

Program no: 01

Date: 24-11-2021

Aim: Perform all matrix operation using python.

Program:

```
import numpy as mato

print("Matrix Operations")
print("#####")

arr1 = mato.array([[10, 15], [5, 20]])
arr2 = mato.array([[7, 5], [3, 2]])

print("Operations with Numpy")
print("Added = ", mato.add(arr1, arr2))
print("Subtract = ", mato.subtract(arr1, arr2))
print("Multiplied = ", mato.multiply(arr1, arr2))
print("Divided = ", mato.divide(arr1, arr2))

print("Dot = ", mato.dot(arr1, arr2))
print("Sum = ", mato.sum(arr1))
print("Sum = ", mato.sum(arr1))
print("Sum of rows= ", mato.sum(arr2, axis=1))
print("Sum of cols= ", mato.sum(arr2, axis=0))

print("Transpose of array1", arr1.T)
print("Transpose of array2", arr2.T)
print("Sqrt of array1", mato.sqrt(arr1))
```

Output:

```
C:\Users\ajcemca\PycharmProjects\Anilect\venv\Scripts\python.exe C:/Users/ajcemca/PycharmProjects/Anilect/addmatrix.py
Matrix Operations
#####
Operations with Numpy
Added = [[17 20]
 [ 8 22]]
Subtract = [[ 3 10]
 [ 2 18]]
Multiplied = [[70 75]
 [15 40]]
Divided = [[ 1.42857143  3.          ]
 [ 1.66666667 10.          ]]
Dot = [[115  80]
 [ 95  65]]
Sum = 50
Sum = 50
Sum of rows= [12  5]
Sum of cols= [10  7]
Transpose of array1 [[10  5]
 [15 20]]
Transpose of array2 [[7 3]
 [5 2]]
Sqrt of array1 [[3.16227766 3.87298335]
 [2.23606798 4.47213595]]

Process finished with exit code 0
```

Result: The program has been executed and output verified

Program no: 02

Date: 01-12-2021

Aim: Program to perform SVD using python.

Program:

```
from numpy import array
from scipy.linalg import svd

Ar = array([[10, 20, 30, 40, 50], [15, 20, 25, 30, 35], [50, 40, 30, 20, 10]])
print(Ar)

i, j, k = svd(Ar)

print("\nDecomposition: ", i)
print("\nInverse Matrix: ", j)
print("\nTranspose of matrix", k)
```

Output:

```
C:\Users\ajcemca\PycharmProjects\Anilect\venv\Scripts\python.exe C:/Users/ajcemca/PycharmProjects/Anilect/svd.py
[[10 20 30 40 50]
 [15 20 25 30 35]
 [50 40 30 20 10]]

Decomposition: [[-0.63018567 -0.54861573 -0.54944226]
 [-0.51671457 -0.23186369  0.82416338]
 [-0.57954471  0.80328078 -0.13736056]]

Inverse Matrix: [1.10469408e+02 4.65994629e+01 4.91043299e-15]

Transpose of matrix [[-0.38951789 -0.41748928 -0.44546066 -0.47343205 -0.50140344]
 [ 0.66953403  0.35454577  0.03955751 -0.27543074 -0.590419  ]
 [-0.38223409  0.33080407  0.58267801 -0.62883185  0.09758387]
 [-0.49419597  0.42632677  0.08789984  0.52200386 -0.54203451]
 [-0.09832317  0.63938576 -0.6728744  -0.17911579  0.3109276  ]]
```

Process finished with exit code 0

Result: The program has been executed and output verified

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 built-in 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()
i = irisData.data
j = irisData.target

i_train, i_test, j_train, j_test = train_test_split(
    i, j, test_size=0.7, random_state=30
)
knn = KNeighborsClassifier(n_neighbors=1)
knn.fit(i_train, j_train)
print(knn.predict(i_test))

# finding Accuracy of algorithm
k = knn.predict(i_test)
l = accuracy_score(j_test, k)
print("Accuracy is", l)
```

Output:

```
C:\Users\ajcemca\PycharmProjects\Anilect\venv\Scripts\python.exe C:/Users/ajcemca/PycharmProjects/Anilect/knn.py
[0 0 0 2 1 1 2 2 1 2 0 2 1 1 0 1 0 0 0 2 2 0 0 0 2 2 2 2 0 1 2 1 2 2 2 1
 2 1 2 2 2 0 2 2 1 1 1 1 1 0 1 2 1 0 2 0 1 1 1 0 1 0 1 0 2 2 0 2 2 2 0 0
 1 0 2 2 2 1 0 1 0 1 1 1 2 0 1 0 1 2 1 0 0 0 2 2 0 1 1 1 0 0 0]
Accuracy is 0.9428571428571428

Process finished with exit code 0
```

Result: The program has been executed and output verified

Program no: 04**Date: 01-12-2021**

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

Program:

```
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

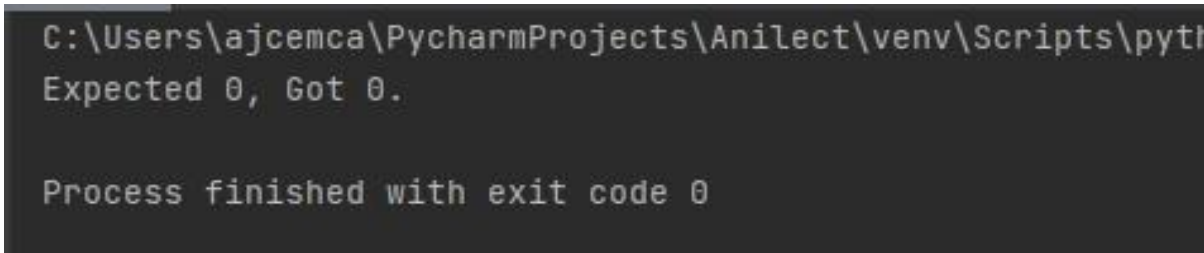
def predict_classification(train, test_row, num_neighbors):
    neighbors = get_neighbors(train, test_row, num_neighbors)
    output_values = [row[-1] for row in neighbors]
    # print(set(output_values))
    prediction = max(set(output_values), key=output_values.count)
    return prediction

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\Anilect\venv\Scripts\python
Expected 0, Got 0.

Process finished with exit code 0
```

Result: The program has been executed and output verified

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:

```
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=0.2, random_state=10)

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)

from sklearn.naive_bayes import GaussianNB

gnb = GaussianNB()
gnb.fit(x_train, y_train)
y_pred = gnb.predict(x_test)
print(y_pred)

from sklearn import metrics

print("Accuracy", metrics.accuracy_score(y_test, y_pred) * 100)

import numpy as nm
import matplotlib.pyplot as mtp
from matplotlib.colors import ListedColormap

x_set, y_set = x_train, y_train
```

```

X1, X2 = nm.meshgrid(nm.arange(start=x_set[:, 0].min() - 1, stop=x_set[:, 0].max() + 1,
                             step=0.01),
                    nm.arange(start=x_set[:, 1].min() - 1, stop=x_set[:, 1].max() + 1, step=0.01))
mtp.contourf(X1, X2, gnb.predict(nm.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
             alpha=0.75, cmap=ListedColormap(('purple', 'green')))
mtp.xlim(X1.min(), X1.max())
mtp.ylim(X2.min(), X2.max())
for i, j in enumerate(nm.unique(y_set)):
    mtp.scatter(x_set[y_set == j, 0], x_set[y_set == j, 1], c=ListedColormap(('purple',
                                                                              'green'))(i), label=j)

