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

AIM : Perform all matrix operation using python.

PROGRAM CODE

```
import numpy

a=numpy.array([[1,2,3],[4,5,6],[7,8,9]])

b=numpy.array([[7,8,9],[4,5,6],[1,2,3]])

print(a)

print(b)

print("Addition of two matrices")

print(numpy.add(a,b))

print("Substraction of two matrices")

print(numpy.subtract(a,b))

print("Division of two matrices")

print(numpy.multiply(a,b))

print("division of two matrices")

print(numpy.divide(a,b))

print("dot of two matrices")

print(numpy.dot(a,b))

print("summation")

print(numpy.sum(a))

print("Transpose")
```

```
print(a.T)

print("Square root")

print(numpy.sqrt(a))
```

OUTPUT

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
[[7 8 9]
 [4 5 6]
 [1 2 3]]
Addition of two matrices
[[ 8 10 12]
 [ 8 10 12]
 [ 8 10 12]]
Substraction of two matrices
[[-6 -6 -6]
 [ 0  0  0]
 [ 6  6  6]]
Division of two matrices
[[ 7 16 27]
 [16 25 36]
 [ 7 16 27]]
division of two matrices
[[0.14285714 0.25       0.33333333]
 [1.         1.         1.        ]
 [7.         4.         3.        ]]
```

```
dot of two matrices
[[ 18  24  30]
 [ 54  69  84]
 [ 90 114 138]]
summation
45
Transpose
[[1 4 7]
 [2 5 8]
 [3 6 9]]
Square root
[[1.         1.41421356 1.73205081]
 [2.         2.23606798 2.44948974]
 [2.64575131 2.82842712 3.        ]]

Process finished with exit code 0
```

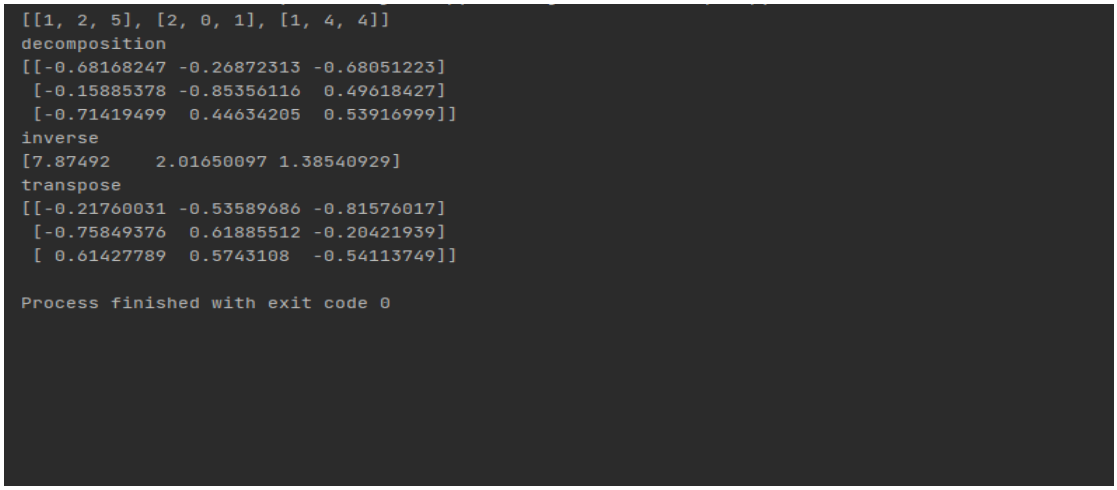
PROGRAM NO : 02 **Date :01/12/2021**

AIM: Program to perform SVD (Singular value Decomposition) using Python.

PROGRAM CODE

```
from scipy. linalg import svd
from numpy import array
A= ([[1,2,5], [2,0,1], [1,4,4]])
print(A)
X, B, T=svd(A)
print("decomposition")
print(X)
print("inverse")
print(B)
print("transpose")
print(T)
```

OUTPUT



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
from sklearn.metrics import accuracy_score
iris = load_iris()
x=iris.data
y=iris.target
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=42)
knn=KNeighborsClassifier(n_neighbors=7)
knn.fit(x_train,y_train)
print(knn.predict(x_test))
V=knn.predict(x_test)
result=accuracy_score (y_test, V)
print ("accuracy:", result)
```

OUTPUT

```
[1 0 2 1 1 0 1 2 1 1 2 0 0 0 0 1 2 1 1 2 0 2 0 2 2 2 2 2 0 0 0 0 1 0 0 2 1
 0 0 0 2 1 1 0 0 1 1 2 1 2 1 2 1 0 2 1 0 0 0 1 2 0 0 0 1 0 1 2]
accuracy: 0.9852941176470589

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)
# Locate the most similar neighbors
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
# Make a classification prediction with 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

# Test distance function
dataset = [[2.781, 2.550,0],
        [1.465, 2.326,3],
        [3.398, 4.429,5],

        [1.388, 1.857,11],
        [3.064, 3.393,3],
        [7.624, 2.235,4],
        [5.338, 2.775,8]]
prediction = predict_classification(dataset, dataset[0], 3)

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

OUTPUT

```
Expected 0, Got 3.

Process finished with exit code 0
```

PROGRAM NO : 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.

PROGRAM CODE

```
import numpy as np
import matplotlib.pyplot as ply
import pandas as pd
dataset = pd.read_csv('Social_Network_Ads.csv')
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_test)
print(X_train)

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)
```

OUTPUT

```
[[-7.98950822e-01  4.94607583e-01]
 [-2.12648508e-02 -5.77359062e-01]
 [-3.12897090e-01  1.46942725e-01]
 [-7.98950822e-01  2.62831011e-01]
 [-3.12897090e-01 -5.77359062e-01]
 [-1.09058306e+00 -1.44652121e+00]
 [-7.01740076e-01 -1.59138156e+00]
 [-2.15686344e-01  2.14601566e+00]
 [-1.96547978e+00 -5.58617754e-02]
 [ 8.53631867e-01 -7.80163563e-01]
 [-7.98950822e-01 -6.06331134e-01]
 [-9.93372315e-01 -4.32498705e-01]
 [-1.18475597e-01 -4.32498705e-01]
 [ 7.59458956e-02  2.04886868e-01]
 [-1.77105829e+00  4.65635512e-01]
 [-6.04529329e-01  1.36376973e+00]
 [-1.18475597e-01  2.04886868e-01]
 [-1.86826903e+00  4.36663440e-01]
 [ 1.63131784e+00  1.74040666e+00]
 [-3.12897090e-01 -1.38857706e+00]
 [-3.12897090e-01 -6.64275277e-01]
 [ 8.53631867e-01  2.14601566e+00]
 [ 2.70367388e-01 -5.48386991e-01]
 [ 8.53631867e-01  1.01610487e+00]
 [-1.47942605e+00 -1.21474464e+00]
```

