 4 1 4 1 4
 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1
 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4]
```

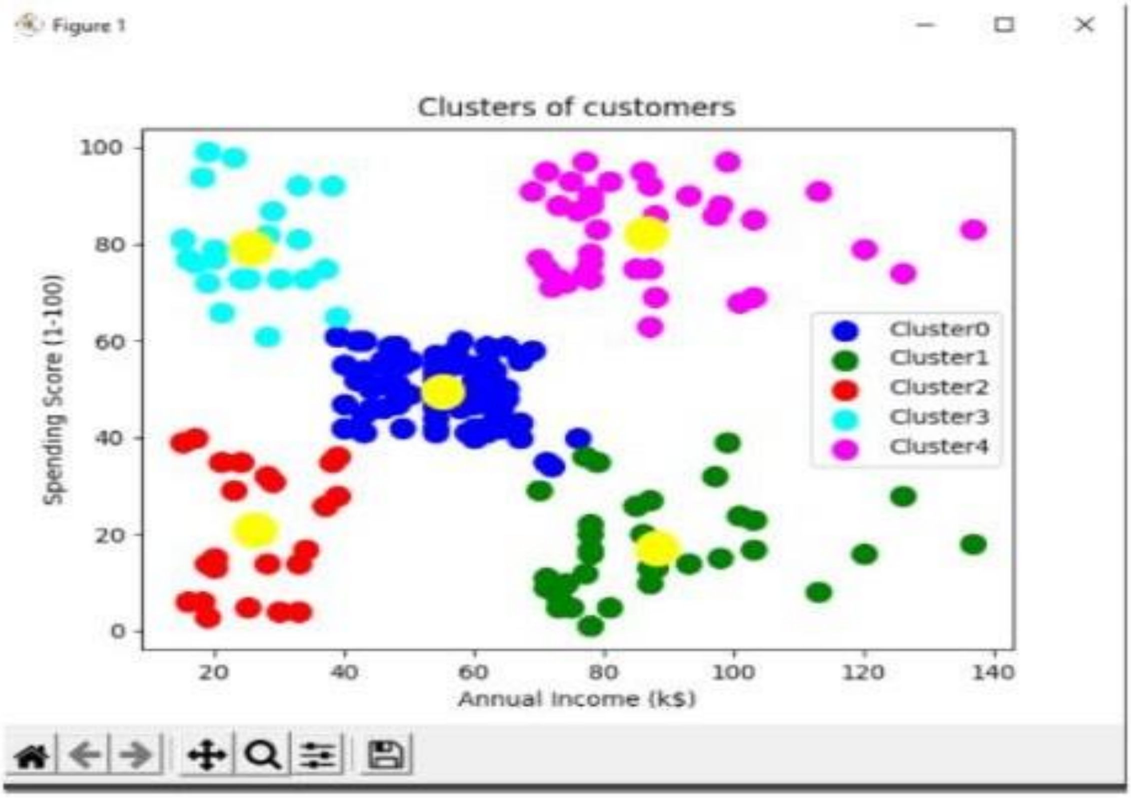


Figure 1
Clusters of customers
Spending Score (1-100)
0
20
40
60
80
100
Annual Income (k$)
20
40
60
80
100
120
140
Cluster0
Cluster1
Cluster2
Cluster3
Cluster4

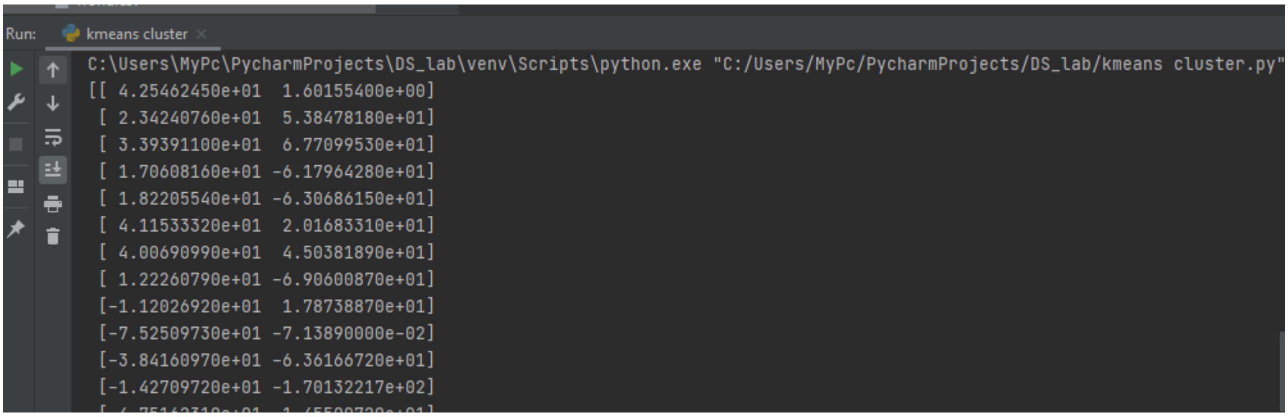
PROGRAM NO : 12 **Date : 05/01/2022**

AIM : Program to implement K-Means clustering technique using any standard dataset available in the public domain.

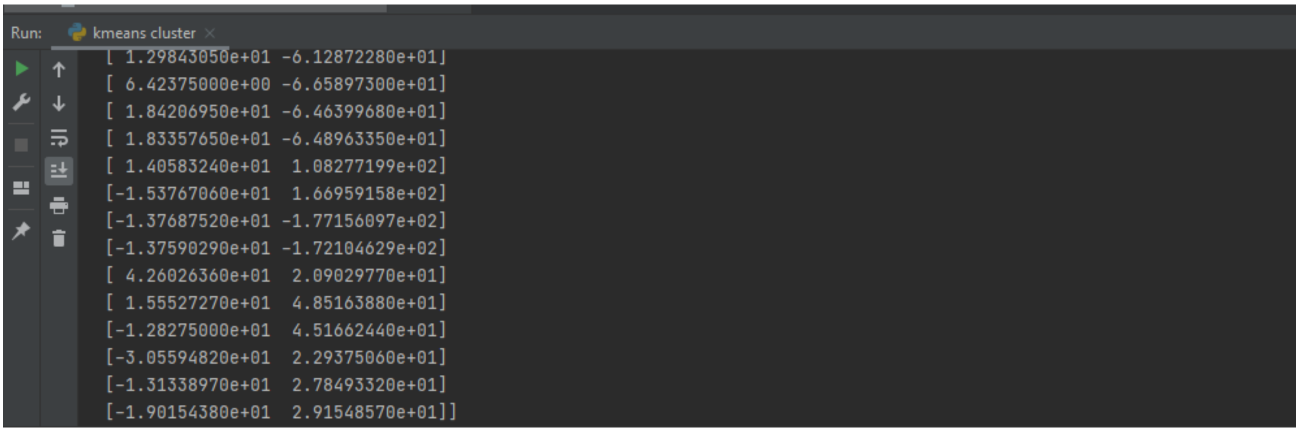
Program Code :