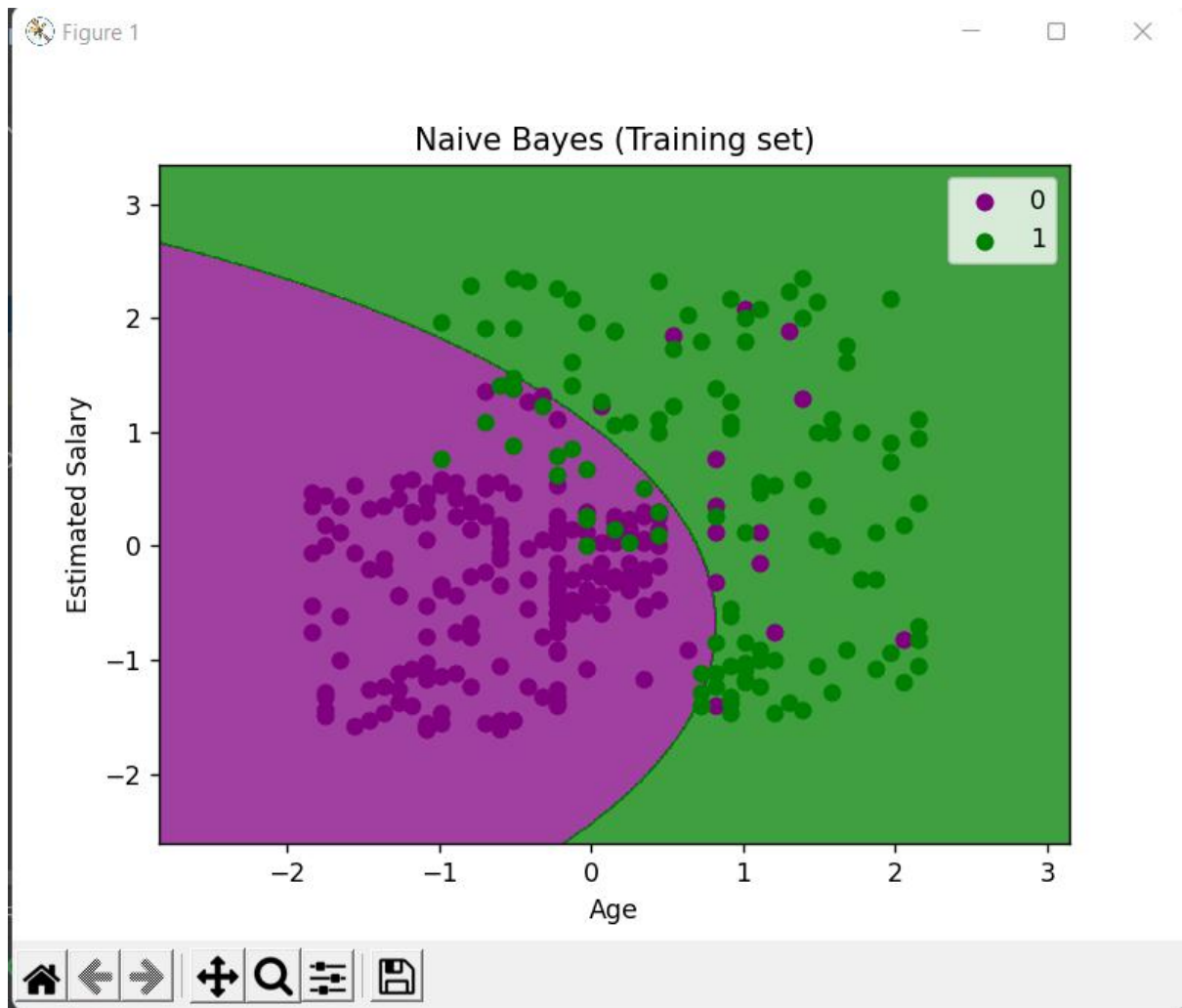
mtp.title('Naive Bayes (Training set)')
mtp.xlabel('Age')
mtp.ylabel('Estimated Salary')
mtp.legend()
mtp.show()

x_set, y_set = x_test, y_test
X1, X2 = nm.meshgrid(nm.arange(start=x_set[:, 0].min() - 1, stop=x_set[:, 0].max() + 1,
                             step=0.01),
                    nm.arange(start=x_set[:, 1].min() - 1, stop=x_set[:, 1].max() + 1, step=0.01))
mtp.contourf(X1, X2, gnb.predict(nm.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
             alpha=0.75, cmap=ListedColormap(('purple', 'green')))
mtp.xlim(X1.min(), X1.max())
mtp.ylim(X2.min(), X2.max())
for i, j in enumerate(nm.unique(y_set)):
    mtp.scatter(x_set[y_set == j, 0], x_set[y_set == j, 1], c=ListedColormap(('purple',
                                                                              'green'))(i), label=j)

mtp.title('Naive Bayes (test set)')
mtp.xlabel('Age')
mtp.ylabel('Estimated Salary')
mtp.legend()
mtp.show()

```


Output:



```
C:\Users\anjali\PycharmProjects\pythonProject1\venv\Scripts\python.exe C:/Users/anjali/PycharmProj
[0 0 1 1 0 1 0 1 0 0 0 0 1 1 1 0 0 0 0 1 0 0 0 1 1 0 0 1 1 0 0 0 0 1 1 0 1
 1 0 0 0 0 0 0 0 0 1 0 0 0 0 1 1 0 0 0 1 0 1 1 0 1 0 1 1 1 0 1 0 0 0 0 0 0
 0 0 0 0 1 1]
Accuracy 91.25
```

Result: The program has been executed and output verified

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: (Build-in Func)

```
import numpy as np
from sklearn.linear_model import LinearRegression
```

```
x = np.array([10,20,30,40,50,60]).reshape(-1,1)
y = np.array([5,10,15,20,25,30])
print("Linear Regression")
print("Array 1: ", x)
print("Array 2: ", 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_)
print("Predicted response: ", y_pred,sep="\n")
```

```
plt.plot(x,y_pred, color = "g")
plt.title('Linear Regression')
plt.xlabel('X')
plt.ylabel('Y')
plt.show()
```

Output:

```
C:\Users\ajcemca\PycharmProjects\Anilect\venv\Scripts\python.exe C:\Users\ajcemc
Linear Regression
Array 1: [[10]
[20]
[30]
[40]
[50]
[60]]
Array 2: [ 5 10 15 20 25 30]
Coefficient of determination: 1.0
Intercept: -3.552713678800501e-15
Slope: [0.5]

Process finished with exit code 0
```

Program no: 07**Date: 15-12-2021**

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

Program:

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

# A basic implementation of linear regression with one variable
# Part of Cosmos by OpenGenus Foundation
```

```
def estimate_coef(x, y):
    # number of observations/points
    n = np.size(x)

    # mean of x and y vector
    m_x, m_y = np.mean(x), np.mean(y)

    # calculating cross-deviation and deviation about x
    SS_xy = np.sum(y * x - n * m_y * m_x)
    SS_xx = np.sum(x * x - n * m_x * m_x)

    # calculating regression coefficients
    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])

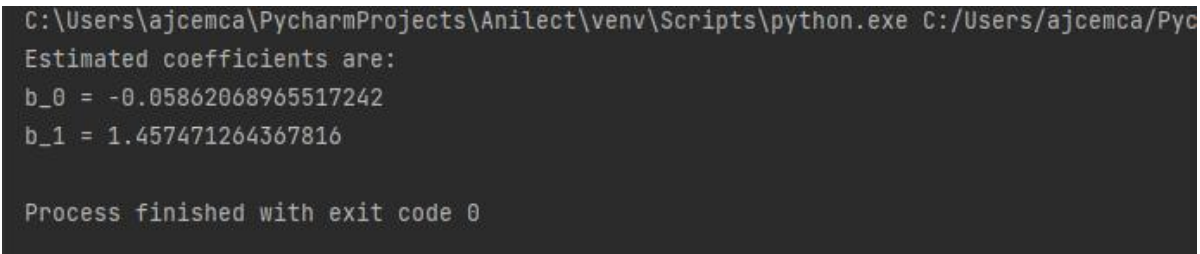
    # estimating coefficients
    b = estimate_coef(x, y)
    print("Estimated coefficients are:\nb_0 = {} \
        \nb_1 = {}".format(b[0], b[1]))

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

if __name__ == "__main__":
    main()

```

Output:

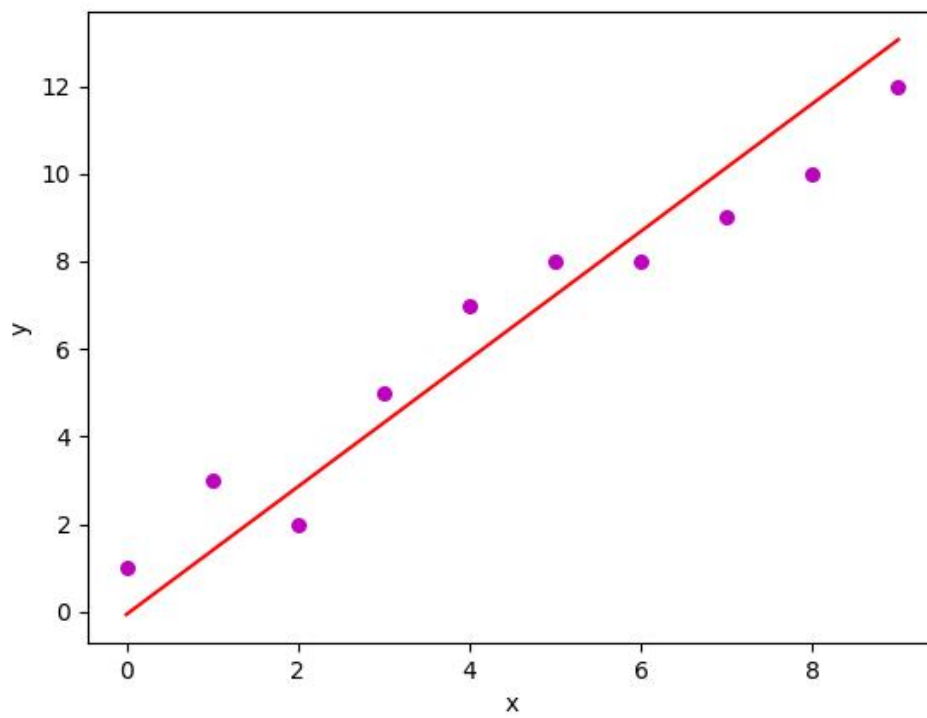


```

C:\Users\ajcemca\PycharmProjects\Anilect\venv\Scripts\python.exe C:/Users/ajcemca/Pyc
Estimated coefficients are:
b_0 = -0.05862068965517242
b_1 = 1.457471264367816

Process finished with exit code 0

```



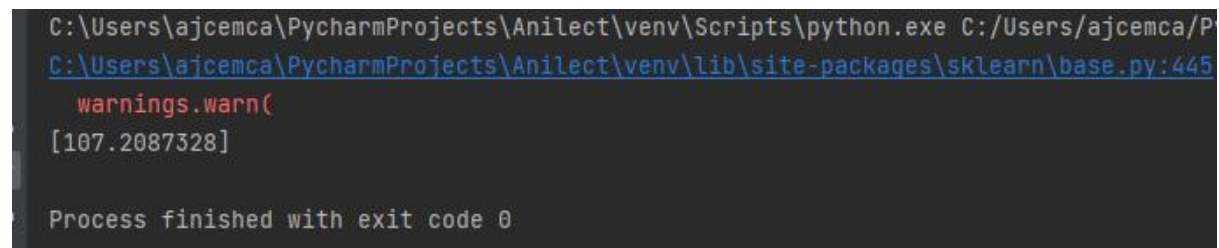
Result: The program has been executed and output verified

Program no: 08**Date: 15-12-2021**

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

Program:

```
import pandas
df = pandas.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\Anilect\venv\Scripts\python.exe C:/Users/ajcemca/P...
C:\Users\ajcemca\PycharmProjects\Anilect\venv\lib\site-packages\sklearn\base.py:445
warnings.warn(
[107.2087328]

Process finished with exit code 0
```