```
 [ 7.59458956e-02  2.62831011e-01]
 [-1.18779381e+00  3.20775154e-01]
 [-1.28500455e+00  2.91803083e-01]
 [-9.93372315e-01  4.36663440e-01]
 [ 1.63131784e+00 -8.96051849e-01]
 [ 1.14526411e+00  5.23579655e-01]
 [ 1.04805336e+00  5.23579655e-01]
 [ 1.33968560e+00  2.31984809e+00]
 [-3.12897090e-01 -1.42777990e-01]
 [ 3.67578135e-01 -4.61470776e-01]
 [-4.10107836e-01 -7.80163563e-01]
 [-1.18475597e-01 -5.19414919e-01]
 [ 9.50842613e-01 -1.15680049e+00]
 [-8.96161568e-01 -7.80163563e-01]
 [-2.15686344e-01 -5.19414919e-01]
 [-1.09058306e+00 -4.61470776e-01]
 [-1.18779381e+00  1.39274180e+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 :08/12/2021

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

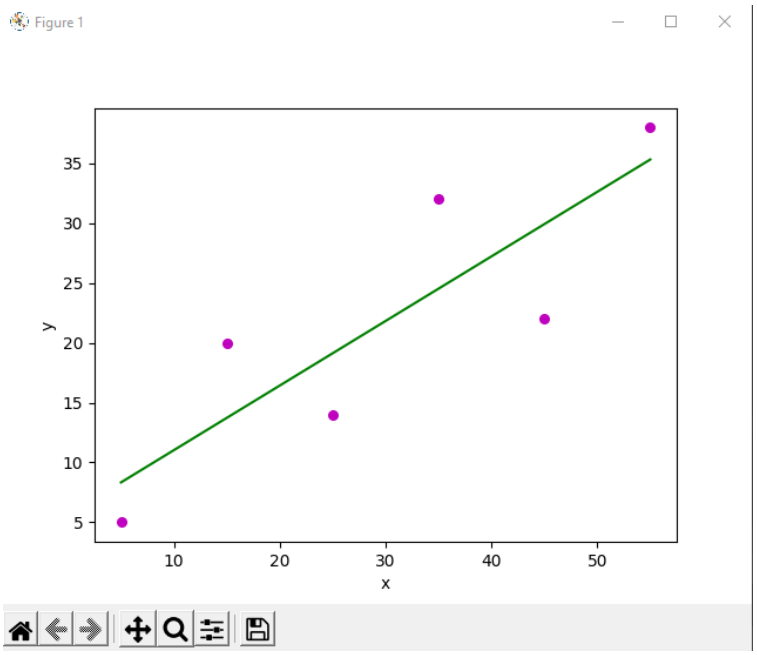
PROGRAM CODE

```
import matplotlib.pyplot as plt

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('coeffecient 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',s=30)

plt.plot(x,y_pred,color = 'g')

plt.xlabel('x')

plt.ylabel('y')
```

OUTPUT

```
[[ 5]
 [15]
 [25]
 [35]
 [45]
 [55]]
[ 5 20 14 32 22 38]
coeffecient 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 :08/12/2021**

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

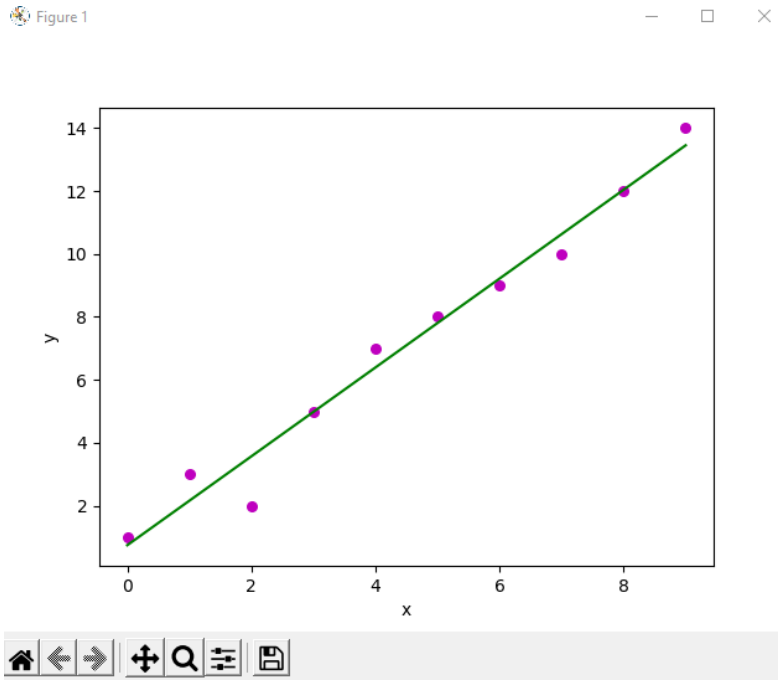
PROGRAM CODE

```
import numpy as np

import matplotlib.pyplot as plt

 x=np.array([0,1,2,3,4,5,6,7,8,9])

    y=np.array([1,3,2,5,7,8,9,10,12,14])

n = np.size(x)

n_x = np.mean(x)

n_y = np.mean(y)

SS__xy = np.sum(y*x)-n* n_y*n_x

SS__xx = np.sum(x*x)-n* n_x*n_x

b_1 = SS__xy/SS__xx

b_0 = n_y - b_1*n_x

y_pred = b_1 * x + b_0

print(y_pred)

plt.scatter(x, y, color='red')
```

```
plt.plot(x, y_pred, color='green')

plt.xlabel('X')

plt.ylabel('y')

plt.show()
```

OUTPUT

```
Estimated coefficients:
 b_0 =0.7454545454545451 \
 b_1 =1.412121212121212
```



PROGRAM NO : 08 **Date :15/12/2021**

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

PROGRAM CODE

```
import pandas

from sklearn import linear_model

df = pandas.read_csv("cars.csv")

X = df[['Weight', 'Volume']]

y = df['CO2']

regr = linear_model.LinearRegression()

regr.fit(X, y)

#predict the CO2

predictedCO2 = regr.predict([[2300, 1300]])

print(predictedCO2)
```

OUTPUT

```
C:\Users\mca\AppData\Local\Programs\Python\Python39\python.exe C:/Users/mca/Desktop/project/multipleRegression.py
[107.2087328]
```

PROGRAM NO : 09 **Date :15/12/2021**

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

from sklearn.metrics import r2_score

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)

V=reg.predict(X_test)
```

```
result=r2_score(y_test, V)

print("accuracy :", result)

print('Coefficients: ', reg.coef_)

print('Variance score:{}'.format(reg.score(X_test, y_test)))
```

OUTPUT

```
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("next.png")
target =df['species']
df1 = df.copy()
df1 = df1.drop('species', axis=1)
print(df1.shape)
```

