PROGRAM NO: 1 **DATE: 24/11/2021**

AIM: PERFORM ALL MATRIX OPERATIONS USING PYTHON [USING NUMPY]

PROGRAM

```
import numpy as np
x = np.array([[4, 3], [12, 6]])
y = np.array([[14, 5], [8, 10]])
print ("Addition of two matrices: ")
print (np.add(x,y))

print ("Subtraction of two matrices : ")
print (np.subtract(x,y))

print ("Matrix Division : ")
print (np.divide(x,y))

print ("Multiplication of two matrices: ")
print (np.multiply(x,y))

print ("The product of two matrices : ")
print (np.dot(x,y))

print ("square root is : ")
print (np.sqrt(x))
print ("The summation of elements : ")
print (np.sum(y))

print ("The column wise summation  : ")
print (np.sum(y,axis=0))

print ("The row wise summation: ")
print (np.sum(y,axis=1))
print ("Matrix transposition : ")
print (x.T)
```

OUTPUT

```
C:\Users\ajcemca\PycharmProjects\pythonProject\venv\Scripts\python.exe
Addition of two matrices:
[[18  8]
 [20 16]]
Subtraction of two matrices :
[[-10  -2]
 [  4  -4]]
Matrix Division :
[[0.28571429 0.6       ]
 [1.5        0.6       ]]
Multiplication of two matrices:
[[56 15]
 [96 60]]
The product of two matrices :
[[ 80  50]
 [216 120]]
square root is :
[[2.         1.73205081]
 [3.46410162 2.44948974]]
The summation of elements :
37
The column wise summation  :
[22 15]
```

```
Multiplication of two matrices:
[[56 15]
 [96 60]]
The product of two matrices :
[[ 80  50]
 [216 120]]
square root is :
[[2.         1.73205081]
 [3.46410162 2.44948974]]
The summation of elements :
37
The column wise summation  :
[22 15]
The row wise summation:
[19 18]
Matrix transposition :
[[ 4 12]
 [ 3  6]]

Process finished with exit code 0
```

RESULT: THE PROGRAM HAS BEEN EXECUTED AND OUTPUT VERIFIED

PROGRAM NO: 2 **DATE: 01/12/2021**

AIM: PERFORM SVD (SINGULAR VALUE DECOMPOSITION)

PROGRAM

```
from numpy import array
from scipy.linalg import svd
A= array([[6,5,2,4,5], [8,1,3,7,8], [4,2,7,10,9], [4,8,7,3,2], [9,10,7,1,5]])
print(A)
X, Y, Z =svd(A)
print("\nDecomposition: ", X)
print("\nInverse Matrix: ",Y)
print("\nTranspose of Matrix: ",Z)
```

OUTPUT

```
C:\Users\ajcemca\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/ajcemca/P
[[ 6  5  2  4  5]
 [ 8  1  3  7  8]
 [ 4  2  7 10  9]
 [ 4  8  7  3  2]
 [ 9 10  7  1  5]]

Decomposition:  [[-0.35795072 -0.02542133 -0.35295795 -0.81761681 -0.27955048]
 [-0.44565919  0.43286247 -0.5080382   0.21004848  0.55838589]
 [-0.51111683  0.54319395  0.53762478  0.10442387 -0.37915059]
 [-0.37713702 -0.41338921  0.52445691 -0.25045479  0.59084209]
 [-0.51940969 -0.58824615 -0.23069986  0.46233152 -0.34235535]]

Inverse Matrix:  [27.79740326 11.37180805  5.80585429  1.97387889  1.17567493]

Transpose of Matrix:  [[-0.50151017 -0.41258672 -0.42833223 -0.40699508 -0.47869223]
 [-0.12879435 -0.68568089 -0.17247434  0.57439219  0.39189278]
 [-0.69068651  0.11903128  0.61827978  0.30156104 -0.18861269]
 [ 0.17809954 -0.53169979  0.61250768 -0.52937497  0.17370194]
 [ 0.47236877 -0.25048114  0.1713142   0.36503657 -0.74264842]]

Process finished with exit code 0
```

RESULT: THE PROGRAM HAS BEEN EXECUTED AND OUTPUT VERIFIED

PROGRAM NO: 3 **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 BUILT FUNCTION)

PROGRAM

```
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

irisData = load_iris()

A = irisData.data
B = irisData.target

A_train, A_test, B_train, B_test = train_test_split(
    A, B, test_size=.3, random_state=10)

knn = KNeighborsClassifier(n_neighbors=2)

knn.fit(A_train, B_train)

print(knn.predict(A_test))

#Finding Accuracy of the Algorithm

W = knn.predict(A_test)
Q = accuracy_score(B_test, W)
print("Accuracy : " , Q)
```

OUTPUT

```
C:\Users\ajcemca\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/ajc
[1 1 0 1 0 1 0 0 2 0 0 2 1 2 1 2 1 0 2 2 0 1 0 0 0 0 1 2 0 2 1 2 0 0 0 1 0
 1 1 2 0 2 2 2 1 2 0 2 1 2 0 0 2 0 2 2 2 0 0 1 2 1 0 2 2 2 0 2 1 1 2 1 0 0
 2 1 0 2 0 0 1 0 2 0 2 1 2 0 2 0 2 0 0 2 2 1 0 1 0 2 2 2 0 0 0 2 2 1 1 1 2
 1 1 1 2 1 1 2 1 2 1 0 1 0 1 0 2 2 1 0 1 1 1 1 2 0 0 1 0 1 2 0 2 1 1 2 2 1
 1]

Process finished with exit code 0
```

```
C:\Users\ajcemca\PycharmProjects\pythonProject\venv\Scripts\python.exe C:
[1]
Accuracy: 1.0

Process finished with exit code 0
```

RESULT: THE PROGRAM HAS BEEN EXECUTED AND OUTPUT VERIFIED

PROGRAM NO: 4 **DATE: 01/12/2021**

AIM: PROGRAM TO IMPLEMENT K-NN CLASSIFICATION USING ANY RANDOM DATASET WITHOUT USING IN BUILT PACKAGES

PROGRAM

```
from math import sqrt
#calculate the euclidean distance between two vectors
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 = [[2.7810836,2.550537003,0],
```

```
    [1.465489372,2.362125076,0],
    [3.396561688,4.400293529,0],
    [1.38807019,1.850220317,0],
    [3.06407232,3.005305973,0],
    [7.627531214,2.759262235,1],
    [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], 3)
print('Expected %d, Got %d.' % (dataset[0][-1], prediction))
```

OUTPUT

```
C:\Users\ajcemca\PycharmProjects\pythonProject\ven
Expected 0, Got 0.

Process finished with exit code 0
```

RESULT: THE PROGRAM HAS BEEN EXECUTED AND OUTPUT VERIFIED

PROGRAM NO: 5 **DATE: 08/12/2021**

AIM: PROGRAM TO IMPLEMENT NAIVE BAYES ALGORITHM USING ANY STANDARD DATASET AVAILABLE IN THE PUBLIC DOMAIN AND FIND THE ACCURACY OF THE ALGORITHM.

PROGRAM

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

# importing dataset
dataset = pd.read_csv("nba.csv")
a = dataset.iloc[:, [2, 3]].values
b = dataset.iloc[:, -1].values

# splitting into test and train dataset

from sklearn.model_selection import train_test_split
a_train, a_test, b_train, b_test = train_test_split(a, b, test_size=0.20, random_state=0)

# Feature scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
a_train = sc.fit_transform(a_train)
a_test = sc.transform(a_test)
print(a_train)
print(a_test)

# training the naive bayes model on the training set
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(a_train, b_train)

# predicting the test set results
b_pred = classifier.predict(a_test)
```

```
print(b_pred)


# making confusion matrix
from sklearn.metrics import confusion_matrix, accuracy_score
ac = accuracy_score(b_test, b_pred)
co = confusion_matrix(b_test, b_pred)
print(ac)
print(co)
```

OUTPUT

```
 [ 2.11737157e+00  3.78719297e-01]
 [-1.38221530e+00  5.52551726e-01]
 [-1.09058306e+00 -3.45582490e-01]
 [ 1.73156642e-01 -6.64275277e-01]
 [ 3.67578135e-01  2.08236764e-03]
 [-6.04529329e-01  2.31984809e+00]
 [-3.12897090e-01  2.04886868e-01]
 [-1.57663679e+00 -2.00722133e-01]
 [ 6.59210374e-01 -1.38857706e+00]
 [-1.09058306e+00  5.52551726e-01]
 [-1.96547978e+00  3.49747226e-01]
 [ 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]]