Result: The program has been executed and output verified

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:

```
import matplotlib.pyplot as plt
from sklearn import datasets, linear_model
from sklearn.metrics import mean_squared_error, 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=0.3, random_state=1)
reg = linear_model.LinearRegression()
reg.fit(X_train, y_train)
predicted = reg.predict(X_test)

# Regression coefficient
print('Coefficients are:\n', reg.coef_)

# Intercept
print('\nIntercept : ', reg.intercept_)

# variance score: 1 means perfect prediction
print('Variance score: ', reg.score(X_test, y_test))

# Mean Squared Error
print("Mean squared error: %.2f" % mean_squared_error(y_test, predicted))

# Original data of X_test
expected = y_test

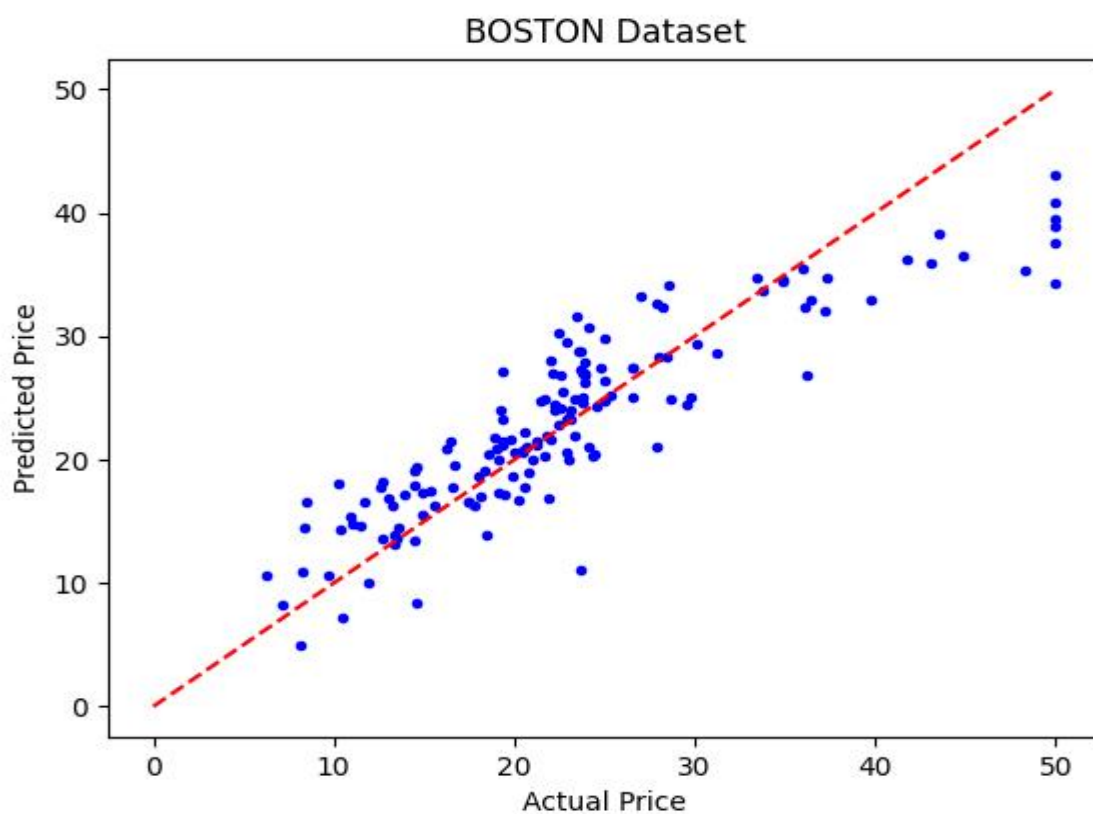
# Plot a graph for expected and predicted values
plt.title('ActualPrice Vs PredictedPrice (BOSTON Housing Dataset)')
plt.scatter(expected, predicted, c='b', marker='.', s=36)
```



```
plt.plot([0, 50], [0, 50], '--r')  
plt.xlabel('Actual Price(1000$)')  
plt.ylabel('Predicted Price(1000$)')  
plt.show()
```