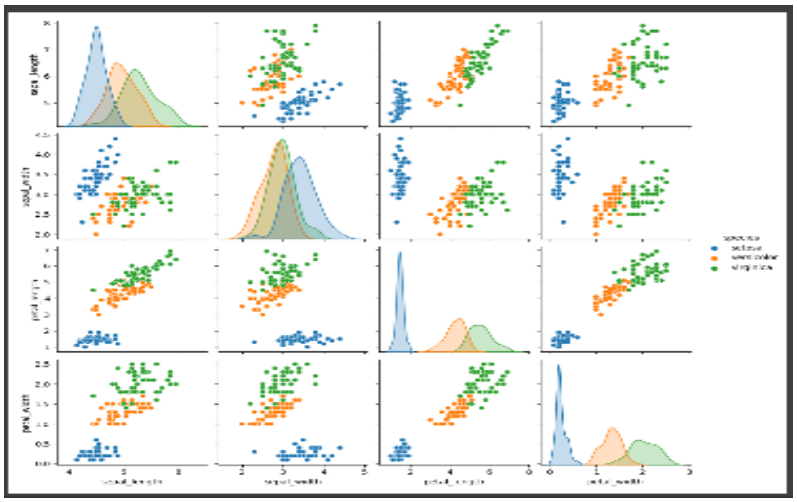
```
print(df1.head())
x=df1
print(target)
le = LabelEncoder()
target = le.fit_transform(target)
print(target)
y= target
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=
42)
print("training split input" , x_train.shape)
print("test split input",x_test.shape)
dtree=DecisionTreeClassifier()
dtree.fit(x_train, y_train)
print("decision tree classifer 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=12)
plt.savefig("two.png")
plt.figure(figsize=(20,20))
dec_tree=plot_tree(decision_tree=dtree,feature_names=df1.columns,class_names=["s
etosa","vercicolor","verginica"],filled=True ,precision=4,rounded=True)
plt.savefig("three.png")
```

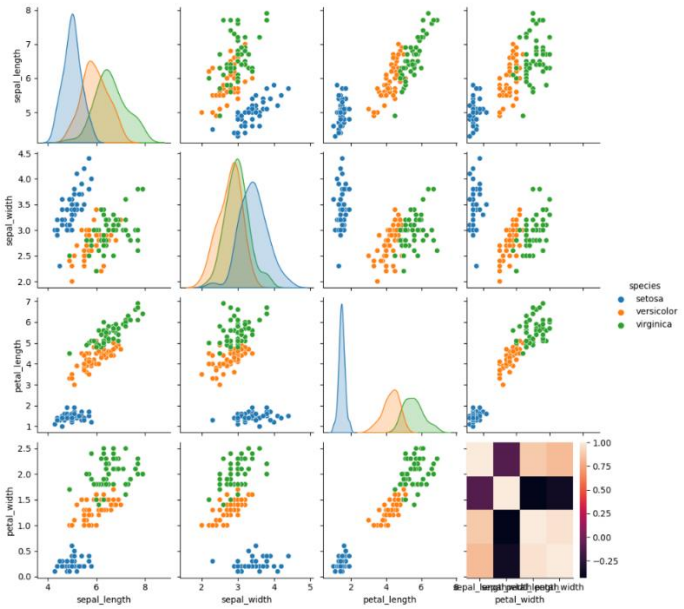
OUTPUT

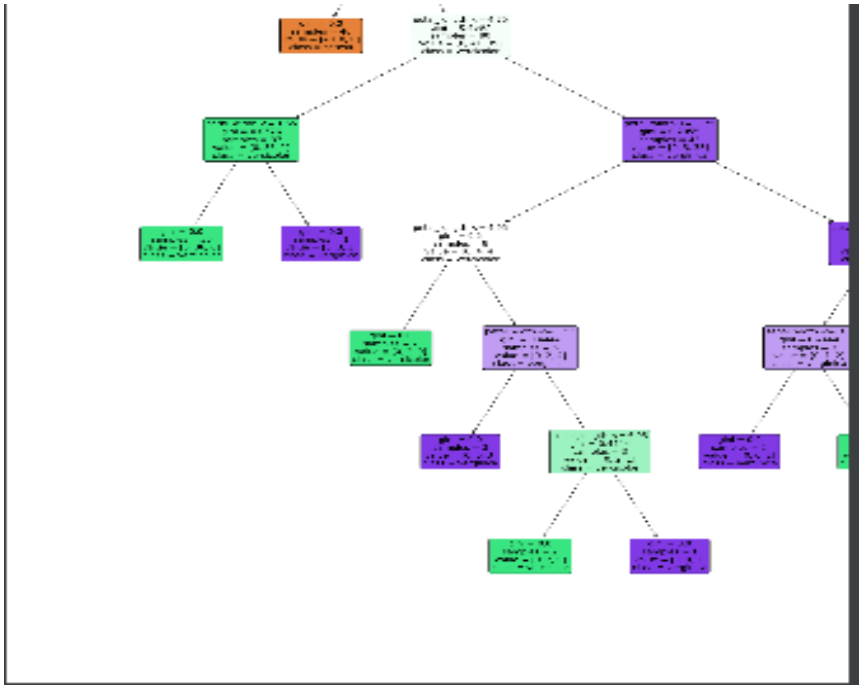
```
   sepal_length  sepal_width  petal_length  petal_width species
0           5.1          3.5           1.4          0.2  setosa
1           4.9          3.0           1.4          0.2  setosa
2           4.7          3.2           1.3          0.2  setosa
3           4.6          3.1           1.5          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)
```

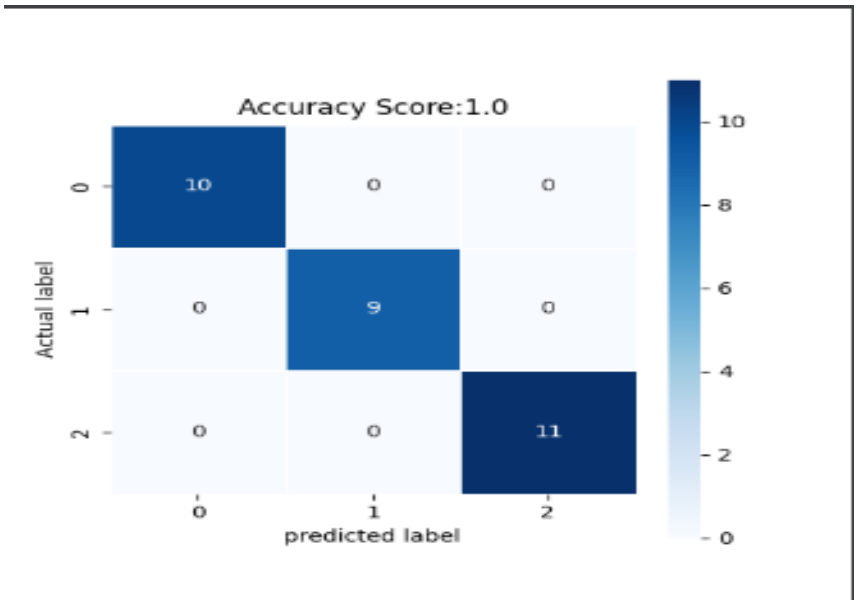
```
   sepal_length  sepal_width  petal_length  petal_width
0           5.1          3.5           1.4          0.2
1           4.9          3.0           1.4          0.2
2           4.7          3.2           1.3          0.2
3           4.6          3.1           1.5          0.2
4           5.0          3.6           1.4          0.2
0         setosa
1         setosa
2         setosa
3         setosa
4         setosa
        ...
145    virginica
146    virginica
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 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)
```

```
Training split input- (120, 4)
Testing split input- (30, 4)
Decision Tree Classifier Created
Classification-
              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
```








Accuracy Score:1.0
Actual label
predicted label
10
0
0
0
9
0
0
0
11
0
1
2
10
8
6
4
2
0