Process finished with exit code 0
```

RESULT: THE PROGRAM HAS BEEN EXECUTED AND OUTPUT VERIFIED

PROGRAM NO: 6 **DATE: 08/12/2021**

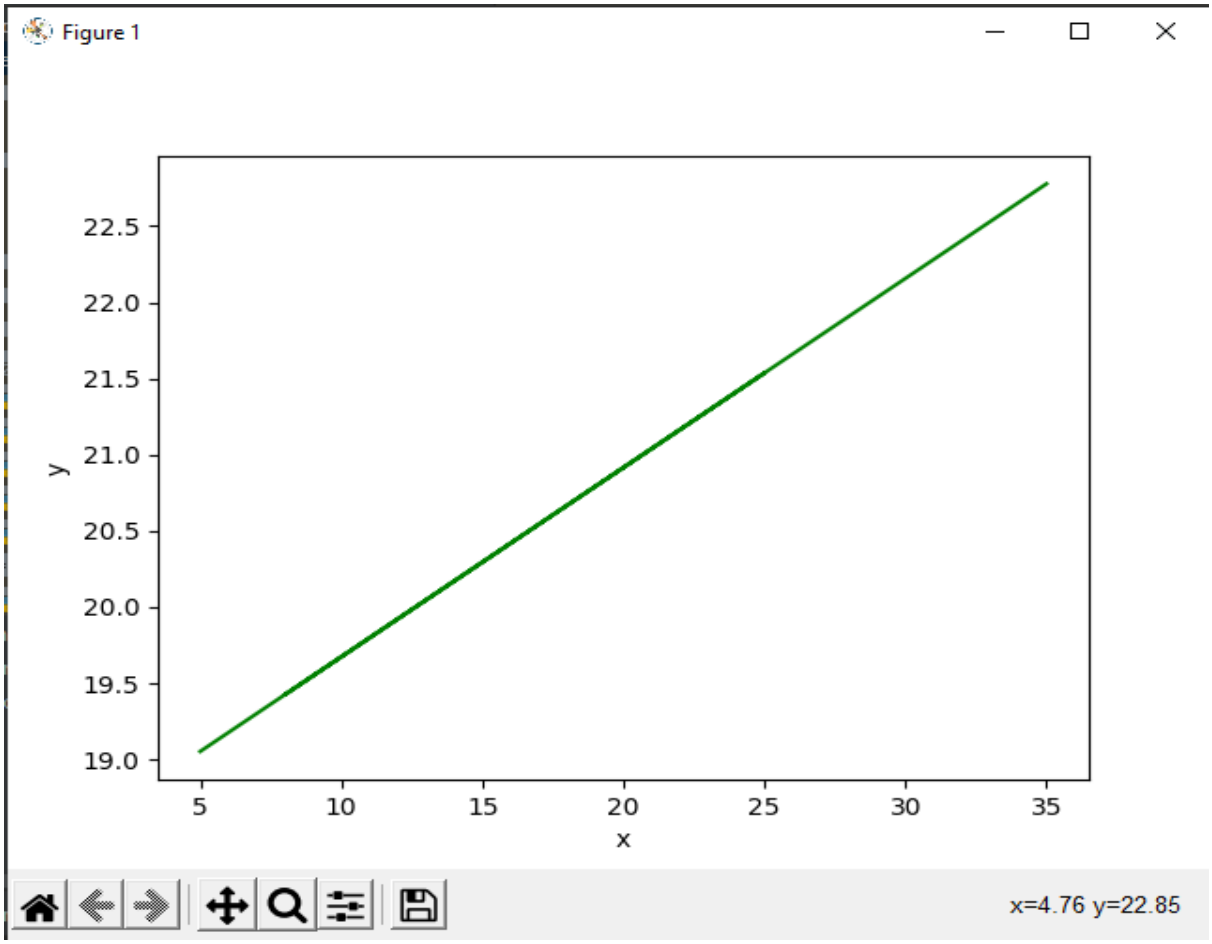
AIM: PROGRAM TO IMPLEMENT LINEAR REGRESSION TECHNIQUES USING ANY STANDARD DATASET AVAILABLE IN THE PUBLIC DOMAIN AND EVALUATE.

PROGRAM

```
import matplotlib.pyplot as plt
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
print("\nLinear Regression \n" )
x=np.array([5,15,20,25,8,35]).reshape((-1,1))
y=np.array([5,20,30,14,33,22])
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, sep='\n' )
plt.scatter(x,y, color="m",
        marker="0", sep=30)
plt.plot(x, y_pred, color="g" )
plt.xlabel('x')
plt.ylabel('y')

plt.show()
```

OUTPUT

```
Linear Regression

[[ 5]
 [15]
 [20]
 [25]
 [ 8]
 [35]]
[ 5 20 30 14 33 22]
Coefficient of determination :  0.017997935807665955
Intercept :  18.431182795698927
slope :  [0.12419355]

Process finished with exit code 0
```



RESULT: THE PROGRAM HAS BEEN EXECUTED AND OUTPUT VERIFIED

PROGRAM NO: 7 **DATE: 08/12/2021**

AIM: PROGRAM TO IMPLEMENT LINEAR REGRESSION TECHNIQUES USING ANY STANDARD DATASET AVAILABLE IN THE PUBLIC DOMAIN AND EVALUATE (WITHOUT USING IN BUILT FUNCTION).

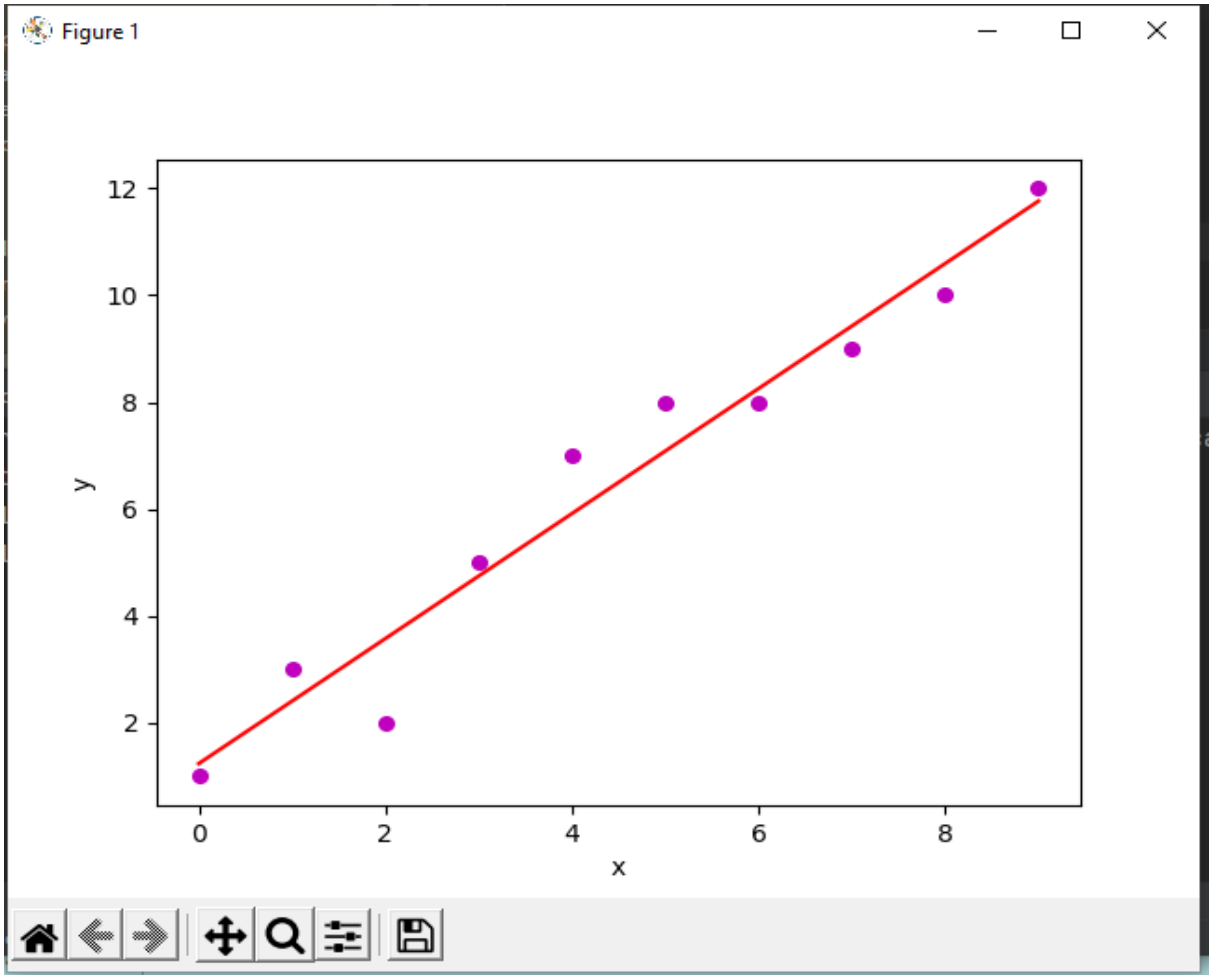
PROGRAM

```
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
    return (b_0, b_1)


def plot_regression_line(x, y, b):
    # plotting the actual points as scatter plot
    plt.scatter(x, y, color="m", marker="o", s=30)

    # predicted response vector
    y_pred = b[0] + b[1] * x

    # plotting the regression line
    plt.plot(x, y_pred, color="r")

    # putting labels
    plt.xlabel('x')
    plt.ylabel('y')
```

```
    # function to show plot
    plt.show()


def main():
    # observations
    x = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
    y = np.array([1, 3, 2, 5, 7, 8, 8, 9, 10, 12])

    # estimating coefficients
    b = estimate_coef(x, y)
    print("Estimated coefficients are:\nb_0 = {}  \
        \nb_1 = {}".format(b[0], b[1]))

    # plotting regression line
    plot_regression_line(x, y, b)


if __name__ == "__main__":
    main()
```

OUTPUT

```
Estimated coefficients are:
b_0 = 1.2363636363636363
b_1 = 1.1696969696969697
```



RESULT: THE PROGRAM HAS BEEN EXECUTED AND OUTPUT VERIFIED

PROGRAM NO: 8 **DATE: 15/12/2021**

AIM: PROGRAM TO IMPLEMENT MULTIPLE LINEAR REGRESSION TECHNIQUES USING ANY STANDARD DATASET AVAILABLE IN THE PUBLIC DOMAIN AND EVALUATE.

PROGRAM

```
import pandas as pd

df = pd.read_csv("cars.csv")
X = df[['Weight', 'Volume']]
y = df['CO2']
from sklearn import linear_model
regr = linear_model.LinearRegression()
regr.fit(X, y)
predictedCO2 = regr.predict([[2300, 1300]])
print(predictedCO2)
```

OUTPUT