```
import numpy as np
import matplotlib.pyplot as mtp
import pandas as pd
dataset=pd.read_csv('world_country_and_usa_states_latitude_and_longitude_values.csv')
x=dataset.iloc[:,[1,2]].values
print(x)
from sklearn.cluster import KMeans
wcss_list = []
for i in range(1, 11):
        kmeans = KMeans(n_clusters=i, init='k-means++')
        kmeans.fit(x)
        wcss_list.append(kmeans.inertia_)
mtp.plot(range(1,11), wcss_list)
mtp.title('The elbow method Graph')
mtp.xlabel('Number of clusters (k)')
mtp.ylabel('wcss_list')
mtp.show()
kmeans = KMeans(n_clusters=3,init='k-means++',random_state=42)
y_predict=kmeans.fit_predict(x)
print(y_predict)
mtp.scatter(x[y_predict == 0,0], x[y_predict ==0,1], s=100, c='blue', label='Cluster0')
mtp.scatter(x[y_predict == 1,0], x[y_predict ==1,1], s=100, c='green', label= 'Cluster1')
mtp.scatter(x[y_predict == 2,0], x[y_predict ==2,1], s=100, c='red', label= 'Cluster2')
mtp.scatter(kmeans.cluster_centers_[:,0],kmeans.cluster_centers_[:,1], s = 300,)
mtp.title('clusters of customers')
mtp.xlabel('latitude')
mtp.ylabel('longitude')
mtp.legend()
mtp.show()
```

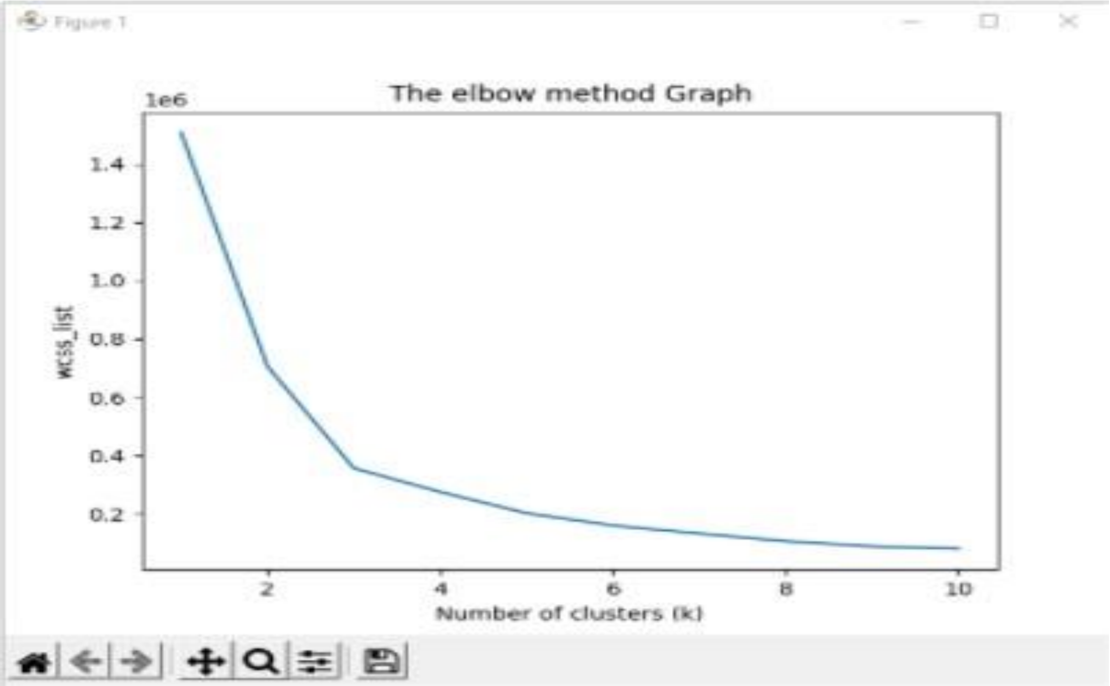
<u>Output:</u>



```
C:\Users\MyPc\PycharmProjects\DS_lab\venv\Scripts\python.exe "C:/Users/MyPc/PycharmProjects/DS_lab/kmeans cluster.py"
[[ 4.25462450e+01  1.60155400e+00]
 [ 2.34240760e+01  5.38478180e+01]
 [ 3.39391100e+01  6.77099530e+01]
 [ 1.70608160e+01 -6.17964280e+01]
 [ 1.82205540e+01 -6.30686150e+01]
 [ 4.11533320e+01  2.01683310e+01]
 [ 4.00690990e+01  4.50381890e+01]
 [ 1.22260790e+01 -6.90600870e+01]
 [-1.12026920e+01  1.78738870e+01]
 [-7.52509730e+01 -7.13890000e-02]
 [-3.84160970e+01 -6.36166720e+01]
 [-1.42709720e+01 -1.70132217e+02]
```