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

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

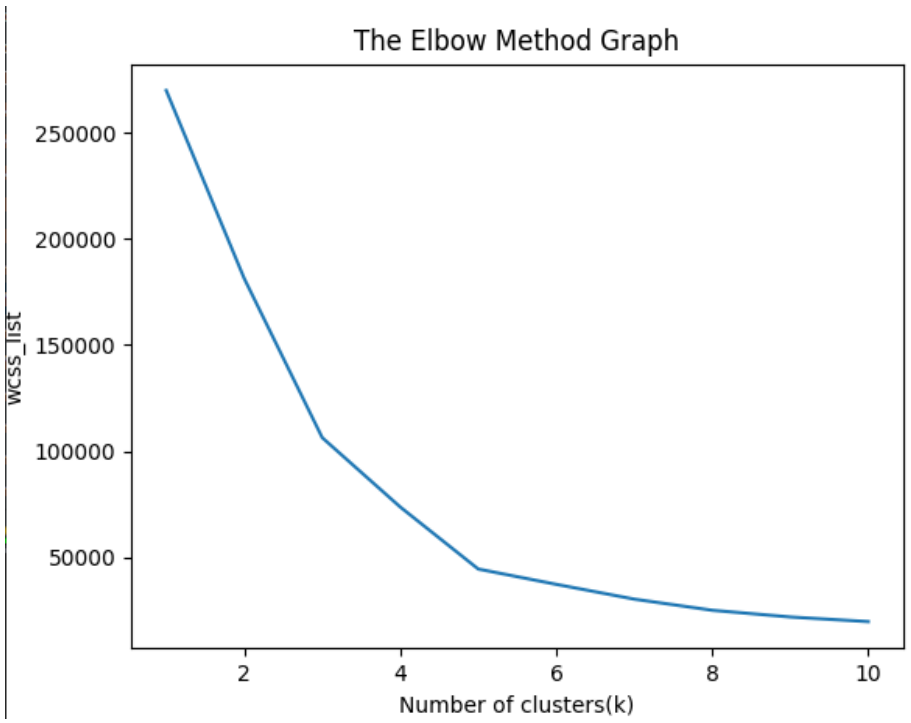
PROGRAM CODE

```
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('predict=',y_predict)

mtp.scatter(x[y_predict==0,0],x[y_predict==0,1],s=100,c='blue',label='Cluster 1')

mtp.scatter(x[y_predict==1,0],x[y_predict==1,1],s=100,c='red',label='Cluster 2')

mtp.scatter(x[y_predict==2,0],x[y_predict==2,1],s=100,c='green',label='Cluster 3')

mtp.scatter(x[y_predict==3,0],x[y_predict==3,1],s=100,c='yellow',label='Cluster 4')

mtp.scatter(x[y_predict==4,0],x[y_predict==4,1],s=100,c='magenta',label='Cluster 5')

mtp.scatter(kmeans.cluster_centers_[:,0],kmeans.cluster_centers_[:,1],s=300,c='black'
)

mtp.title('Clusters of Customer')

mtp.xlabel('Annual Income(k$)')

mtp.ylabel('Spending Score (1-100)')

mtp.legend();

mtp.show();
```

OUTPUT

```
[[ 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]
 [ 21  35]
 [ 21  66]
 [ 23  29]
 [ 23  98]
 [ 24  35]
 [ 24  73]
 [ 25   5]
 [ 25  73]
 [ 28  14]
```


The Elbow Method Graph
wcss_list
250000
200000
150000
100000
50000
2
4
6
8
10
Number of clusters(k)

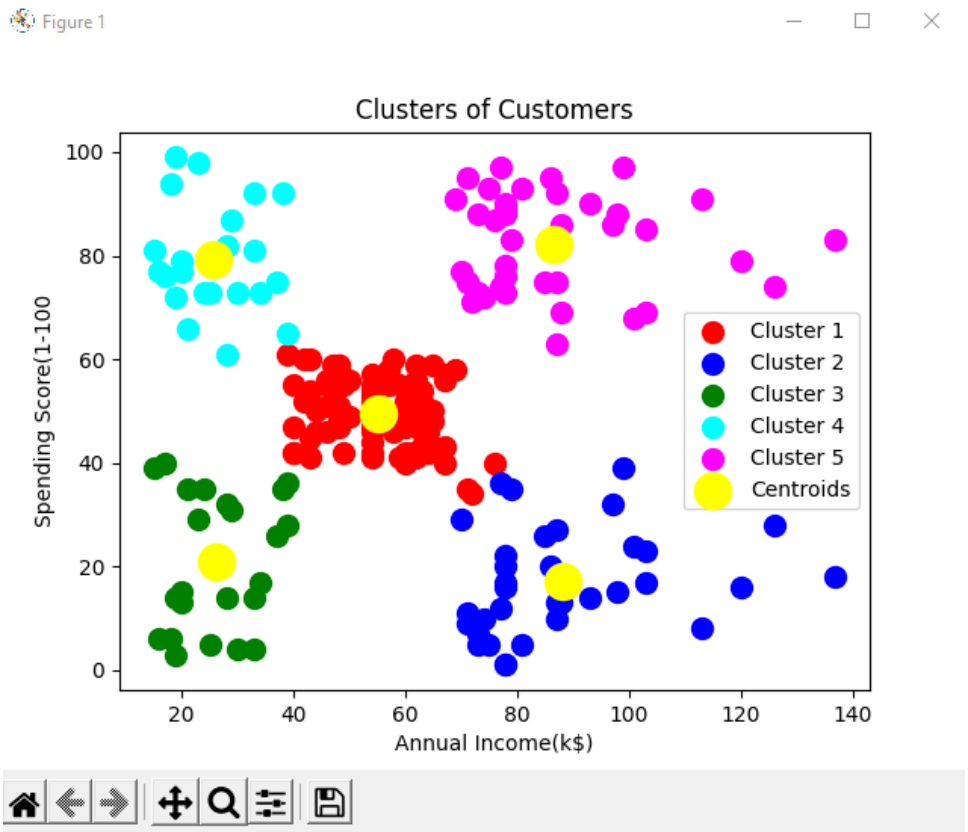


Figure 1
Clusters of Customers
Spending Score(1-100
100
80
60
40
20
0
20
40
60
80
100
120
140
Annual Income(k$)
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Cluster 5
Centroids

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 nm

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++',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=3,init='k-means++',random_state=42)

y_predict=kmeans.fit_predict(x)

print('predict=',y_predict)

mtp.scatter(x[y_predict==0,0],x[y_predict==0,1],s=100,c='blue',label='Cluster 1')

mtp.scatter(x[y_predict==1,0],x[y_predict==1,1],s=100,c='red',label='Cluster 2')

mtp.scatter(x[y_predict==2,0],x[y_predict==2,1],s=100,c='green',label='Cluster 3')

mtp.scatter(kmeans.cluster_centers_[:,0],kmeans.cluster_centers_[:,1],s=300,c='black'
)

mtp.title('Clusters of world Country')

mtp.xlabel('latitude')

mtp.ylabel('longitude')

mtp.legend();

mtp.show()
```

OUTPUT

```
[[ 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]
 [ 4.75162310e+01  1.45500720e+01]
 [-2.52743980e+01  1.33775136e+02]
 [ 1.25211100e+01 -6.99683380e+01]
 [ 4.01431050e+01  4.75769270e+01]
 [ 4.39158860e+01  1.76790760e+01]
 [ 1.31938870e+01 -5.95431980e+01]
 [ 2.36849940e+01  9.03563310e+01]
 [ 5.05038870e+01  4.46993600e+00]
 [ 1.22383330e+01 -1.56159300e+00]
 [ 4.27338830e+01  2.54858300e+01]
 [ 2.59304140e+01  5.06377720e+01]
 [-3.37305600e+00  2.99188860e+01]
 [ 9.30769000e+00  2.31583400e+00]
```