```
C:\Users\ajcemca\PycharmProjects\pythonProject\venv\Scripts\python.exe "
[107.2087328]

Process finished with exit code 0
```

RESULT: THE PROGRAM HAS BEEN EXECUTED AND OUTPUT VERIFIED

PROGRAM NO: 9 **DATE: 15/12/2021**

AIM: PROGRAM TO IMPLEMENT MULTIPLE LINEAR REGRESSION TECHNIQUES USING BOSTON DATASET AVAILABLE IN THE PUBLIC DOMAIN AND EVALUATE ACCURACY AND PLOTTING POINT.

PROGRAM

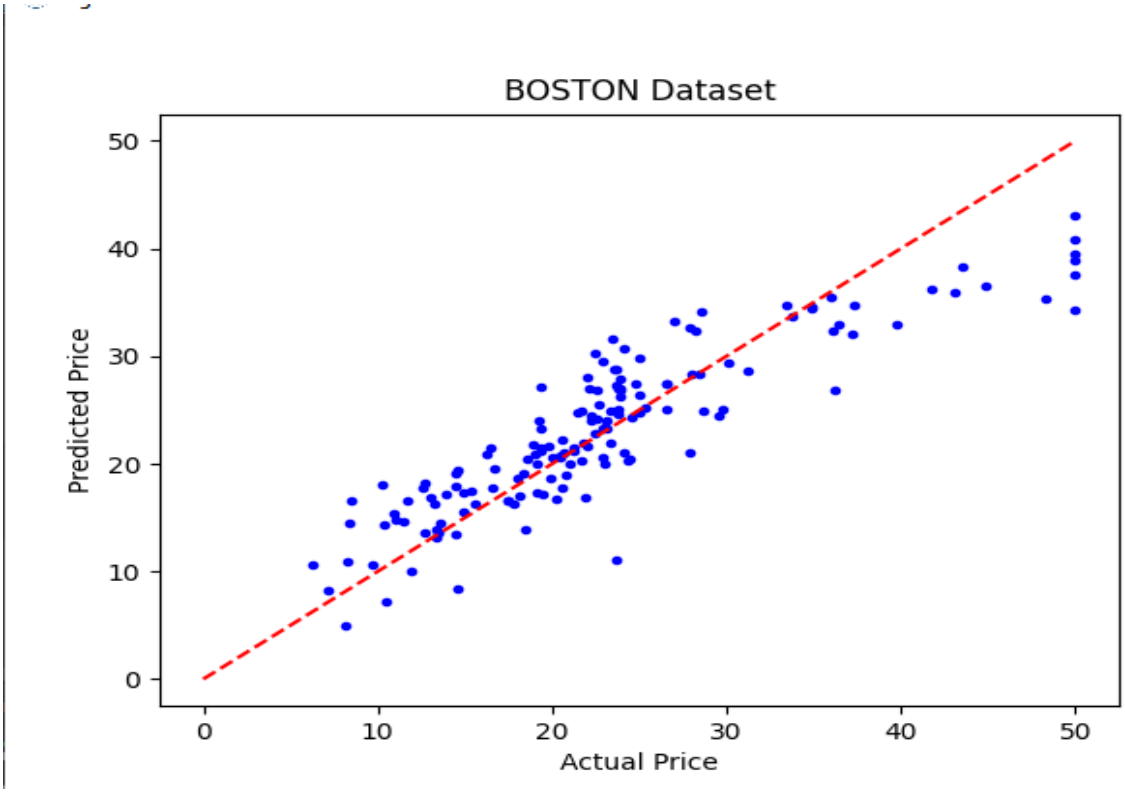
#Accuracy and plotting

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn import datasets, linear_model, metrics
from sklearn.metrics import  mean_squared_error, r2_score

#load the boston dataset
boston = datasets.load_boston(return_X_y=False)

#defining feature matrix(x) and response vector(y)
x = boston.data
y = boston.target

#splitting x and y into training and testing sets
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size=0.3,
                                   random_state=1)

reg = linear_model.LinearRegression()
reg.fit(x_train, y_train)
predicted = reg.predict(x_test)

#regression coefficient
print('coefficient: ', reg.coef_)

expected = y_test

# Plot a graph for expected and predicted values
```

```
plt.title( 'BOSTON Dataset')
plt.scatter(expected,predicted,c='b',marker='.',s=36)
plt.plot([0, 50], [0, 50], '--r')
plt.xlabel('Actual Price')
plt.ylabel('Predicted Price')
plt.show()
```

OUTPUT

```
C:\Users\ajcemca\PycharmProjects\pythonProject\venv\Scripts\python.exe "C:/Users/a
coefficient:  [-9.85424717e-02  6.07841138e-02  5.91715401e-02  2.43955988e+00
 -2.14699650e+01  2.79581385e+00  3.57459778e-03 -1.51627218e+00
  3.07541745e-01 -1.12800166e-02 -1.00546640e+00  6.45018446e-03
 -5.68834539e-01]
variance score :0.7836295385076291

Process finished with exit code 0
```



RESULT: THE PROGRAM HAS BEEN EXECUTED AND OUTPUT VERIFIED

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

AIM: PROGRAM TO IMPLEMENT DECISION TREES USING ANY STANDARD DATASET AVAILABLE IN THE PUBLIC DOMAIN AND FIND THE ACCURACY OF THE ALGORITHM.

PROGRAM

```
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())
```

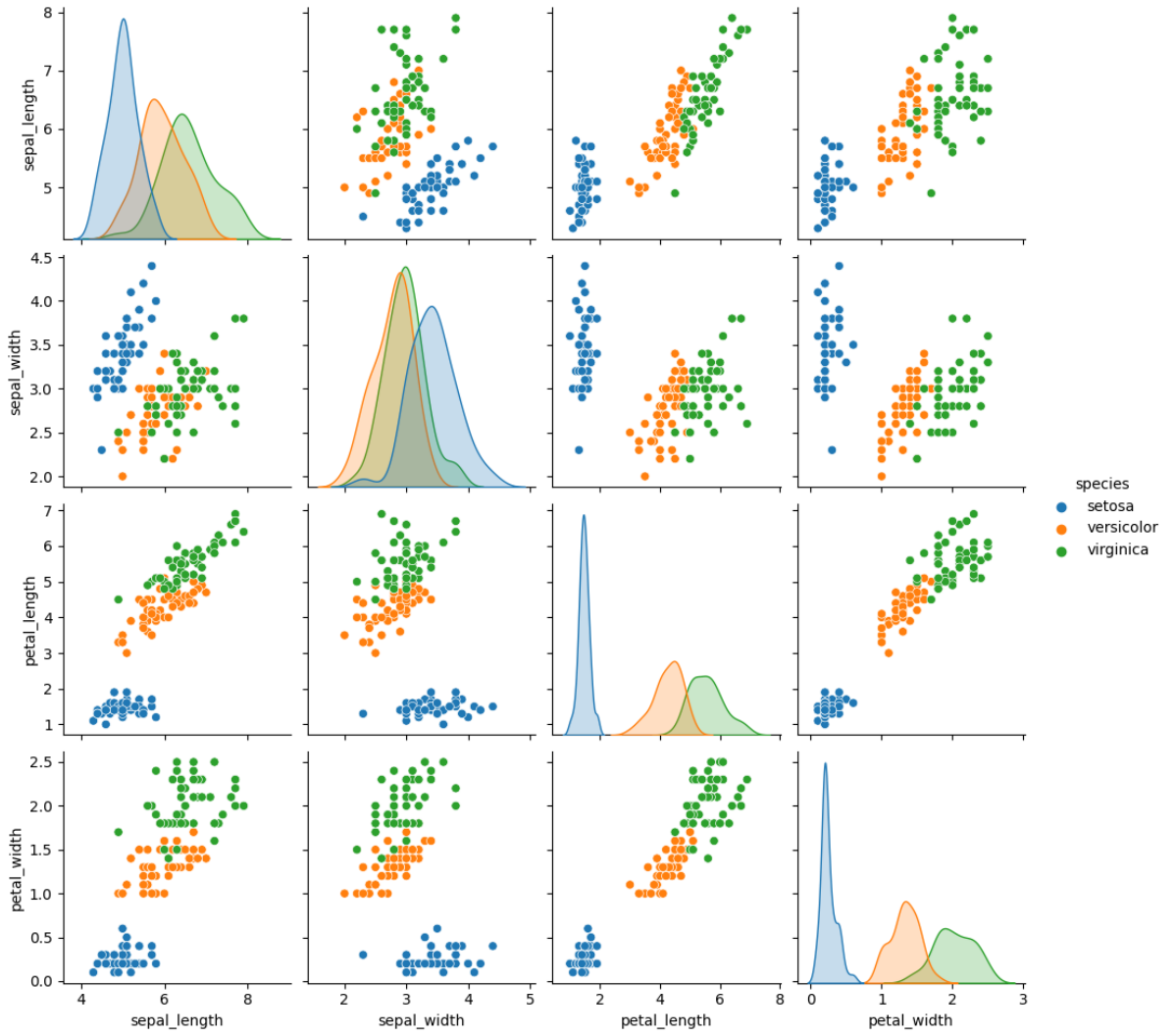
```
x = df1
print(target)

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,
```