```
 [ 1.29843050e+01 -6.12872280e+01]
 [ 6.42375000e+00 -6.65897300e+01]
 [ 1.84206950e+01 -6.46399680e+01]
 [ 1.83357650e+01 -6.48963350e+01]
 [ 1.40583240e+01  1.08277199e+02]
 [-1.53767060e+01  1.66959158e+02]
 [-1.37687520e+01 -1.77156097e+02]
 [-1.37590290e+01 -1.72104629e+02]
 [ 4.26026360e+01  2.09029770e+01]
 [ 1.55527270e+01  4.85163880e+01]
 [-1.28275000e+01  4.51662440e+01]
 [-3.05594820e+01  2.29375060e+01]
 [-1.31338970e+01  2.78493320e+01]
 [-1.90154380e+01  2.91548570e+01]]
```



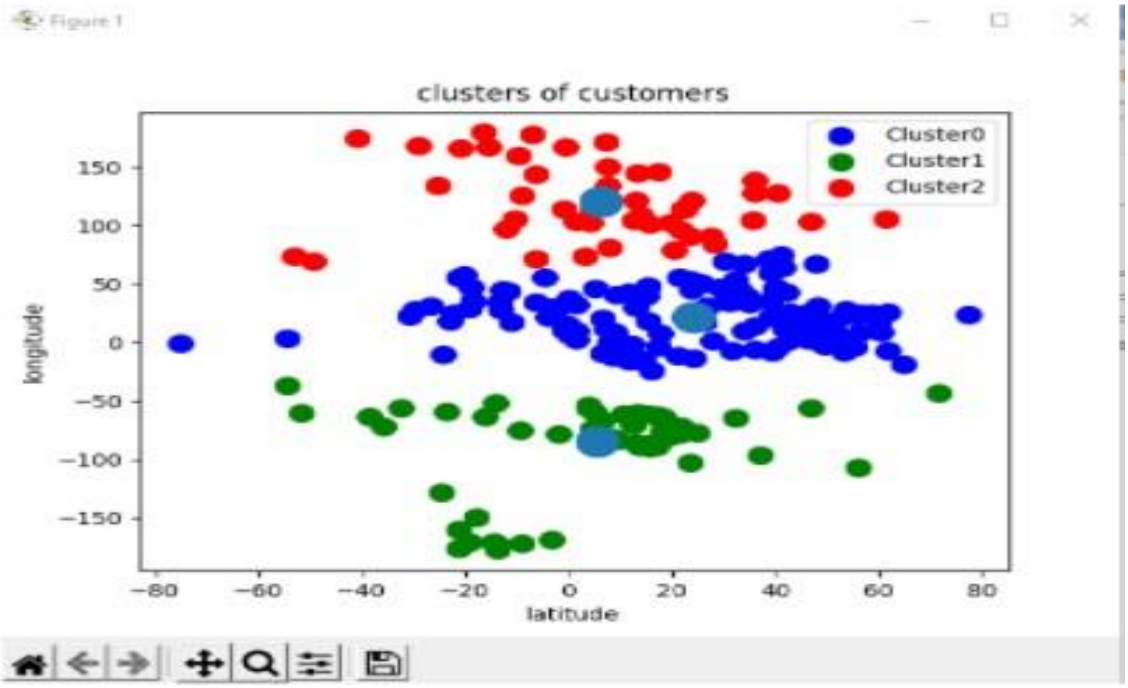


Figure 1
clusters of customers
Cluster0
Cluster1
Cluster2
150
100
50
0
−50
−100
−150
longitude
−80
−60
−40
−20
0
20
40
60
80
latitude

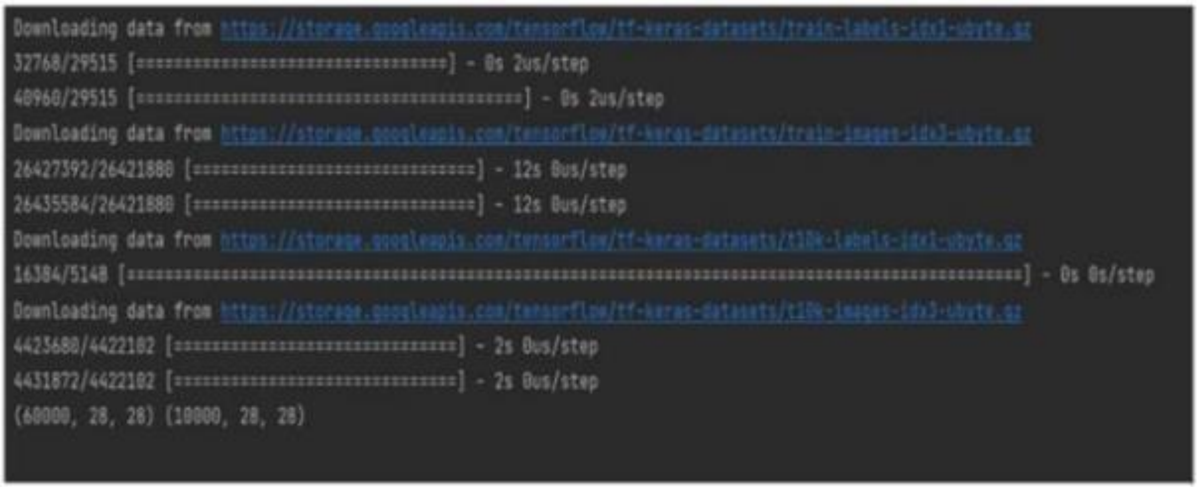
PROGRAM NO : 13 **Date : 02/02/2022**

AIM : Programs on convolutional neural network to classify images from any standard dataset in the public domain.

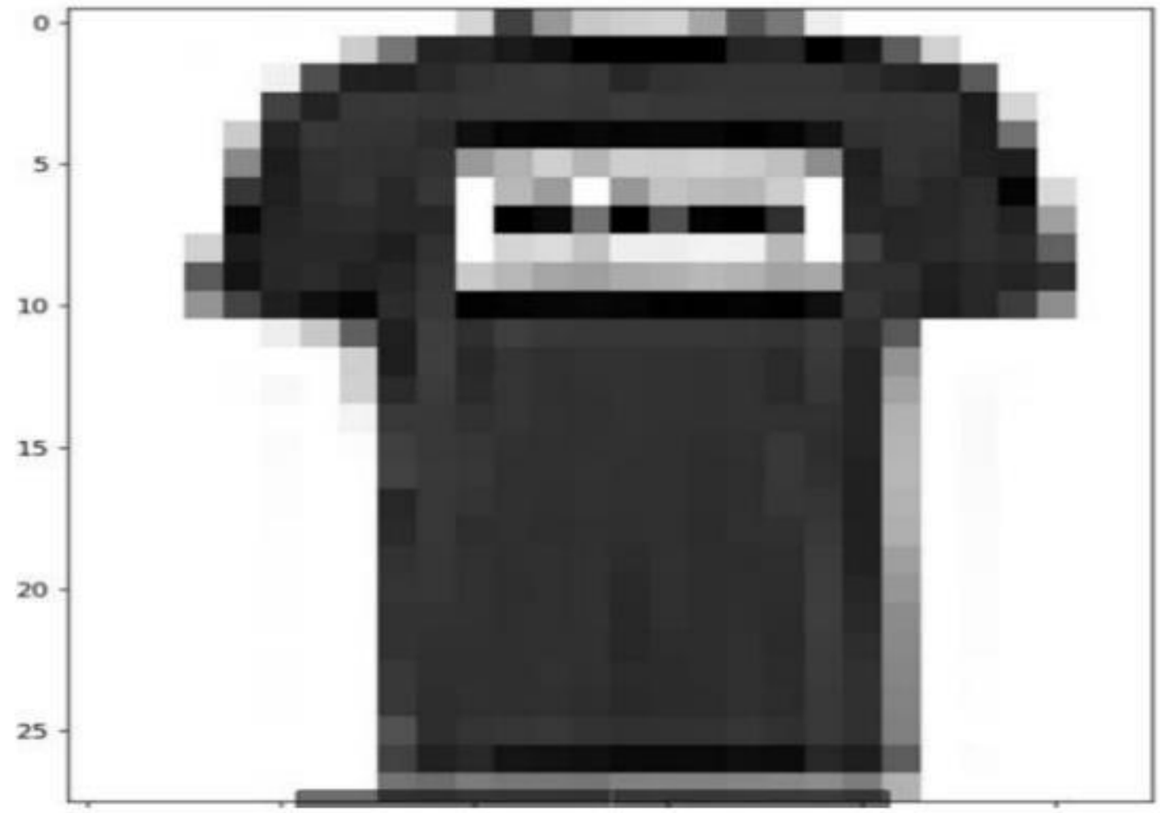
Program Code :

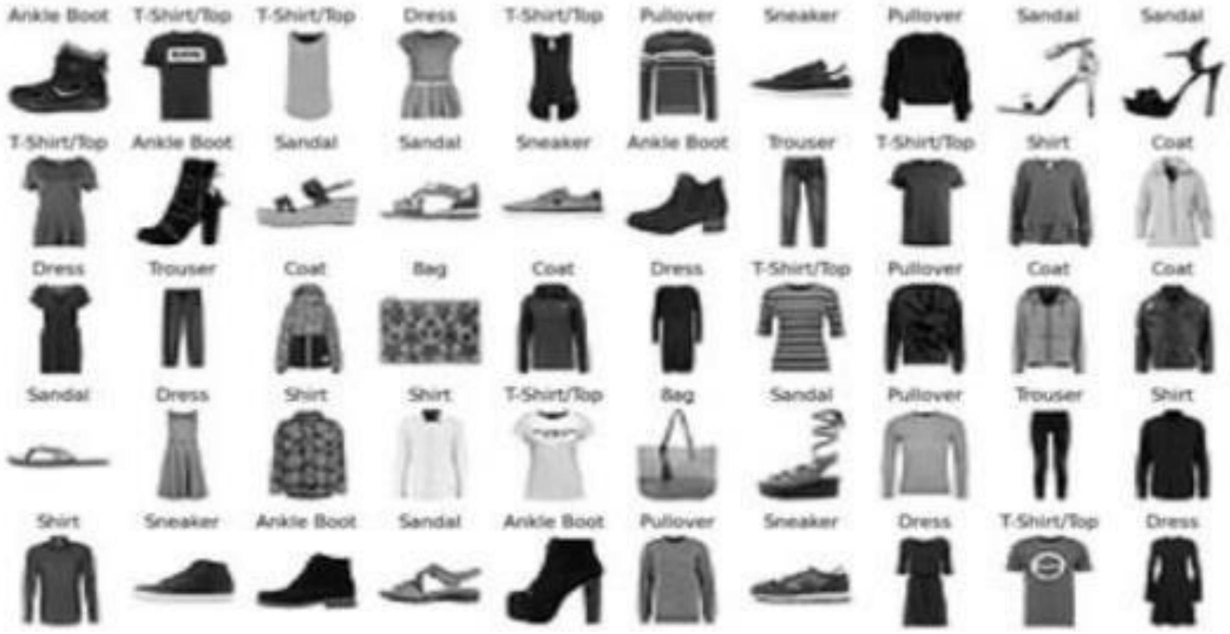
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow import keras
np.random.seed(42)
# tf.set.random. seed(42)
fashion_mnist = keras.datasets.fashion_mnist
(X_train, y_train), (X_test, y_test) = fashion_mnist.load_data()
print(X_train.shape, X_test.shape)
X_train = X_train / 255.0
X_test = X_test / 255.0
plt.imshow(X_train[1], cmap='binary')
plt.show()
np.unique(y_test)
class_names = ['T-Shirt/Top', 'Trouser', 'Pullover', 'Dress', 'Coat',
'Sandal', 'Shirt', 'Sneaker', '8ag', 'Ankle Boot']
n_rows = 5
n_cols = 10
plt.figure(figsize=(n_cols * 1.4, n_rows * 1.6))
for row in range(n_rows):
for col in range(n_cols):
index = n_cols * row + col
plt.subplot(n_rows, n_cols, index + 1)
plt.imshow(X_train[index], cmap='binary', interpolation='nearest')
plt.axis('off')
plt.title(class_names[y_train[index]])
plt.show()
model_CNN = keras.models.Sequential()
model_CNN.add(keras.layers.Conv2D(filters=32, kernel_size=7,
padding='same', activation='relu', input_shape=[28, 28, 1]))
model_CNN.add(keras.layers.MaxPooling2D(pool_size=2))
model_CNN.add(keras.layers.Conv2D(filters=64, kernel_size=3,
padding='same', activation='relu'))
model_CNN.add(keras.layers.MaxPooling2D(pool_size=2))
model_CNN.add(keras.layers.Conv2D(filters=32, kernel_size=3,
padding='same', activation='relu'))
model_CNN.add(keras.layers.MaxPooling2D(pool_size=2))
model_CNN.summary()
model_CNN.add(keras.layers.Flatten())


model_CNN.add(keras.layers.Dense(units=128, activation='relu'))
model_CNN.add(keras.layers.Dense(units=64, activation='relu'))
model_CNN.add(keras.layers.Dense(units=10, activation='softmax'))
model_CNN.summary()
model_CNN.compile(loss='sparse_categorical_crossentropy',
optimizer='adam', metrics=['accuracy'])
X_train = X_train[..., np.newaxis]
X_test = X_test[..., np.newaxis]
history_CNN = model_CNN.fit(X_train, y_train, epochs=2,
validation_split=0.1)
pd.DataFrame(history_CNN.history).plot()
plt.grid(True)
plt.xlabel('epochs')
plt.ylabel('loss/accuracy')

plt.title('Training and validation plot')
plt.show()
test_loss, test_accuracy = model_CNN.evaluate(X_test, y_test)
print(' Test Loss :{}, Test Accuracy : {}'.format(test_loss,
test_accuracy))
```