```
[ 4.13774910e+01  6.45852620e+01]
[ 4.19029160e+01  1.24533890e+01]
[ 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]]
0 0 0 1 1 0 0 1 0 0 1 1 0 2 1 0 0 1 2 0 0 0 0 0 0 1 2 1 1 1 2 0 0 0 1 1 2
0 0 0 0 0 1 1 0 2 1 1 1 0 2 0 0 0 0 0 1 1 0 1 0 0 0 0 0 0 0 2 1 2 0 0 0 0
1 0 1 0 0 0 1 0 0 1 0 0 1 1 2 0 1 0 2 2 1 0 1 0 2 0 0 0 2 2 0 0 0 0 0 1 0
2 0 0 2 1 0 1 2 2 0 1 0 2 0 1 0 2 0 0 0 0 0 0 0 0 0 0 0 2 0 0 2 2 2 2 1 0
1 0 0 2 0 1 2 0 0 2 0 2 0 1 0 0 2 2 1 2 0 1 1 1 2 2 0 0 1 1 1 0 0 2 1 0 0
0 0 2 0 0 2 0 0 0 2 0 0 0 0 0 0 0 0 1 0 1 0 0 1 0 2 0 2 0 1 2 0 0 1 0 1 2
2 0 0 0 1 1 0 0 1 1 1 1 2 2 1 1 0 0 0 0 0 0]
```

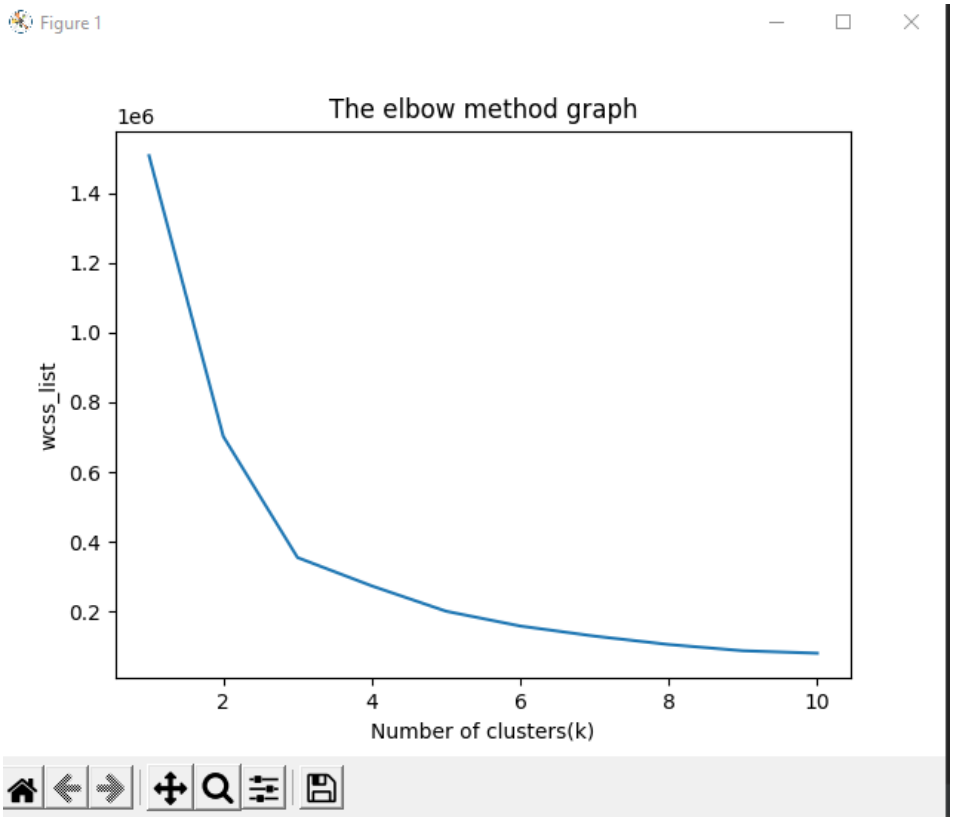


Figure 1
The elbow method graph
1e6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
wcss_list
2
4
6
8
10
Number of clusters(k)

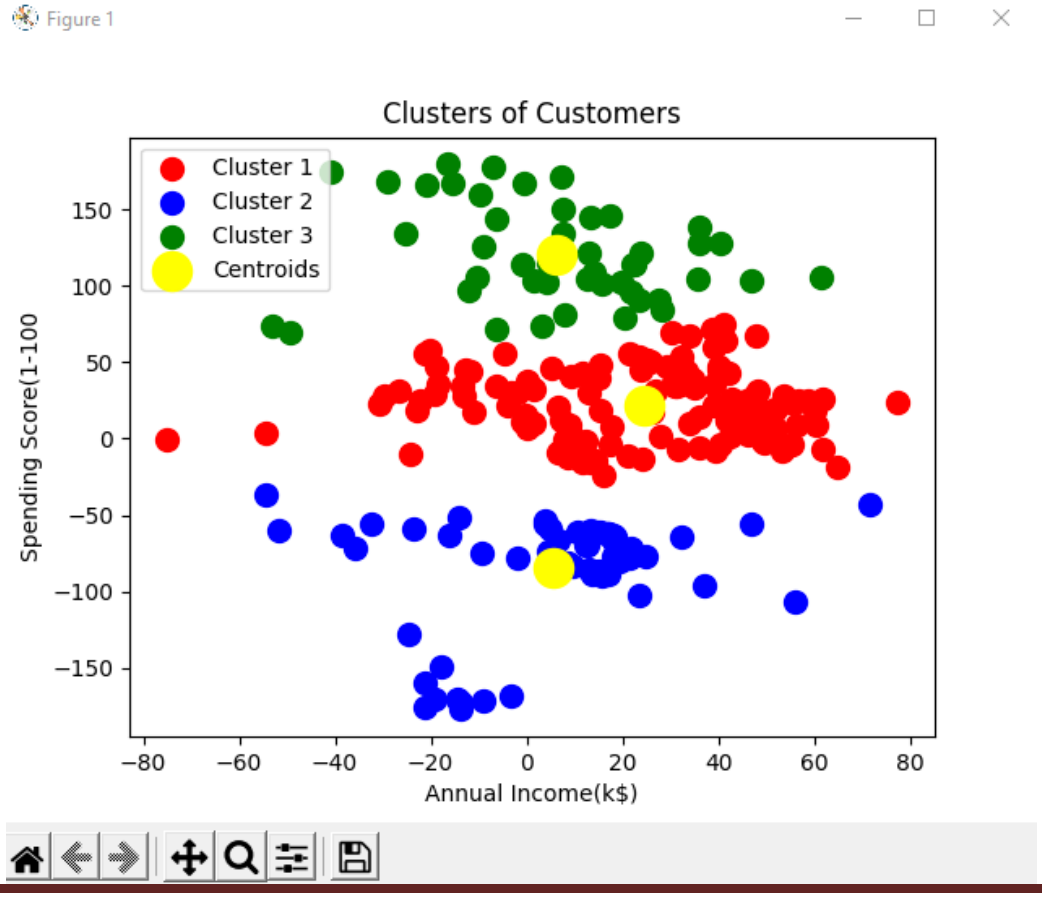


Figure 1
Clusters of Customers
Cluster 1
Cluster 2
Cluster 3
Centroids
150
100
50
0
−50
−100
−150
Spending Score(1-100
−80
−60
−40
−20
0
20
40
60
80
Annual Income(k$)

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

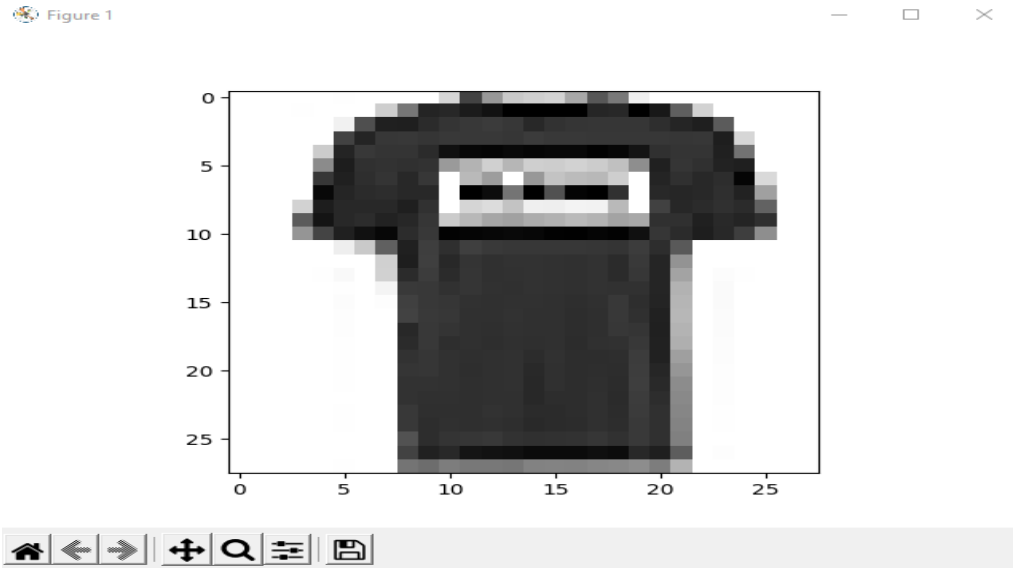
```
import numpy as np