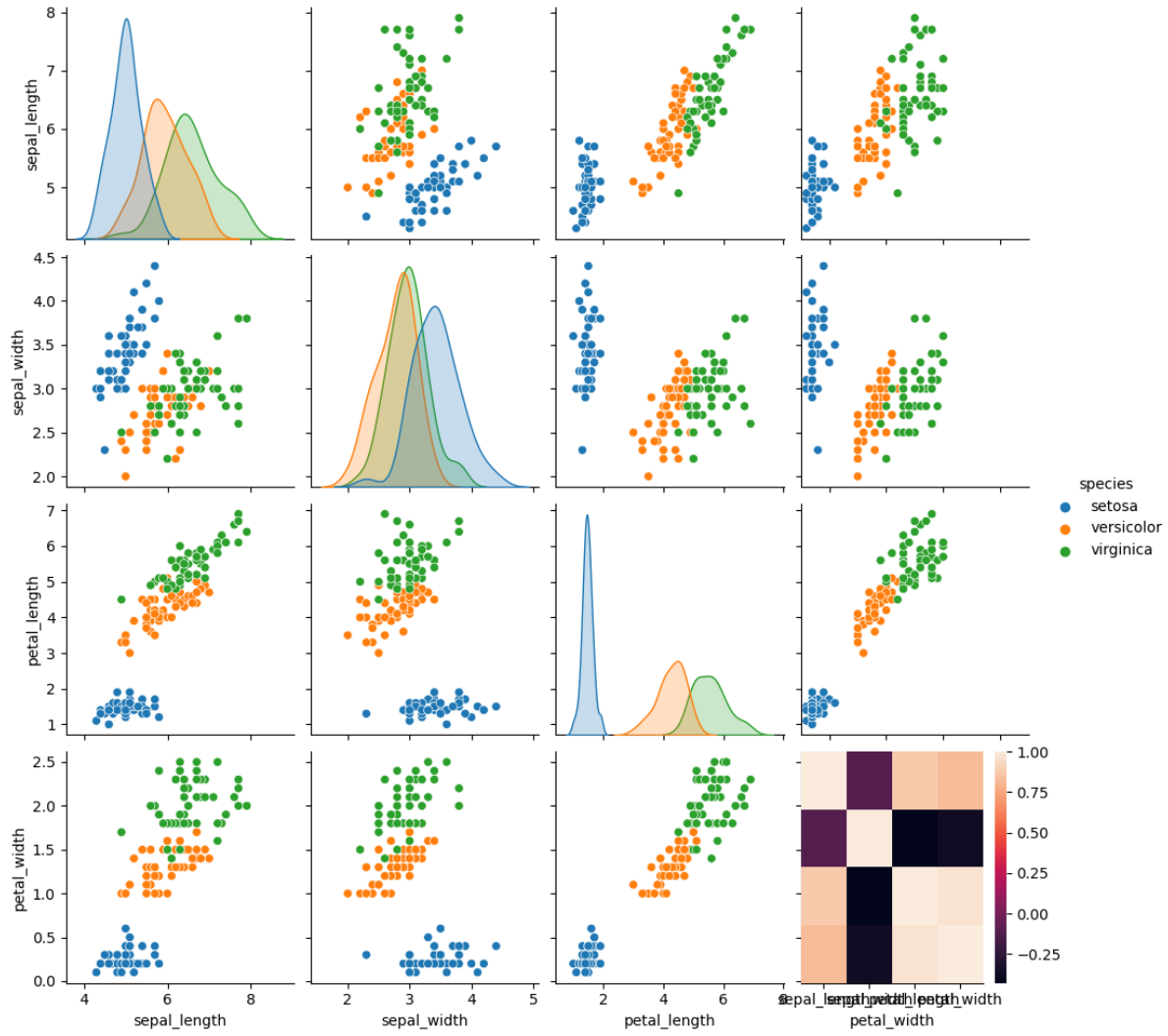
class_names=["setosa", "vercicolor", "verginica"] , filled=True, precision=4, rounded=True)

plt.savefig("three.png")

OUTPUT

```
C:\Users\hp\PycharmProjects\pythonProject\venv\Scripts\python.exe
   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)

Process finished with exit code 0
```


sepal_length
sepal_width
petal_length
petal_width
species
setosa
versicolor
virginica

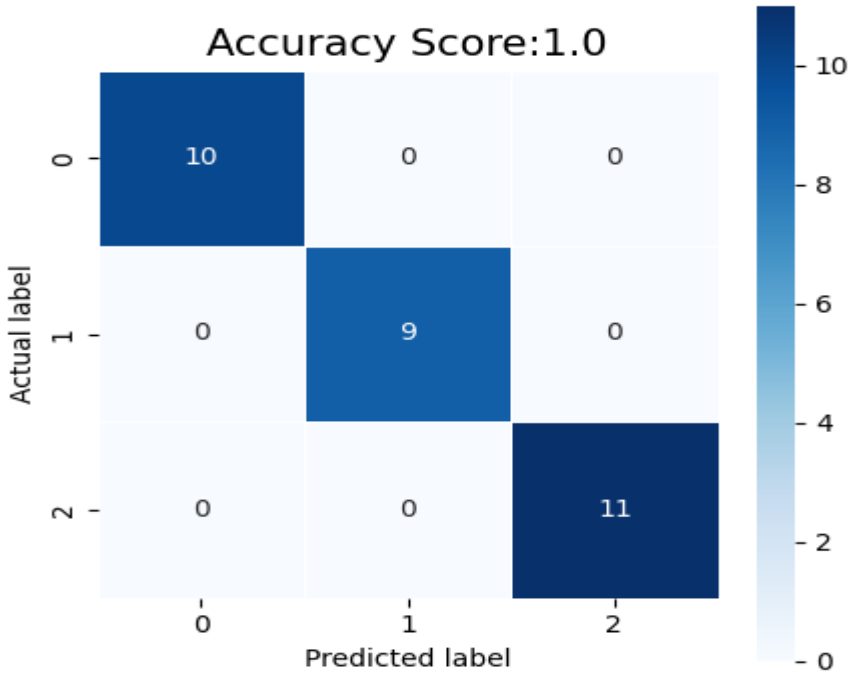


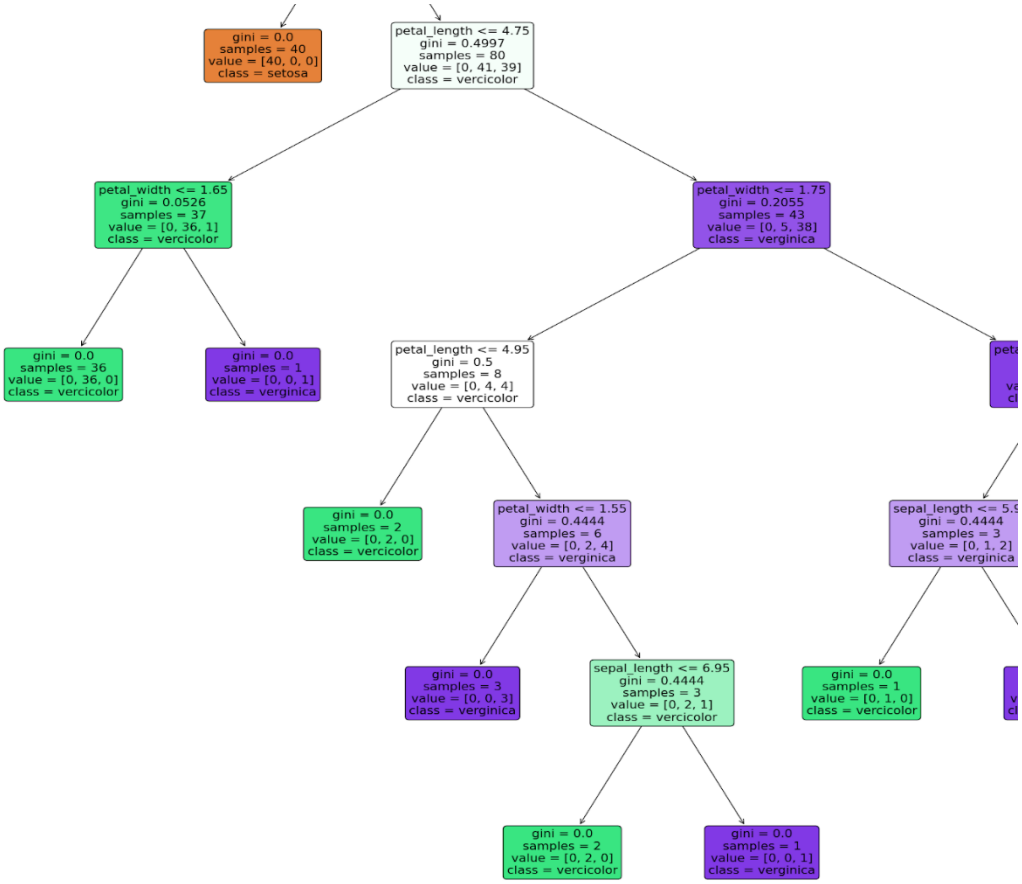

```
(150, 4)
   sepal_length  sepal_width  petal_length  petal_width
0           5.1          3.5           1.4          0.2
1           4.9          3.0           1.4          0.2
2           4.7          3.2           1.3          0.2
3           4.6          3.1           1.5          0.2
4           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 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]
```

```
Training split input-   (120, 4)
Testing split input-   (30, 4)
```





RESULT: THE PROGRAM HAS BEEN EXECUTED AND OUTPUT VERIFIED

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

AIM: PROGRAM TO IMPLEMENT K-MEANS CLUSTERING TECHNIQUE USING ANY STANDARD DATASET AVAILABLE IN THE PUBLIC DOMAIN.