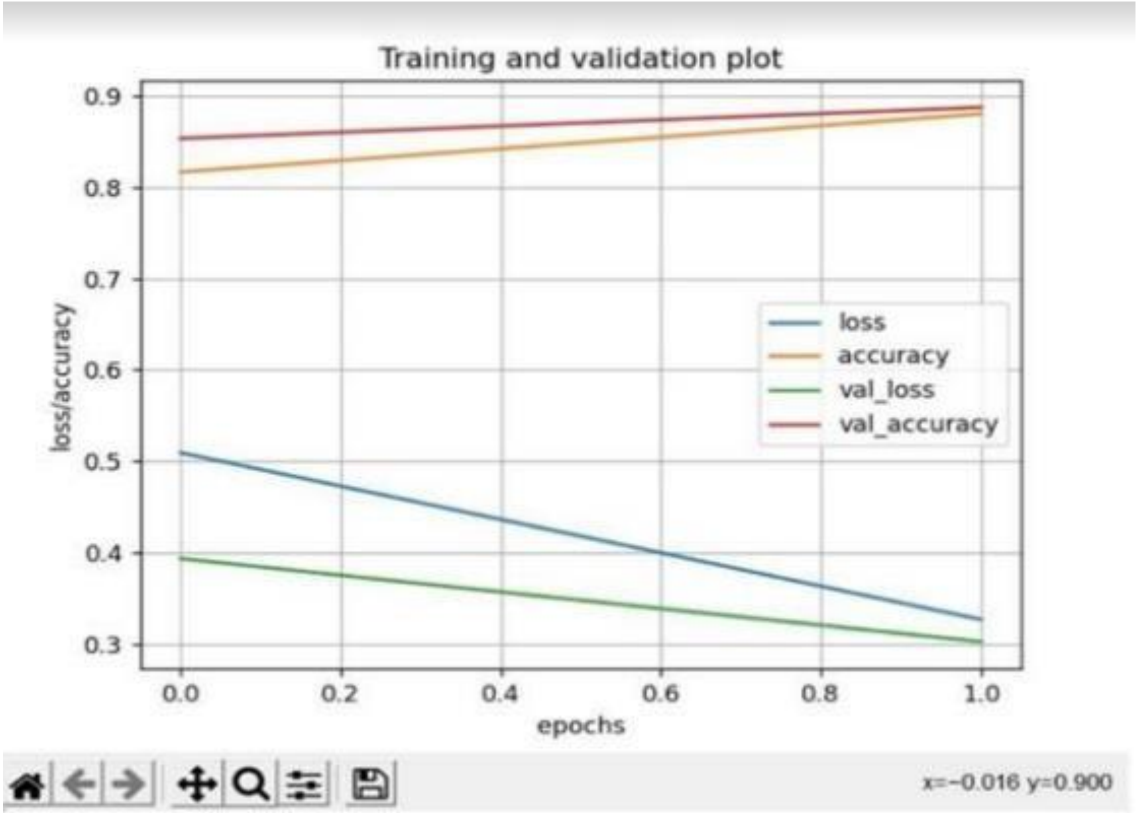
Output:

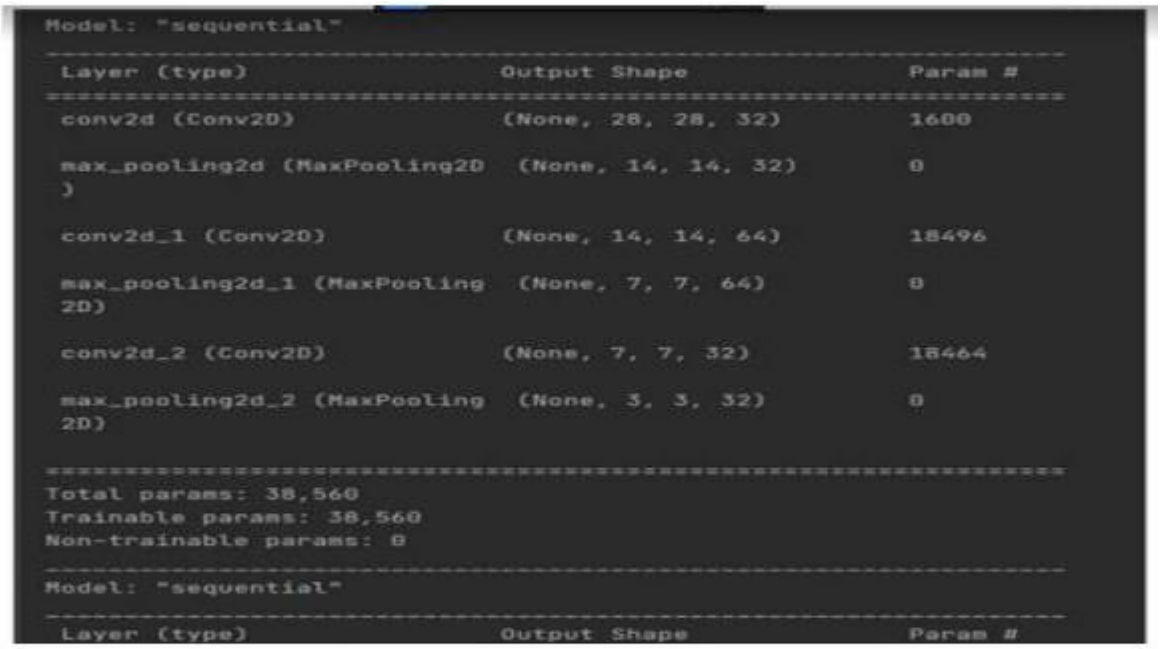


```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-labels-idx1-ubyte.gz
32768/29515 [==============================] - 0s 2us/step
40960/29515 [=========================================] - 0s 2us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-images-idx3-ubyte.gz
26427392/26421880 [==============================] - 12s 0us/step
26435584/26421880 [==============================] - 12s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-labels-idx1-ubyte.gz
16384/5148 [==========================================================================================] - 0s 0s/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-images-idx3-ubyte.gz
4423680/4422102 [==============================] - 2s 0us/step
4431872/4422102 [==============================] - 2s 0us/step
(60000, 28, 28) (10000, 28, 28)
```