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)
```

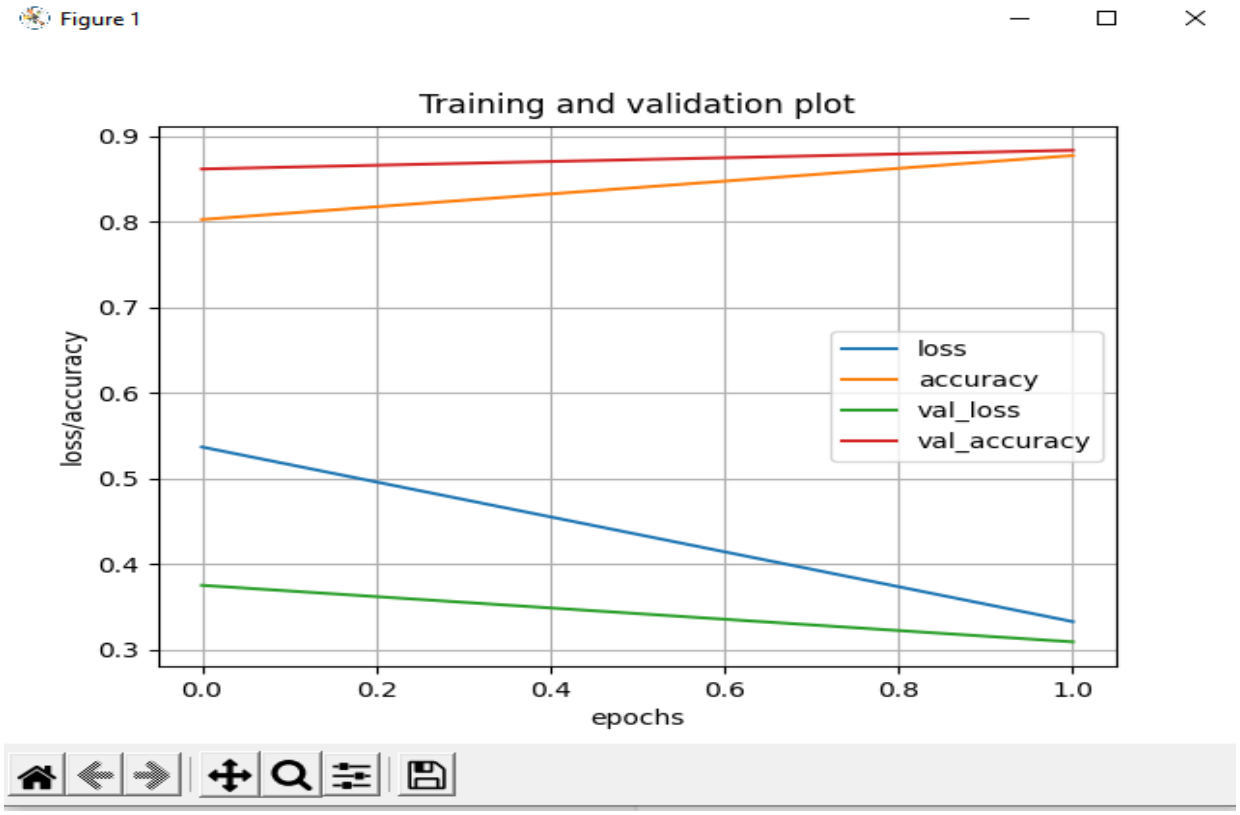
```
class_names = ['T-Shirt/Top', 'Trouser', 'Pullover', 'Dress', 'Coat', 'Sandal', 'Shirt',
'Sneaker', '8ag', 'Ankle Boot']

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






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

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

flatten (Flatten)           (None, 288)               0

dense (Dense)               (None, 128)               36992

dense_1 (Dense)             (None, 64)                8256

dense_2 (Dense)             (None, 10)                650

=================================================================
Total params: 84,458
Trainable params: 84,458
Non-trainable params: 0
_________________________________________________________________
Epoch 1/2
1688/1688 [==============================] - 104s 61ms/step - loss: 0.5369 - accuracy: 0.8024 - val_loss: 0.3755 - val_accuracy: 0.8613
Epoch 2/2
1688/1688 [==============================] - 103s 61ms/step - loss: 0.3332 - accuracy: 0.8770 - val_loss: 0.3096 - val_accuracy: 0.8832
```

PROGRAM NO : 14 **Date :16/02/2022**

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

PROGRAM CODE