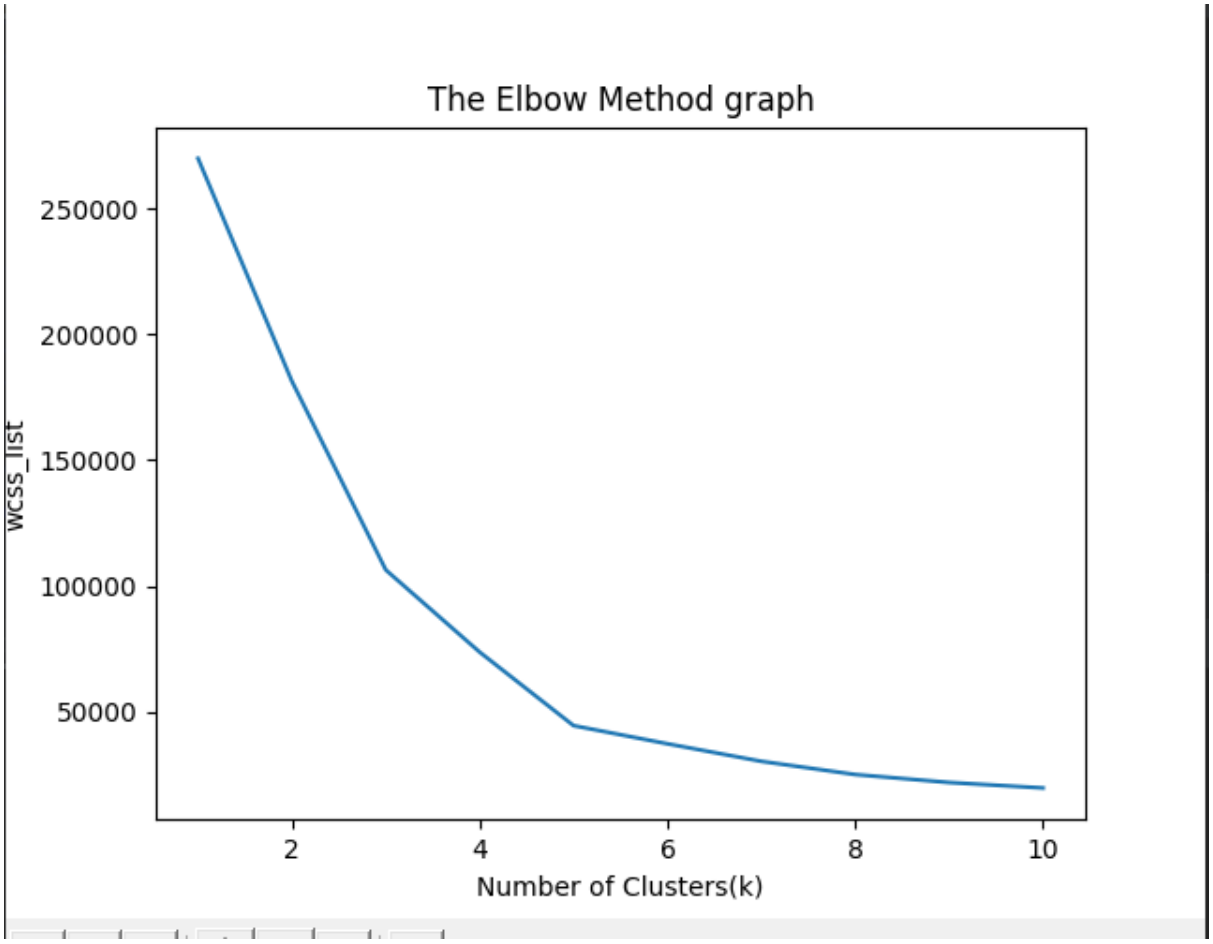
PROGRAM

```
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=60, c='red', label='Cluster1')
mtp.scatter(X[y_predict == 1, 0], X[y_predict == 1, 1], s=60, c='blue', label='Cluster2')
mtp.scatter(X[y_predict == 2, 0], X[y_predict == 2, 1], s=60, c='green', label='Cluster3')
mtp.scatter(X[y_predict == 3, 0], X[y_predict == 3, 1], s=60, c='violet', label='Cluster4')
```

```
mtp.scatter(X[y_predict == 4, 0], X[y_predict == 4, 1], s=60, c='yellow', label='Cluster5')

mtp.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s=100, c='black', label='Centroids')

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

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

mtp.legend()


mtp.show()
```

OUTPUT

```
C:\Users\ajcemca\PycharmProjects\pythonProject\venv\Scripts\python.exe "C:/
[[ 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]
 [ 93  14]
 [ 93  90]
 [ 97  32]
 [ 97  86]
 [ 98  15]
 [ 98  88]
 [ 99  39]
 [ 99  97]
 [101  24]
 [101  68]
 [103  17]
 [103  85]
 [103  23]
 [103  69]
 [113   8]
 [113  91]
 [120  16]
 [120  79]
 [126  28]
 [126  74]
 [137  18]
 [137  83]]
```




```
[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 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]

Process finished with exit code 0
```

RESULT: THE PROGRAM HAS BEEN EXECUTED AND OUTPUT VERIFIED

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

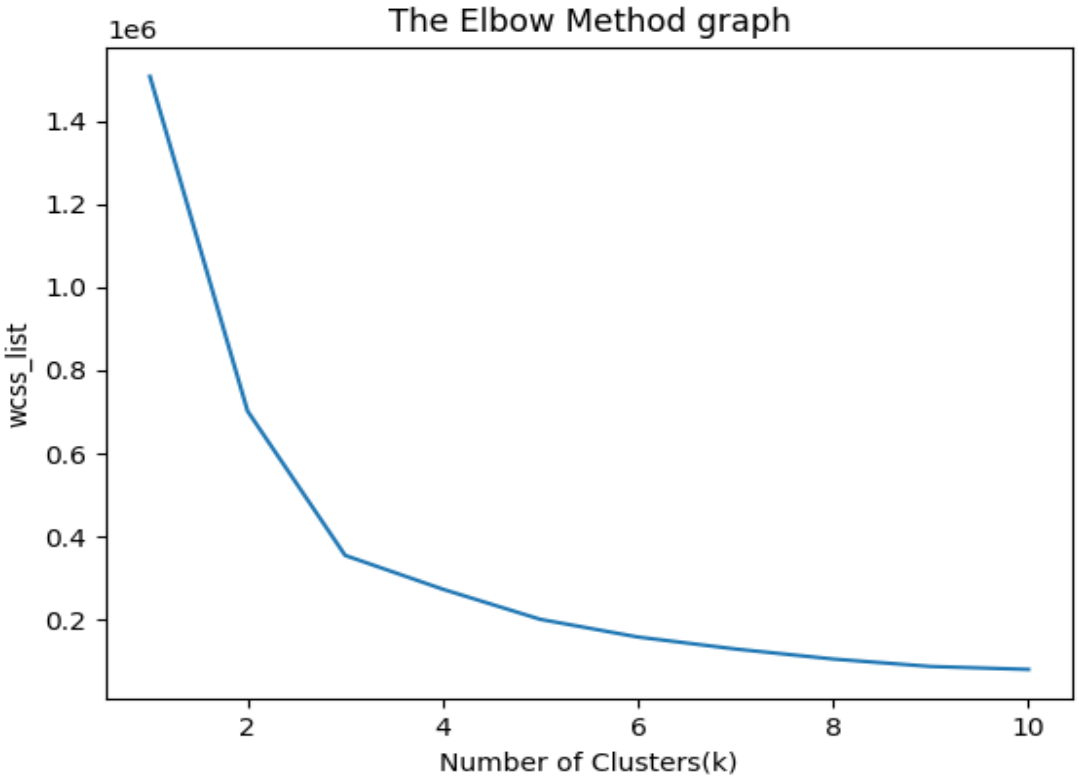
AIM: PROGRAM TO IMPLEMENT K-MEANS CLUSTERING TECHNIQUE USING ANY STANDARD DATASET AVAILABLE IN THE PUBLIC DOMAIN.

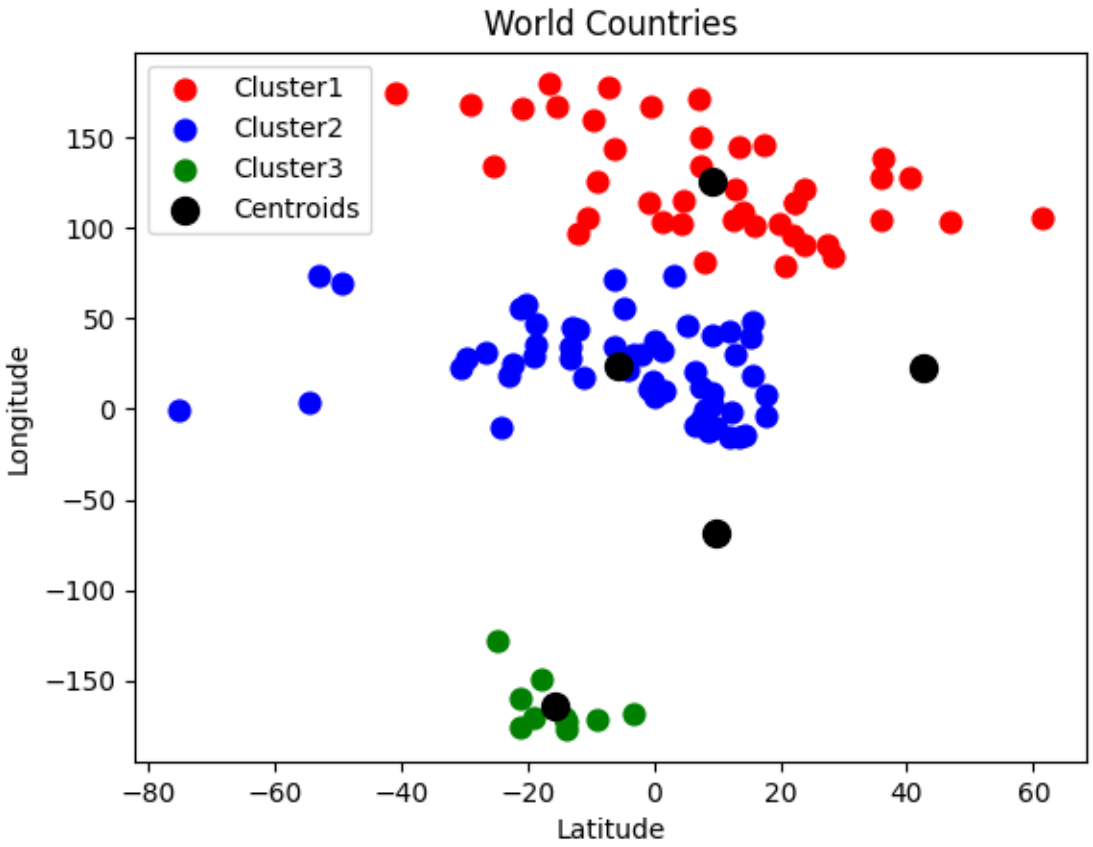
PROGRAM

```
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(y_predict)

mtp.scatter(X[y_predict == 0, 0], X[y_predict == 0, 1], s=60, c='red', label='Cluster1')
mtp.scatter(X[y_predict == 1, 0], X[y_predict == 1, 1], s=60, c='blue', label='Cluster2')
mtp.scatter(X[y_predict == 2, 0], X[y_predict == 2, 1], s=60, c='green', label='Cluster3')
```

```
mtp.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s=100, c='black', label='Centroids')

mtp.title('World Countries')

mtp.xlabel('Latitude')

mtp.ylabel('Longitude')

mtp.legend()


mtp.show()
```