0
5
10
15
20
25


Ankle Boot
T-Shirt/Top
T-Shirt/Top
Dress
T-Shirt/Top
Pullover
Sneaker
Pullover
Sandal
Sandal
T-Shirt/Top
Ankle Boot
Sandal
Sandal
Sneaker
Ankle Boot
Trouser
T-Shirt/Top
Shirt
Coat
Dress
Trouser
Coat
Bag
Coat
Dress
T-Shirt/Top
Pullover
Coat
Coat
Sandal
Dress
Shirt
Shirt
T-Shirt/Top
Bag
Sandal
Pullover
Trouser
Shirt
Shirt
Sneaker
Ankle Boot
Sandal
Ankle Boot
Pullover
Sneaker
Dress
T-Shirt/Top
Dress





```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 28, 28, 32)        1600

 max_pooling2d (MaxPooling2D  (None, 14, 14, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 14, 14, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 7, 7, 64)         0
 2D)

 conv2d_2 (Conv2D)           (None, 7, 7, 32)          18464

 max_pooling2d_2 (MaxPooling  (None, 3, 3, 32)         0
 2D)

=================================================================
Total params: 38,560
Trainable params: 38,560
Non-trainable params: 0
_________________________________________________________________
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
```

Date : 16/02/2022

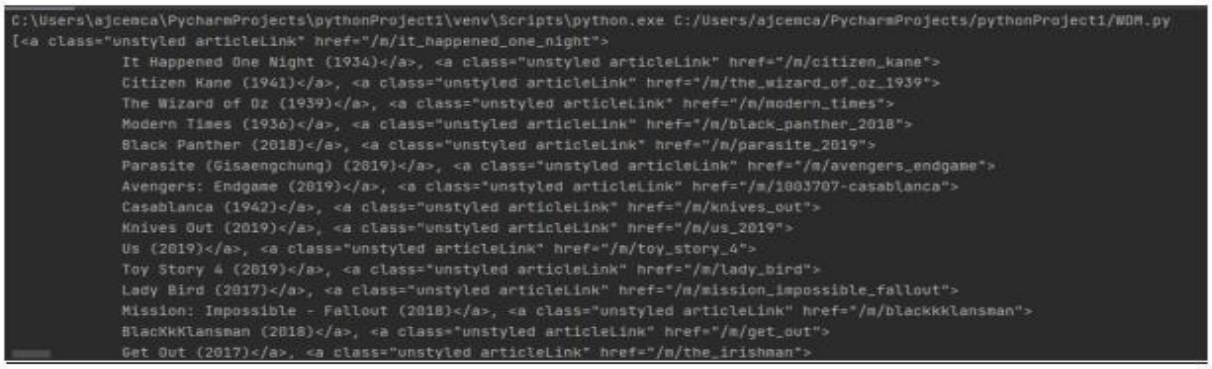
PROGRAM NO:14

AIM : Program to implement a simple web crawler using python.

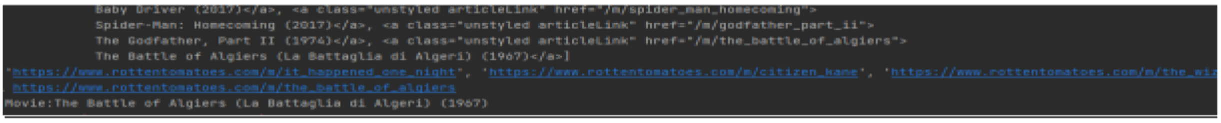
Program Code :

```
import requests
import lxml
from bs4 import BeautifulSoup
url = "https://www.rottentomatoes.com/top/bestofrt/"
headers = {
'User-Agent' : 'Mozilla/5.0(Windows NT 6.1; WOW64) AppleWebkit/537.36
(KHTML, like Gecko) Chrome/63.0.3239.132 Safari/537.36 QIHU 360SE'
}
f = requests.get(url, headers = headers)
movies_lst = []
soup = BeautifulSoup(f.content, 'html.parser')
movies = soup.find('table', {
'class': 'table'
}).find_all('a')
print(movies)
num = 0
for anchor in movies:
urls = 'https://www.rottentomatoes.com' + anchor['href']
movies_lst.append(urls)
print(movies_lst)
num += 1
movie_url = urls
movie_f = requests.get(movie_url, headers = headers)
movie_soup = BeautifulSoup(movie_f.content, 'lxml')
movie_content = movie_soup.find('div', {

'class': 'Movies_synopsis clamp clamp-6 js-clamp'
})
print(num,urls,'\n','Movie:' +anchor.string.strip())
print('Movie info:' + movie_content.string.strip())
```

Output:



```
C:\Users\ajcemca\PycharmProjects\pythonProject1\venv\Scripts\python.exe C:/Users/ajcemca/PycharmProjects/pythonProject1/WDM.py
[<a class="unstyled articleLink" href="/m/it_happened_one_night">
            It Happened One Night (1934)</a>, <a class="unstyled articleLink" href="/m/citizen_kane">
            Citizen Kane (1941)</a>, <a class="unstyled articleLink" href="/m/the_wizard_of_oz_1939">
            The Wizard of Oz (1939)</a>, <a class="unstyled articleLink" href="/m/modern_times">
            Modern Times (1936)</a>, <a class="unstyled articleLink" href="/m/black_panther_2018">
            Black Panther (2018)</a>, <a class="unstyled articleLink" href="/m/parasite_2019">
            Parasite (Gisaengchung) (2019)</a>, <a class="unstyled articleLink" href="/m/avengers_endgame">
            Avengers: Endgame (2019)</a>, <a class="unstyled articleLink" href="/m/1003707-casablanca">
            Casablanca (1942)</a>, <a class="unstyled articleLink" href="/m/knives_out">
            Knives Out (2019)</a>, <a class="unstyled articleLink" href="/m/us_2019">
            Us (2019)</a>, <a class="unstyled articleLink" href="/m/toy_story_4">
            Toy Story 4 (2019)</a>, <a class="unstyled articleLink" href="/m/lady_bird">
            Lady Bird (2017)</a>, <a class="unstyled articleLink" href="/m/mission_impossible_fallout">
            Mission: Impossible - Fallout (2018)</a>, <a class="unstyled articleLink" href="/m/blackkklansman">
            BlacKkKlansman (2018)</a>, <a class="unstyled articleLink" href="/m/get_out">
            Get Out (2017)</a>, <a class="unstyled articleLink" href="/m/the_irishman">
```



```
            Baby Driver (2017)</a>, <a class="unstyled articleLink" href="/m/spider_man_homecoming">
            Spider-Man: Homecoming (2017)</a>, <a class="unstyled articleLink" href="/m/godfather_part_ii">
            The Godfather, Part II (1974)</a>, <a class="unstyled articleLink" href="/m/the_battle_of_algiers">
            The Battle of Algiers (La Battaglia di Algeri) (1967)</a>]
'https://www.rottentomatoes.com/m/it_happened_one_night', 'https://www.rottentomatoes.com/m/citizen_kane', 'https://www.rottentomatoes.com/m/the_wiz
 https://www.rottentomatoes.com/m/the_battle_of_algiers
Movie:The Battle of Algiers (La Battaglia di Algeri) (1967)
```