```
import requests

import lxml

from bs4 import BeautifulSoup

#import beautifulsoup4

url = "https://www.rottentomatoes.com/top/bestofrt/"

headers = { 'User-Agents' : 'Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/63.0.3239.132 Safari/537.36 QIHU 36OSE'}

f = requests.get(url, headers = headers)

movies_list = []

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_list.append(urls)
print(movies_list)
num +=1
movie_url=urls
#movie_url=movies_lst
movie_f=requests.get(movie_url,headers=headers)


movie_soup=BeautifulSoup(movie_f.content,'lxml')
movie_content=movie_soup.find('div',{
    'class':'movie_synopsis clamp clamp-6 js-clamp'
})
print(num,urls,'\n','Movie:' + anchor.string.strip())
print('Movie info:' + movie_content.string.strip())
```

OUTPUT


webcrawler
C:\Users\ajcemca\PycharmProjects\pythonProject2\venv\Scripts\python.exe C:/Users/ajcemca/PycharmProjects/pythonProject2/webcrawler.py
[
It Happened One Night (1934),
Citizen Kane (1941),
The Wizard of Oz (1939),
Modern Times (1936),
Black Panther (2018),
Parasite (Gisaengchung) (2019),
Avengers: Endgame (2019),
Casablanca (1942),
Knives Out (2019),
Zootopia (2016),
Alien (1979),
King Kong (1933),
Shadow of a Doubt (1943),
Call Me by Your Name (2018),
Psycho (1960),
1917 (2020),
L.A. Confidential (1997),
The Florida Project (2017),
War for the Planet of the Apes (2017),
Paddington 2 (2018),
A Hard Day's Night (1964),
Widows (2018),
Never Rarely Sometimes Always (2020),
Baby Driver (2017),
Spider-Man: Homecoming (2017),
The Godfather, Part II (1974),
The Battle of Algiers (La Battaglia di Algeri) (1967)]
['https://www.rottentomatoes.com/m/it_happened_one_night', 'https://www.rottentomatoes.com/m/citizen_kane', 'https://www.rottentomatoes.com/m/the_wizard_of_o
1 https://www.rottentomatoes.com/m/the_battle_of_algiers
Movie:The Battle of Algiers (La Battaglia di Algeri) (1967)
Movie info:Paratrooper commander Colonel Mathieu (Jean Martin), a former French Resistance fighter during World War II, is sent to 1950s Algeria to reinforce
Process finished with exit code 0

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}:{link["href"]}\n')
                crawler(new_link)
            except:
                continue

crawler('https://en.wikipedia.org')
```

OUTPUT

```
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
Basilica - Wikipedia; Basilica; /wiki/Basilicas_in_the_Catholic_Church
Basilicas in the Catholic Church - Wikipedia; Basilicas in the Catholic Church; /wiki/List_of_Catholic_basilicas
List of Catholic basilicas - Wikipedia; List of Catholic basilicas; /wiki/Catholic_Church
```

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

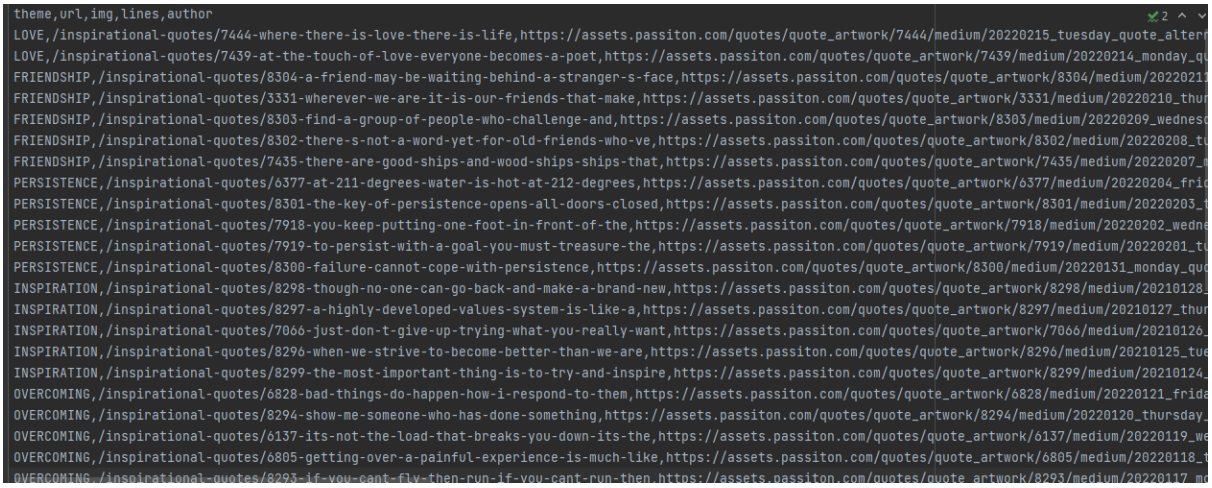
AIM : Program to implement scrap of any website.

PROGRAM CODE

```
import requests
from bs4 import BeautifulSoup
import csv
URL = "http://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)
```

OUTPUT



```
theme,url,img,lines,author
LOVE,/inspirational-quotes/7444-where-there-is-love-there-is-life,https://assets.passiton.com/quotes/quote_artwork/7444/medium/20220215_tuesday_quote_alterr
LOVE,/inspirational-quotes/7439-at-the-touch-of-love-everyone-becomes-a-poet,https://assets.passiton.com/quotes/quote_artwork/7439/medium/20220214_monday_qu
FRIENDSHIP,/inspirational-quotes/8304-a-friend-may-be-waiting-behind-a-stranger-s-face,https://assets.passiton.com/quotes/quote_artwork/8304/medium/2022021
FRIENDSHIP,/inspirational-quotes/3331-wherever-we-are-it-is-our-friends-that-make,https://assets.passiton.com/quotes/quote_artwork/3331/medium/20220210_thur
FRIENDSHIP,/inspirational-quotes/8303-find-a-group-of-people-who-challenge-and,https://assets.passiton.com/quotes/quote_artwork/8303/medium/20220209_wednes
FRIENDSHIP,/inspirational-quotes/8302-there-s-not-a-word-yet-for-old-friends-who-ve,https://assets.passiton.com/quotes/quote_artwork/8302/medium/20220208_tu
FRIENDSHIP,/inspirational-quotes/7435-there-are-good-ships-and-wood-ships-ships-that,https://assets.passiton.com/quotes/quote_artwork/7435/medium/20220207_
PERSISTENCE,/inspirational-quotes/6377-at-211-degrees-water-is-hot-at-212-degrees,https://assets.passiton.com/quotes/quote_artwork/6377/medium/20220204_frid
PERSISTENCE,/inspirational-quotes/8301-the-key-of-persistence-opens-all-doors-closed,https://assets.passiton.com/quotes/quote_artwork/8301/medium/20220203_
PERSISTENCE,/inspirational-quotes/7918-you-keep-putting-one-foot-in-front-of-the,https://assets.passiton.com/quotes/quote_artwork/7918/medium/20220202_wedne
PERSISTENCE,/inspirational-quotes/7919-to-persist-with-a-goal-you-must-treasure-the,https://assets.passiton.com/quotes/quote_artwork/7919/medium/20220201_tu
PERSISTENCE,/inspirational-quotes/8300-failure-cannot-cope-with-persistence,https://assets.passiton.com/quotes/quote_artwork/8300/medium/20220131_monday_quo
INSPIRATION,/inspirational-quotes/8298-though-no-one-can-go-back-and-make-a-brand-new,https://assets.passiton.com/quotes/quote_artwork/8298/medium/20210128_
INSPIRATION,/inspirational-quotes/8297-a-highly-developed-values-system-is-like-a,https://assets.passiton.com/quotes/quote_artwork/8297/medium/20210127_thur
INSPIRATION,/inspirational-quotes/7066-just-don-t-give-up-trying-what-you-really-want,https://assets.passiton.com/quotes/quote_artwork/7066/medium/20210126_
INSPIRATION,/inspirational-quotes/8296-when-we-strive-to-become-better-than-we-are,https://assets.passiton.com/quotes/quote_artwork/8296/medium/20210125_tue
INSPIRATION,/inspirational-quotes/8299-the-most-important-thing-is-to-try-and-inspire,https://assets.passiton.com/quotes/quote_artwork/8299/medium/20210124_
OVERCOMING,/inspirational-quotes/6828-bad-things-do-happen-how-i-respond-to-them,https://assets.passiton.com/quotes/quote_artwork/6828/medium/20220121_frida
OVERCOMING,/inspirational-quotes/8294-show-me-someone-who-has-done-something,https://assets.passiton.com/quotes/quote_artwork/8294/medium/20220120_thursday_
OVERCOMING,/inspirational-quotes/6137-its-not-the-load-that-breaks-you-down-its-the,https://assets.passiton.com/quotes/quote_artwork/6137/medium/20220119_we
OVERCOMING,/inspirational-quotes/6805-getting-over-a-painful-experience-is-much-like,https://assets.passiton.com/quotes/quote_artwork/6805/medium/20220118_t
OVERCOMING,/inspirational-quotes/8293-if-you-cant-fly-then-run-if-you-cant-run-then,https://assets.passiton.com/quotes/quote_artwork/8293/medium/20220117_m
```