OUTPUT






```
C:\Users\ajcemca\PycharmProjects\pythonProject\venv\Scr
[[ 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]
```

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

RESULT: THE PROGRAM HAS BEEN EXECUTED AND OUTPUT VERIFIED

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

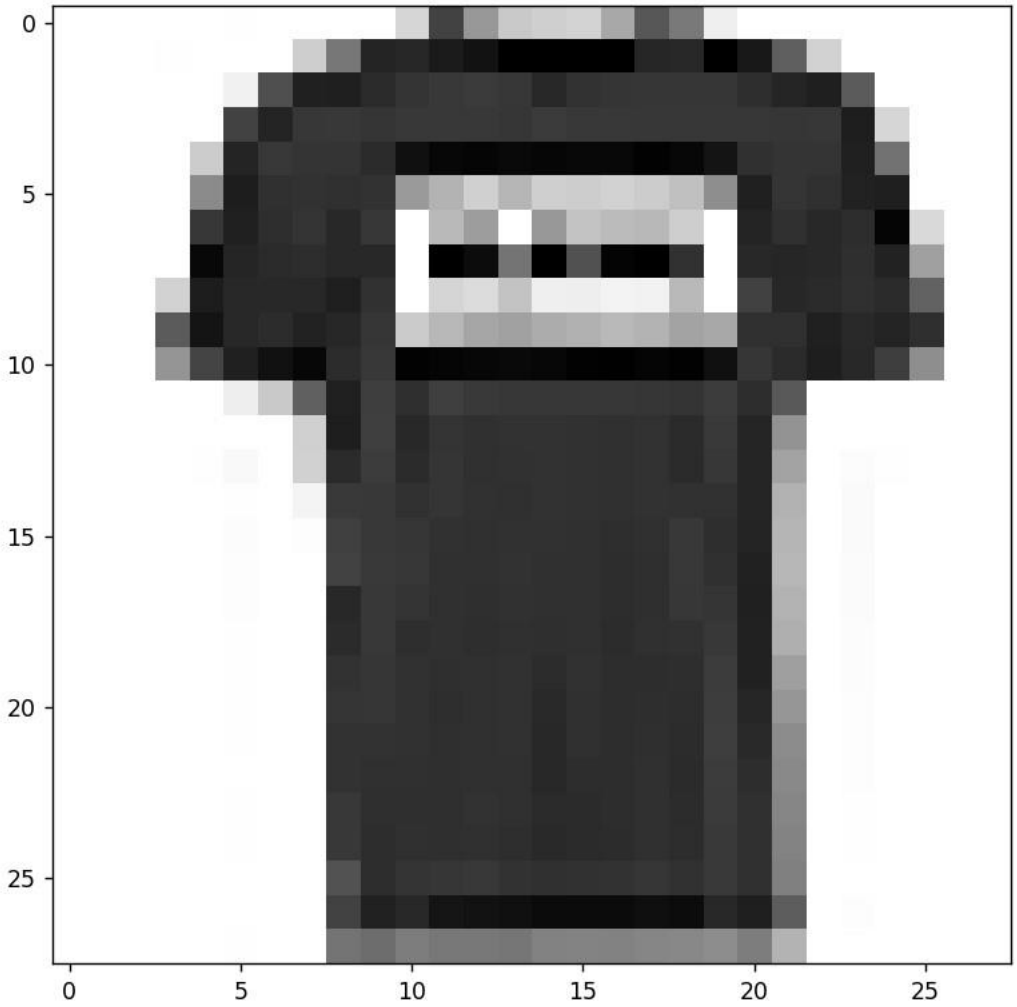
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow import keras

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)
```




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
8ag
Coat
Dress
T-Shirt/Top
Pullover
Coat
Coat
Sandal
Dress
Shirt
Shirt
T-Shirt/Top
8ag
Sandal
Pullover
Trouser
Shirt
Shirt
Sneaker
Ankle Boot
Sandal
Ankle Boot
Pullover
Sneaker
Dress
T-Shirt/Top
Dress


Training and validation plot
loss/accuracy
epochs
loss
accuracy
val_loss
val_accuracy
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.0
0.2
0.4
0.6
0.8
1.0
x=−0.016 y=0.900

```
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 #
=================================================================
 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)

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 [==============================] - 55s 32ms/step - loss: 0.5097 - accuracy: 0.8164 - val_loss: 0.3940 - val_accuracy: 0.8532
Epoch 2/2
1688/1688 [==============================] - 60s 36ms/step - loss: 0.3274 - accuracy: 0.8801 - val_loss: 0.3031 - val_accuracy: 0.8872
```

RESULT: THE PROGRAM HAS BEEN EXECUTED AND OUTPUT VERIFIED

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

AIM: IMPLEMENT A SIMPLE WEB CRAWLER

PROGRAM

```
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)
movies_soup = BeautifulSoup(movie_f.content, 'lxml')
movie_content = movies_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

```
[<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">
            The Irishman (2019)</a>, <a class="unstyled articleLink" href="/m/godfather">
            The Godfather (1972)</a>, <a class="unstyled articleLink" href="/m/mad_max_fury_road">
            Mad Max: Fury Road (2015)</a>, <a class="unstyled articleLink" href="/m/spider_man_into_the_spider_verse">
            Spider-Man: Into the Spider-Verse (2018)</a>, <a class="unstyled articleLink" href="/m/moonlight_2016">
            Moonlight (2016)</a>, <a class="unstyled articleLink" href="/m/sunset_boulevard">
            King Kong (1933)</a>, <a class="unstyled articleLink" href="/m/1018688-shadow_of_a_doubt">
            Shadow of a Doubt (1943)</a>, <a class="unstyled articleLink" href="/m/call_me_by_your_name">
            Call Me by Your Name (2018)</a>, <a class="unstyled articleLink" href="/m/psycho">
            Psycho (1960)</a>, <a class="unstyled articleLink" href="/m/1917_2019">
            1917 (2020)</a>, <a class="unstyled articleLink" href="/m/la_confidential">
            L.A. Confidential (1997)</a>, <a class="unstyled articleLink" href="/m/the_florida_project">
            The Florida Project (2017)</a>, <a class="unstyled articleLink" href="/m/war_for_the_planet_of_the_apes">
            War for the Planet of the Apes (2017)</a>, <a class="unstyled articleLink" href="/m/paddington_2">
            Paddington 2 (2018)</a>, <a class="unstyled articleLink" href="/m/beatles_a_hard_days_night">
            A Hard Day's Night (1964)</a>, <a class="unstyled articleLink" href="/m/widows_2018">
            Widows (2018)</a>, <a class="unstyled articleLink" href="/m/never_rarely_sometimes_always">
            Never Rarely Sometimes Always (2020)</a>, <a class="unstyled articleLink" href="/m/baby_driver">
            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_w
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

Process finished with exit code 0
```