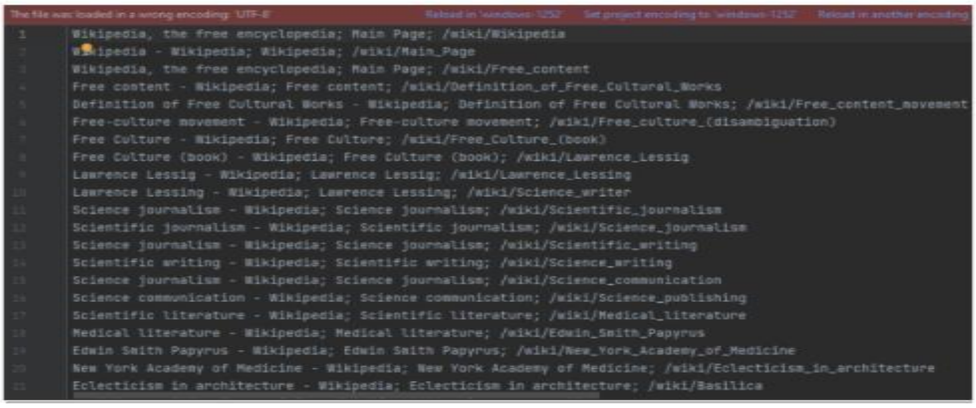
PROGRAM NO : 15 **Date : 16/02/2022**

AIM : Program to implement a simple web crawler using python.

Program Code :

```
from bs4 import BeautifulSoup
import requests
pages_crawled = []
def crawler(url):
page = requests.get(url)
soup = BeautifulSoup(page.text, 'html.parser')
links = soup.find_all('a')
for link in links:
if 'href' in link.attrs:
if link['href'].startswith('/wiki') and ':' not in link['href']:
if link['href'] not in pages_crawled:
new_link = f"https://en.wikipedia.org{link['href']}"
pages_crawled.append(link['href'])
try:
with open('data.csv', 'a') as file:
file.write(f'{soup.title.text}; {soup.h1.text}; {link["href"]}\n')
crawler(new_link)
except:
continue
crawler('https://en.wikipedia.org')
```

Output:



The file was loaded in a wrong encoding: 'UTF-8'   Reload in 'windows-1252'   Set project encoding to 'windows-1252'   Reload in another encoding

```
Wikipedia, the free encyclopedia; Main Page; /wiki/Wikipedia
Wikipedia - Wikipedia; Wikipedia; /wiki/Main_Page
Wikipedia, the free encyclopedia; Main Page; /wiki/Free_content
Free content - Wikipedia; Free content; /wiki/Definition_of_Free_Cultural_Works
Definition of Free Cultural Works - Wikipedia; Definition of Free Cultural Works; /wiki/Free_content_movement
Free-culture movement - Wikipedia; Free-culture movement; /wiki/Free_culture_(disambiguation)
Free Culture - Wikipedia; Free Culture; /wiki/Free_Culture_(book)
Free Culture (book) - Wikipedia; Free Culture (book); /wiki/Lawrence_Lessig
Lawrence Lessig - Wikipedia; Lawrence Lessig; /wiki/Lawrence_Lessing
Lawrence Lessing - Wikipedia; Lawrence Lessing; /wiki/Science_writer
Science journalism - Wikipedia; Science journalism; /wiki/Scientific_journalism
Scientific journalism - Wikipedia; Scientific journalism; /wiki/Science_journalism
Science journalism - Wikipedia; Science journalism; /wiki/Scientific_writing
Scientific writing - Wikipedia; Scientific writing; /wiki/Science_writing
Science journalism - Wikipedia; Science journalism; /wiki/Science_communication
Science communication - Wikipedia; Science communication; /wiki/Science_publishing
Scientific literature - Wikipedia; Scientific literature; /wiki/Medical_literature
Medical literature - Wikipedia; Medical literature; /wiki/Edwin_Smith_Papyrus
Edwin Smith Papyrus - Wikipedia; Edwin Smith Papyrus; /wiki/New_York_Academy_of_Medicine
New York Academy of Medicine - Wikipedia; New York Academy of Medicine; /wiki/Eclecticism_in_architecture
Eclecticism in architecture - Wikipedia; Eclecticism in architecture; /wiki/Basilica
```

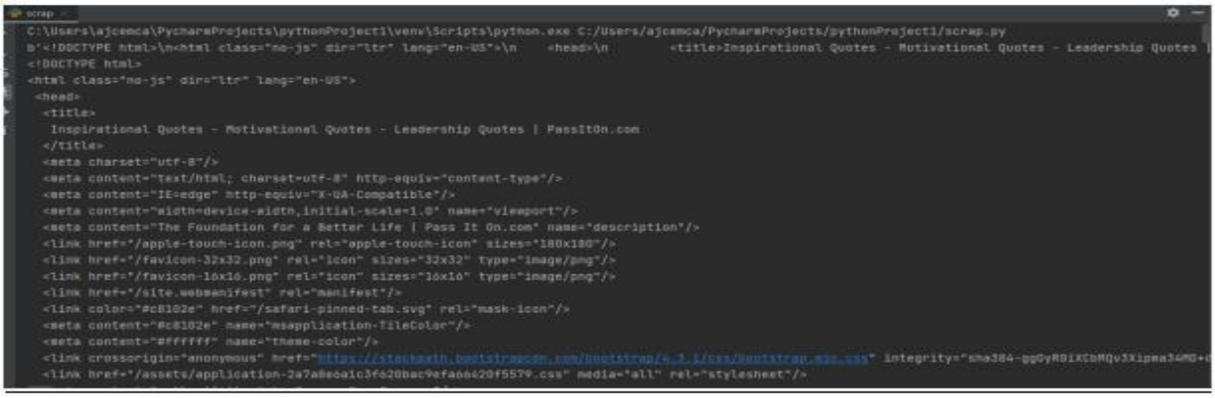
<u>PROGRAM NO</u> : 16 **Date : 16/02/2022**

AIM : Program to implement scrap of any website.

<u>Program Code</u> :

```
import requests
from bs4 import BeautifulSoup
import csv
import lxml
URL = "https://www.values.com/inspirational-quotes"
r = requests.get(URL)
print(r.content)
soup = BeautifulSoup(r.content, 'lxml')
print(soup.prettify())
quotes = []
table = soup.find('div', attrs={'id': 'all_quotes'})
for row in table.findAll('div',attrs={'class':'col-6 col-lg-3 text-center margin-30px-bottom sm-
margin-30px-top'}):
quote = {}
quote['theme'] = row.h5.text
quote['url'] = row.a['href']
quote['img'] = row.img['src']
quote['lines'] = row.img['alt'].split("#")[0]
quote['author'] = row.img['alt'].split("#")[1]
quotes.append(quote)
filename = 'inspirational_quotes.csv'
with open(filename, 'w', newline='') as f:
w = csv.DictWriter(f, ['theme', 'url', 'img', 'lines', 'author'])
w.writeheader()
for quote in quotes:
w.writerow(quote)
```