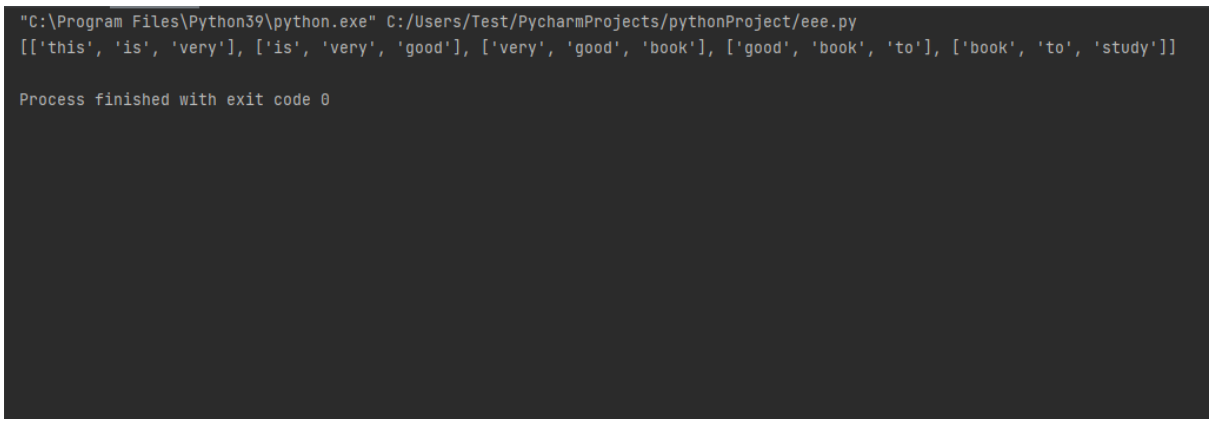
PROGRAM NO : 17 **Date :16/02/2022**

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+1 + WordsToCombine])
        return output
x=generate_ngrams(text='this is very good book to study',WordsToCombine=3)
print(x)
```

OUTPUT



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

nltk.download('punkt')

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

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. " \
    "Sukanya 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

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

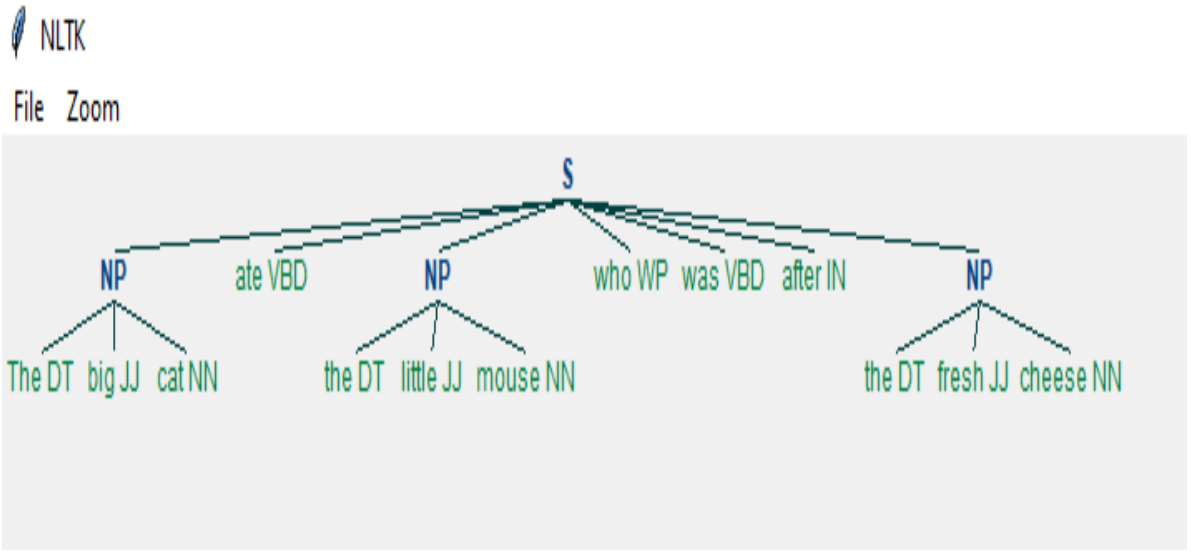
AIM : Python program which performs Natural language processing which perform Chunking.

PROGRAM CODE

```
import nltk
nltk.download()
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\mca\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/mca/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', 'I
(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))
```



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

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

PROGRAM CODE

```
import nltk
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 Ramayana.'"


tokenized=nltk.sent_tokenize(sample_text)
for i in tokenized:
    words=nltk.word_tokenize(i)
    #print(words)
    tagged_words=nltk.pos_tag(words)
    #print(tagged_words)
    chunkGram=r"""VB: {}"""
    chunkParser=nltk.RegexpParser(chunkGram)
    chunked=chunkParser.parse(tagged_words)
    print(chunked)
    chunked.draw()
```

OUTPUT

```
(S
  Rama/NNP
  killed/VBD
  Ravana/NNP
  to/TO
  save/VB
  Sita/NNP
  from/IN
  Lanka/NNP
  ./.)
```