RESULT: THE PROGRAM HAS BEEN EXECUTED AND OUTPUT VERIFIED

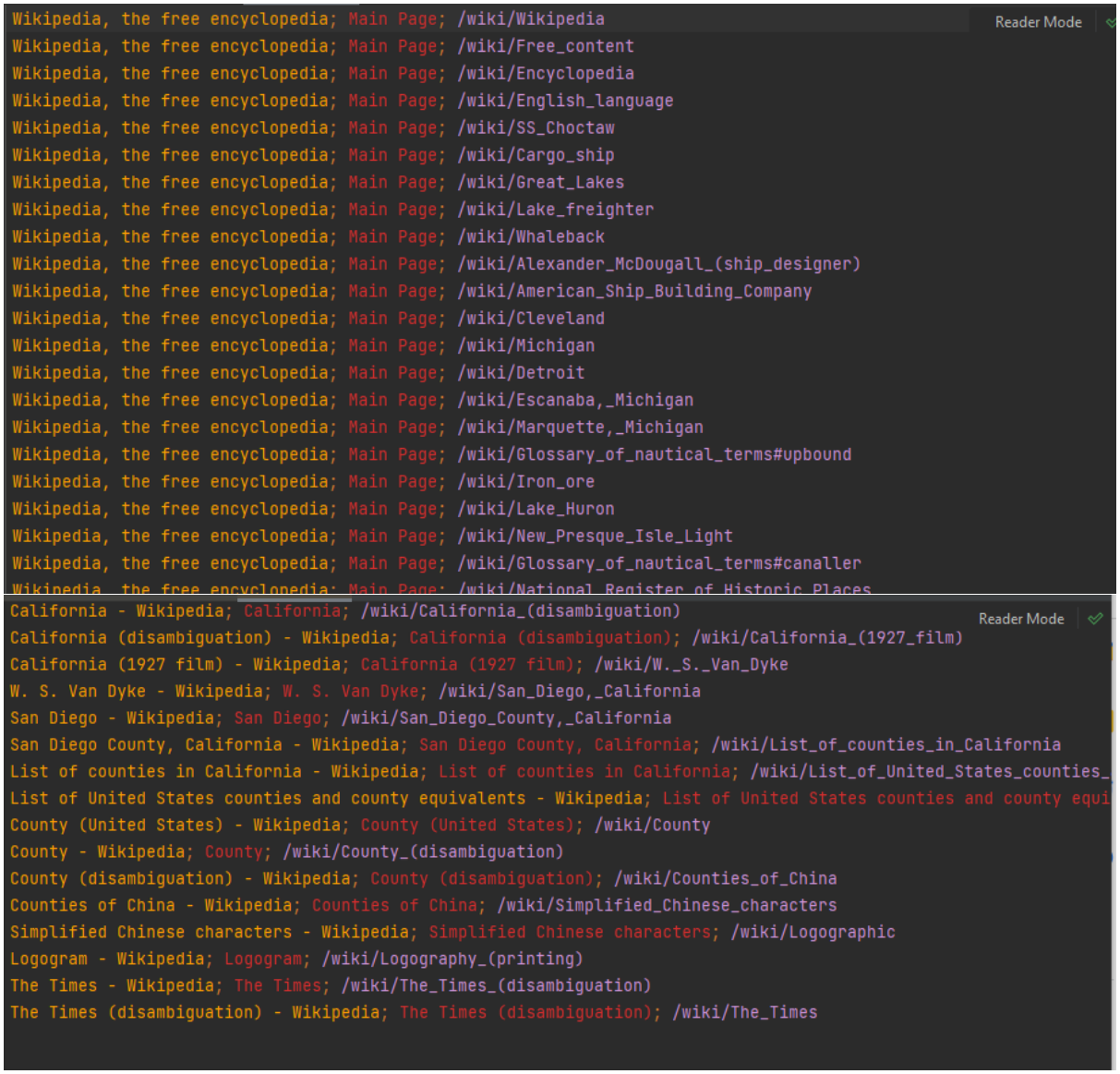
PROGRAM NO: 15 **DATE:16/02/2022**

AIM: IMPLEMENT A SIMPLE WEB CRAWLER

PROGRAM

```
from bs4 import BeautifulSoup
import requests

pages_crawler = []

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_crawler:
                    new_link = f"https://en.wikipedia.org{link['href']}"
                    pages_crawler.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



RESULT: THE PROGRAM HAS BEEN EXECUTED AND OUTPUT VERIFIED

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

AIM:IMPLEMENT A PROGRAM TO SCRAP THE WEB PAGE OF ANY POPULAR WEBSITE-SUGGESTED PYTHON PACKAGE IS SCRAPPY

PROGRAM

```
import requests
from bs4 import BeautifulSoup
import csv
import lxml

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'].aplit("#")[1]
    quotes.append(quote)

filename = 'inspirational_quote.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

```
C:\Users\ajcemca\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/ajcemca/PycharmProjects/pythonProject/venv/scrabing/scrabing.py
b'<!DOCTYPE html>\n<html class="no-js" dir="ltr" lang="en-US">\n    <head>\n        <title>Inspirational Quotes - Motivational Quotes - Leadership Quo
<!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-ggOyR0iXCbMQv3Xipma
  <link href="/assets/application-2a7a8e6a1c3f620bac9efa66420f5579.css" media="all" rel="stylesheet"/>
  <meta content="authenticity_token" name="csrf-param"/>
      </div>
     </div>
    </div>
   </div>
  </footer>
  <a class="scroll-top-arrow" href="javascript:void(0);">
   <i class="ti-arrow-up">
   </i>
  </a>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/1.12.4/jquery.js">
  </script>
  <script crossorigin="anonymous" integrity="sha384-UO2eT0CpHqdSJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86dIHNDz0W1" src="https://cdnjs.cloudflare.com/aja
  </script>
  <script crossorigin="anonymous" integrity="sha384-JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaFf/nJGzIxFDsf4x0xIM+B07jRM" src="https://stackpath.bootstrapcdn.c
  </script>
  <script src="/assets/pofo-1a7dc0d92519266568dcfcc8a6e53534.js">
  </script>
 </body>
</html>


Process finished with exit code 0
```

```
theme,url,img,lines,author
LOVE,/inspirational-quotes/7444-where-there-is-love-there-is-life,https://assets.passiton.com/quotes/quote_artwork
LOVE,/inspirational-quotes/7439-at-the-touch-of-love-everyone-becomes-a-poet,https://assets.passiton.com/quotes/qu
FRIENDSHIP,/inspirational-quotes/8304-a-friend-may-be-waiting-behind-a-stranger-s-face,https://assets.passiton.com
FRIENDSHIP,/inspirational-quotes/3331-wherever-we-are-it-is-our-friends-that-make,https://assets.passiton.com/quot
FRIENDSHIP,/inspirational-quotes/8303-find-a-group-of-people-who-challenge-and,https://assets.passiton.com/quotes/
FRIENDSHIP,/inspirational-quotes/8302-there-s-not-a-word-yet-for-old-friends-who-ve,https://assets.passiton.com/qu
FRIENDSHIP,/inspirational-quotes/7435-there-are-good-ships-and-wood-ships-ships-that,https://assets.passiton.com/q
PERSISTENCE,/inspirational-quotes/6377-at-211-degrees-water-is-hot-at-212-degrees,https://assets.passiton.com/quot
PERSISTENCE,/inspirational-quotes/8301-the-key-of-persistence-opens-all-doors-closed,https://assets.passiton.com/q
PERSISTENCE,/inspirational-quotes/7918-you-keep-putting-one-foot-in-front-of-the,https://assets.passiton.com/quote
PERSISTENCE,/inspirational-quotes/7919-to-persist-with-a-goal-you-must-treasure-the,https://assets.passiton.com/qu
PERSISTENCE,/inspirational-quotes/8300-failure-cannot-cope-with-persistence,https://assets.passiton.com/quotes/quo
INSPIRATION,/inspirational-quotes/8298-though-no-one-can-go-back-and-make-a-brand-new,https://assets.passiton.com/
INSPIRATION,/inspirational-quotes/8297-a-highly-developed-values-system-is-like-a,https://assets.passiton.com/quot
INSPIRATION,/inspirational-quotes/7066-just-don-t-give-up-trying-what-you-really-want,https://assets.passiton.com/
INSPIRATION,/inspirational-quotes/8296-when-we-strive-to-become-better-than-we-are,https://assets.passiton.com/quo
INSPIRATION,/inspirational-quotes/8299-the-most-important-thing-is-to-try-and-inspire,https://assets.passiton.com/
OVERCOMING,/inspirational-quotes/6828-bad-things-do-happen-how-i-respond-to-them,https://assets.passiton.com/quote
OVERCOMING,/inspirational-quotes/8294-show-me-someone-who-has-done-something,https://assets.passiton.com/quotes/qu
OVERCOMING,/inspirational-quotes/6137-its-not-the-load-that-breaks-you-down-its-the,https://assets.passiton.com/qu
```

RESULT: THE PROGRAM HAS BEEN EXECUTED AND OUTPUT VERIFIE

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

AIM:PYTHON PROGRAM FOR NATURAL PROGRAM LANGUAGE PROCESSING WHICH PERFORMS N-GRAMS

PROGRAM

```
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\pythonProject\venv\Scripts\python.exe C:/Users/ajcemca/PycharmProjects/pythonProject/venv/n_grams.
[['this', 'is', 'a'], ['is', 'a', 'very'], ['a', 'very', 'good'], ['very', 'good', 'book'], ['good', 'book', 'to'], ['book', 'to',

Process finished with exit code 0
```

RESULT: THE PROGRAM HAS BEEN EXECUTED AND OUTPUT VERIFIED

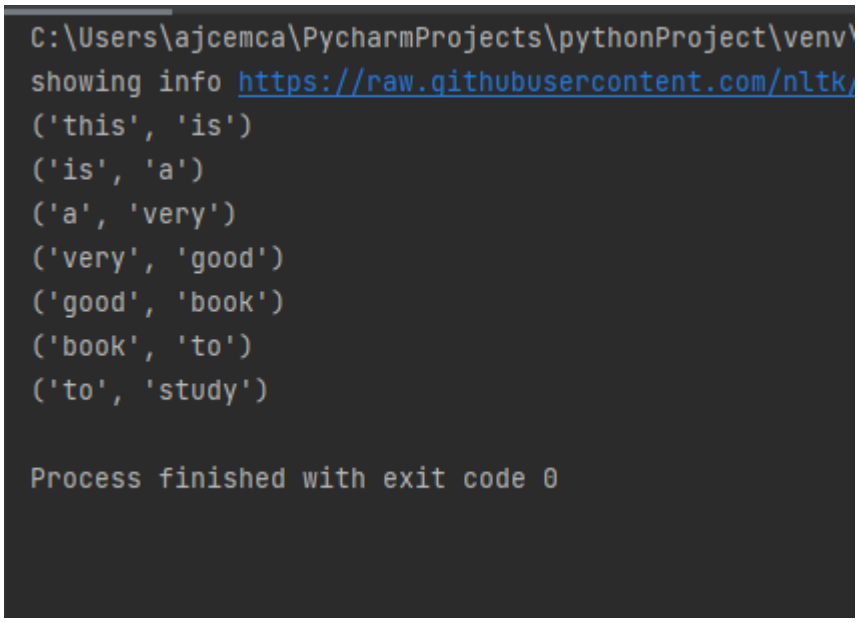
PROGRAM NO: 18 **DATE:16/02/2022**

AIM:PYTHON PROGRAM FOR NATURAL PROGRAM LANGUAGE PROCESSING WHICH PERFORMS N-GRAMS [USING IN BUILT FUNCTION]