<u>Output:</u>



```
C:\Users\ajcemca\PycharmProjects\pythonProject1\venv\Scripts\python.exe C:/Users/ajcemca/PycharmProjects/pythonProject1/scrap.py
b'<!DOCTYPE html>\n<html class="no-js" dir="ltr" lang="en-US">\n    <head>\n        <title>Inspirational Quotes - Motivational Quotes - Leadership Quotes |
<!DOCTYPE html>
<html class="no-js" dir="ltr" lang="en-US">
 <head>
  <title>
   Inspirational Quotes - Motivational Quotes - Leadership Quotes | PassItOn.com
  </title>
  <meta charset="utf-8"/>
  <meta content="text/html; charset=utf-8" http-equiv="content-type"/>
  <meta content="IE=edge" http-equiv="X-UA-Compatible"/>
  <meta content="width=device-width,initial-scale=1.0" name="viewport"/>
  <meta content="The Foundation for a Better Life | Pass It On.com" name="description"/>
  <link href="/apple-touch-icon.png" rel="apple-touch-icon" sizes="180x180"/>
  <link href="/favicon-32x32.png" rel="icon" sizes="32x32" type="image/png"/>
  <link href="/favicon-16x16.png" rel="icon" sizes="16x16" type="image/png"/>
  <link href="/site.webmanifest" rel="manifest"/>
  <link color="#c8102e" href="/safari-pinned-tab.svg" rel="mask-icon"/>
  <meta content="#c8102e" name="msapplication-TileColor"/>
  <meta content="#ffffff" name="theme-color"/>
  <link crossorigin="anonymous" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css" integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+d
  <link href="/assets/application-2a7a8e6a1c3fe20bac9efa66420f5579.css" media="all" rel="stylesheet"/>
```

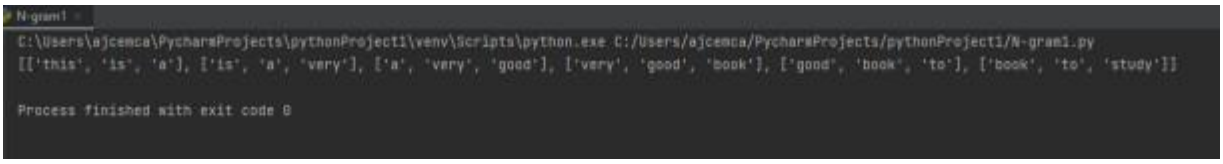
PROGRAM NO : 17 **Date : 16/02/2022**

AIM : Program for Natural Language Processing which performs n-grams.

Program Code :

```
def generate_ngrams(text, WordsToCombine):
words = text.split()
output = []
for i in range(len(words) - WordsToCombine + 1):
output.append(words[i:i + WordsToCombine])
return output
x=generate_ngrams(text='this is a very good book to study', WordsToCombine=3)
print(x)
```

Output:



```
C:\Users\ajcemca\PycharmProjects\pythonProject1\venv\Scripts\python.exe C:/Users/ajcemca/PycharmProjects/pythonProject1/N-gram1.py
[['this', 'is', 'a'], ['is', 'a', 'very'], ['a', 'very', 'good'], ['very', 'good', 'book'], ['good', 'book', 'to'], ['book', 'to', 'study']]

Process finished with exit code 0
```

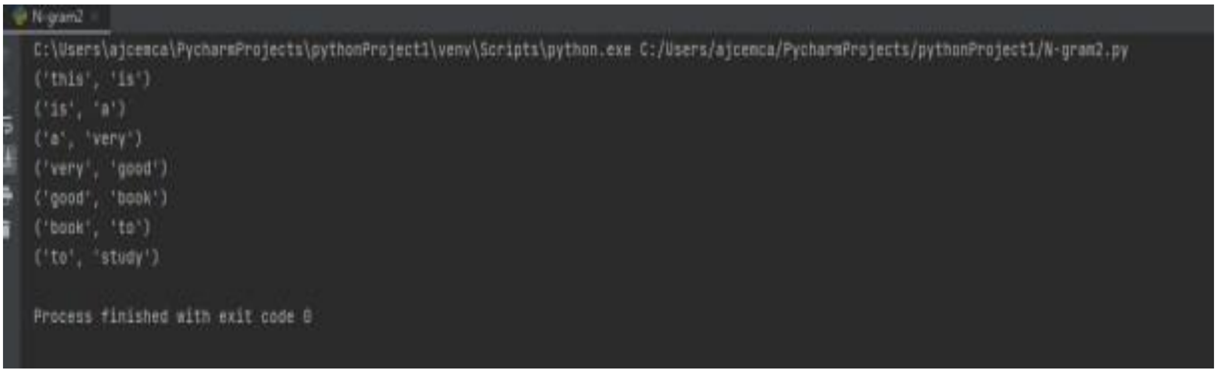
PROGRAM NO : 18 **Date : 16/02/2022**

AIM : Program for Natural Language Processing which performs n-grams (Using in built functions).

Program Code :

```
import nltk
from nltk.util import ngrams
sampleText = 'this is a very good book to study'
NGRAMS = ngrams(sequence=nltk.word_tokenize(sampleText), n=2)
for grams in NGRAMS:
print(grams)
```

Output:



```
C:\Users\ajcemca\PycharmProjects\pythonProject1\venv\Scripts\python.exe C:/Users/ajcemca/PycharmProjects/pythonProject1/N-gram2.py
('this', 'is')
('is', 'a')
('a', 'very')
('very', 'good')
('good', 'book')
('book', 'to')
('to', 'study')

Process finished with exit code 0
```

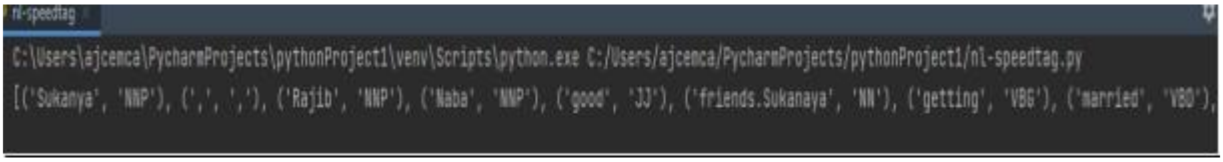
PROGRAM NO : 19 **Date : 16/02/2022**

AIM : Program for Natural Language Processing which performs speech tagging.

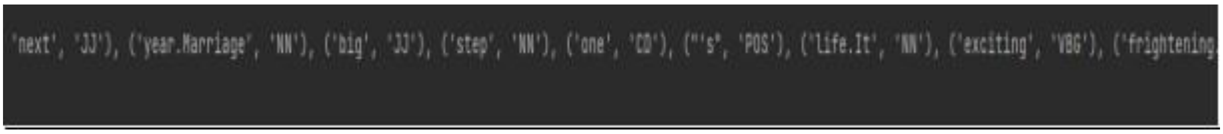
Program Code :

```
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize, sent_tokenize
stop_words = set(stopwords.words('english'))
txt = "Sukanya, Rajib and Naba are my good friends."\
"Sukanaya is getting married next year."\
"Marriage is a big step in one's life."\
"It is both exciting and frightening."\
"But friendship is a sacred bond between people."\
"it is a special kind of love between us."\
"Many of you must have tried searching for a friend."\
"But never found the right one."
tokenized = sent_tokenize(txt)
for i in tokenized:
wordsList = nltk.word_tokenize(i)
wordsList = [w for w in wordsList if not w in stop_words]
tagged = nltk.pos_tag(wordsList)
print(tagged)
```

Output:



```
C:\Users\ajcemca\PycharmProjects\pythonProject1\venv\Scripts\python.exe C:/Users/ajcemca/PycharmProjects/pythonProject1/nl-speedtag.py
[('Sukanya', 'NNP'), (',', ','), ('Rajib', 'NNP'), ('Naba', 'NNP'), ('good', 'JJ'), ('friends.Sukanaya', 'NN'), ('getting', 'VBG'), ('married', 'VBD'),
```



```
'next', 'JJ'), ('year.Marriage', 'NN'), ('big', 'JJ'), ('step', 'NN'), ('one', 'CD'), ("'s", 'POS'), ('life.It', 'NN'), ('exciting', 'VBG'), ('frightening
```

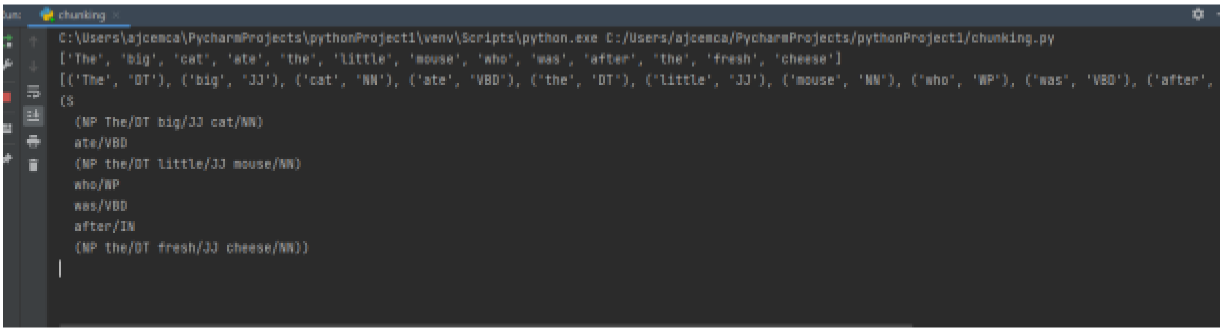
PROGRAM NO : 20 **Date : 23/02/2022**

AIM : Write a python program for natural language processing which perform chunking

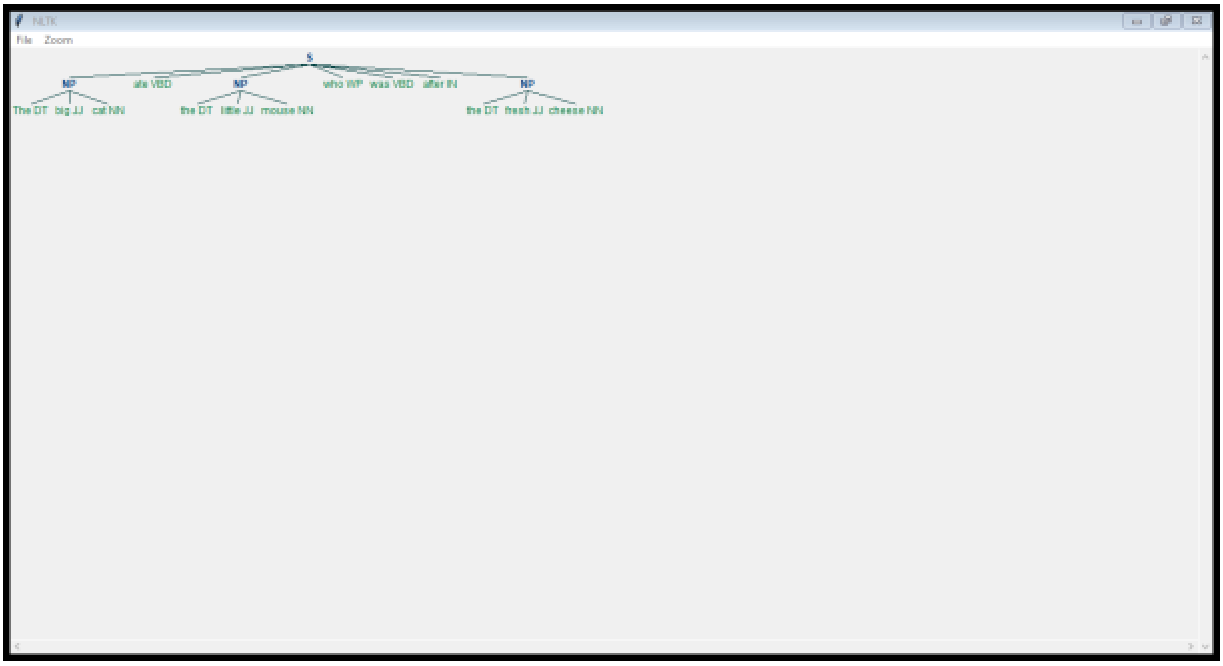
Program Code :

```
import nltk
new="The big cat ate the little mouse who was after the fresh cheese"
new_tokens=nltk.word_tokenize(new)
print(new_tokens)
new_tag=nltk.pos_tag(new_tokens)
print(new_tag)
grammer=r"NP: {<DT>?<JJ>*<NN>}"
chunkParser=nltk.RegexpParser(grammer)
chunked=chunkParser.parse(new_tag)
print(chunked)
chunked.draw()
```

Output



```
C:\Users\ajcemca\PycharmProjects\pythonProject1\venv\Scripts\python.exe C:/Users/ajcemca/PycharmProjects/pythonProject1/chunking.py
['The', 'big', 'cat', 'ate', 'the', 'little', 'mouse', 'who', 'was', 'after', 'the', 'fresh', 'cheese']
[('The', 'DT'), ('big', 'JJ'), ('cat', 'NN'), ('ate', 'VBD'), ('the', 'DT'), ('little', 'JJ'), ('mouse', 'NN'), ('who', 'WP'), ('was', 'VBD'), ('after',
(S
  (NP The/DT big/JJ cat/NN)
  ate/VBD
  (NP the/DT little/JJ mouse/NN)
  who/WP
  was/VBD
  after/IN
  (NP the/DT fresh/JJ cheese/NN))
```


NLTK
File
Zoom
S
NP
ate VBD
NP
who WP
was VBD
after IN
NP
The DT
big JJ
cat NN
the DT
little JJ
mouse NN
the DT
fresh JJ
cheese NN

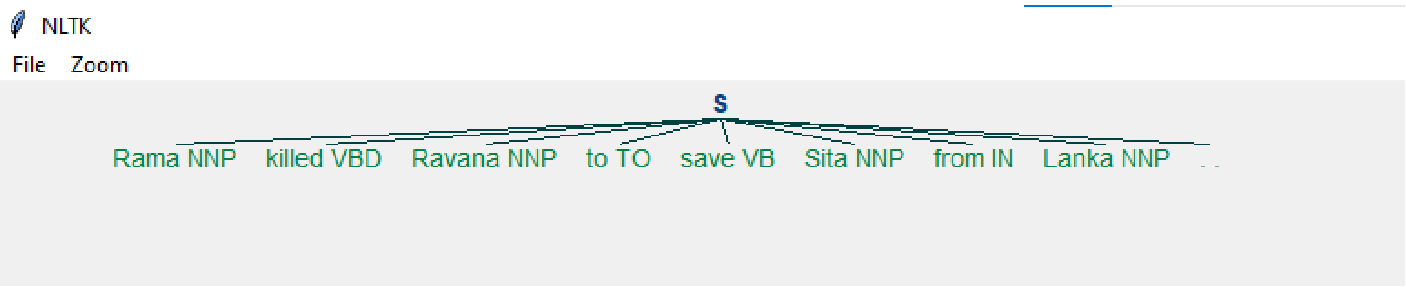
PROGRAM NO : 21 **Date : 23/02/2022**

AIM : Write a python program for natural language processing which perform chunking

Program Code :

```
import nltk
nltk.download('averaged_perception_tagger')
sample_text="""
Rama killed Ravana to save Sita from Lanka .
The legend of the 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"""VB: {} """
 chunkParser=nltk.RegexpParser(chunkGram)
 chunked=chunkParser.parse(tagged_words)
  print(chunked)
  chunked.draw()
```

Output



NLTK
File
Zoom
S
Rama NNP
killed VBD
Ravana NNP
to TO
save VB
Sita NNP
from IN
Lanka NNP
. .