PROGRAM

```
import nltk
nltk.download()
from nltk.util import ngrams

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

OUTPUT



RESULT: THE PROGRAM HAS BEEN EXECUTED AND OUTPUT VERIFIED

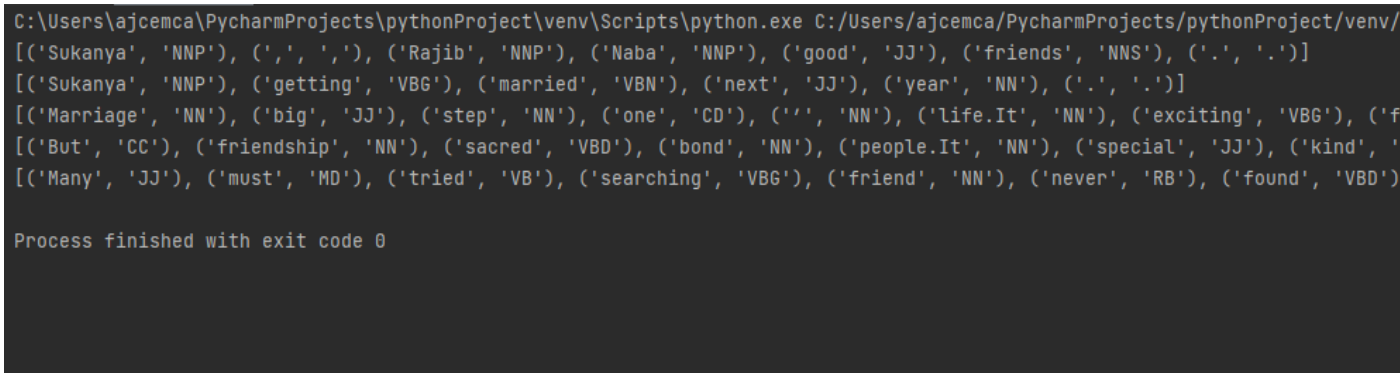
PROGRAM NO: 19 **DATE:16/02/2022**

AIM:PYTHON PROGRAM FOR NATURAL PROGRAM LANGUAGE PROCESSING WHICH PERFORMS SPEECH TAGGING

PROGRAM

```
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



RESULT: THE PROGRAM HAS BEEN EXECUTED AND OUTPUT VERIFIED

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

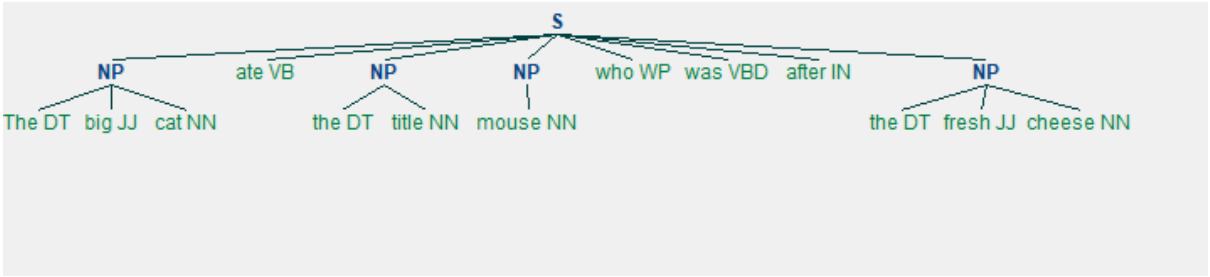
AIM:PYTHON PROGRAM FOR NATURAL PROGRAM LANGUAGE PROCESSING WHICH PERFORMS CHUNKING

PROGRAM

```
import nltk
new = "The big cat ate the title 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\pythonProject\venv\Scripts\python.exe C:/Users/ajcemca/PycharmProjects/pythonProject/venv/chun
['The', 'big', 'cat', 'ate', 'the', 'title', 'mouse', 'who', 'was', 'after', 'the', 'fresh', 'cheese']
[('The', 'DT'), ('big', 'JJ'), ('cat', 'NN'), ('ate', 'VB'), ('the', 'DT'), ('title', 'NN'), ('mouse', 'NN'), ('who', 'WP'), ('
(S
  (NP The/DT big/JJ cat/NN)
  ate/VB
  (NP the/DT title/NN)
  (NP mouse/NN)
  who/WP
  was/VBD
  after/IN
  (NP the/DT fresh/JJ cheese/NN))
```



RESULT: THE PROGRAM HAS BEEN EXECUTED AND OUTPUT VERIFIED

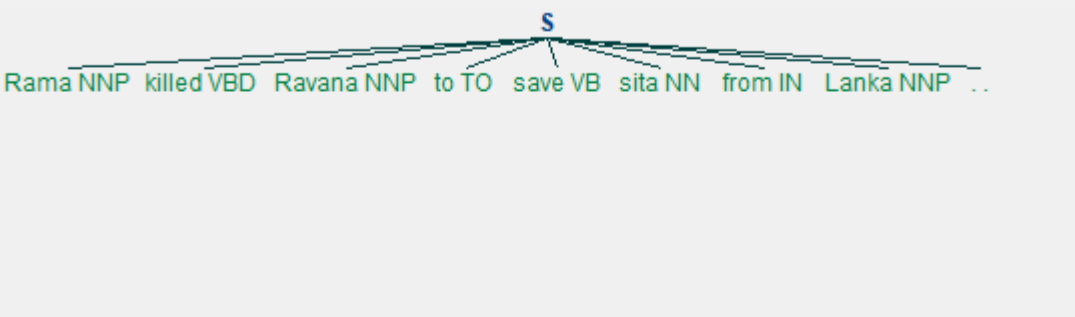
PROGRAM NO: 21 **DATE:23/02/2022**

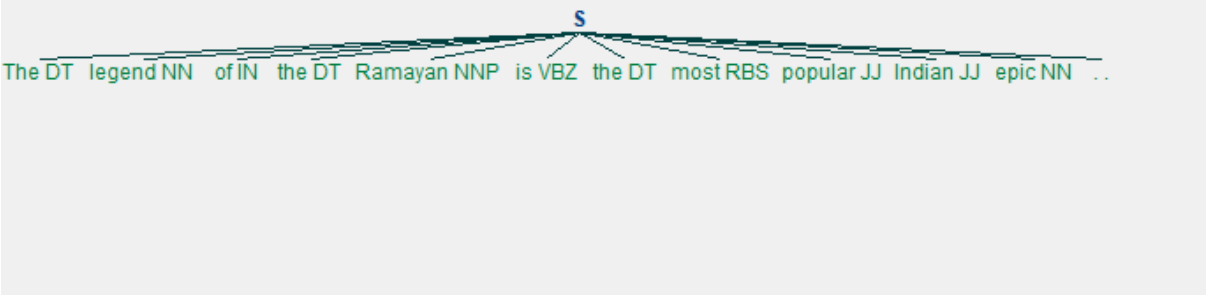
AIM:PYTHON PROGRAM FOR NATURAL PROGRAM LANGUAGE PROCESSING WHICH PERFORMS CHUNKING

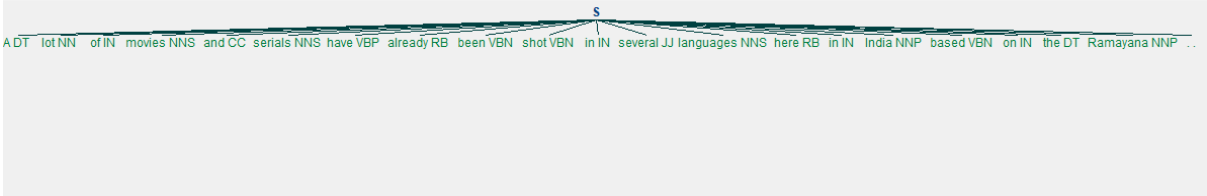
PROGRAM

```
import nltk

nltk.download('averaged_perceptron_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 Ramayana."""


tokenized = nltk.sent_tokenize(sample_text)

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

OUTPUT








```
(S
  Rama/NNP
  killed/VBD
  Ravana/NNP
  to/TO
  save/VB
  sita/NN
  from/IN
  Lanka/NNP
  ./.)
(S
  The/DT
  legend/NN
  of/IN
  the/DT
  Ramayan/NNP
  is/VBZ
  the/DT
  most/RBS
  popular/JJ
  Indian/JJ
  epic/NN
  ./.)
(S
  A/DT
  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/NNP
  based/VBN
  on/IN
  the/DT
  Ramayana/NNP
  ./.)
```

RESULT: THE PROGRAM HAS BEEN EXECUTED AND OUTPUT VERIFIED