Output:

```
Coefficients are:  
[-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 tree 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)

# Let's plot pair plot to visualise the attributes all at once
sns.pairplot(data=df, hue="species")
plt.savefig('pne.png')

# Correction matrix
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())
# Defining the attributes
x = df1
print(target)
# label encoding
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)

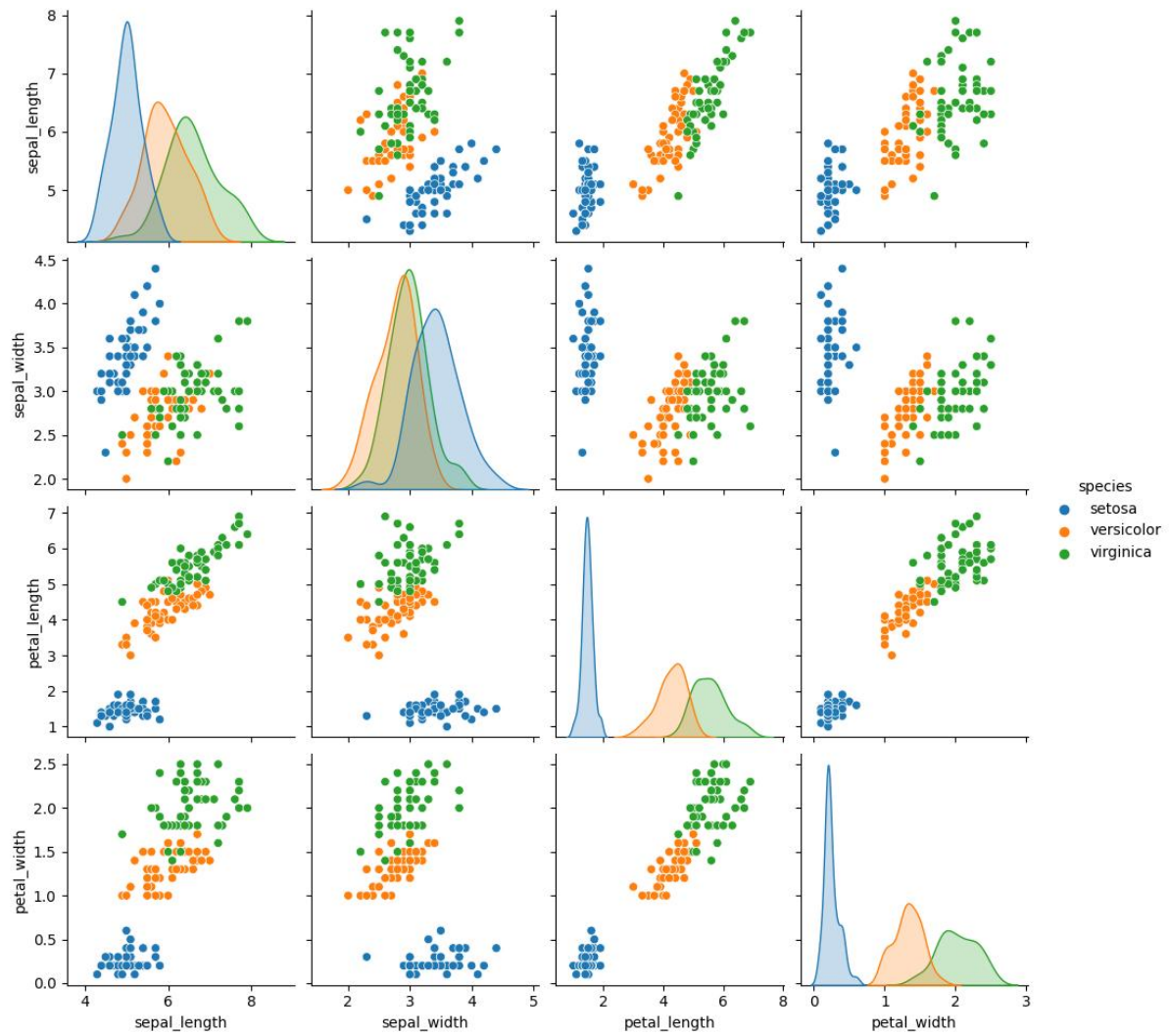
print("Training split input- ", X_train.shape)
print("Testing split input- ", X_test.shape)
# Defining the decision tree algorithm
dtree = DecisionTreeClassifier()
dtree.fit(X_train, y_train)

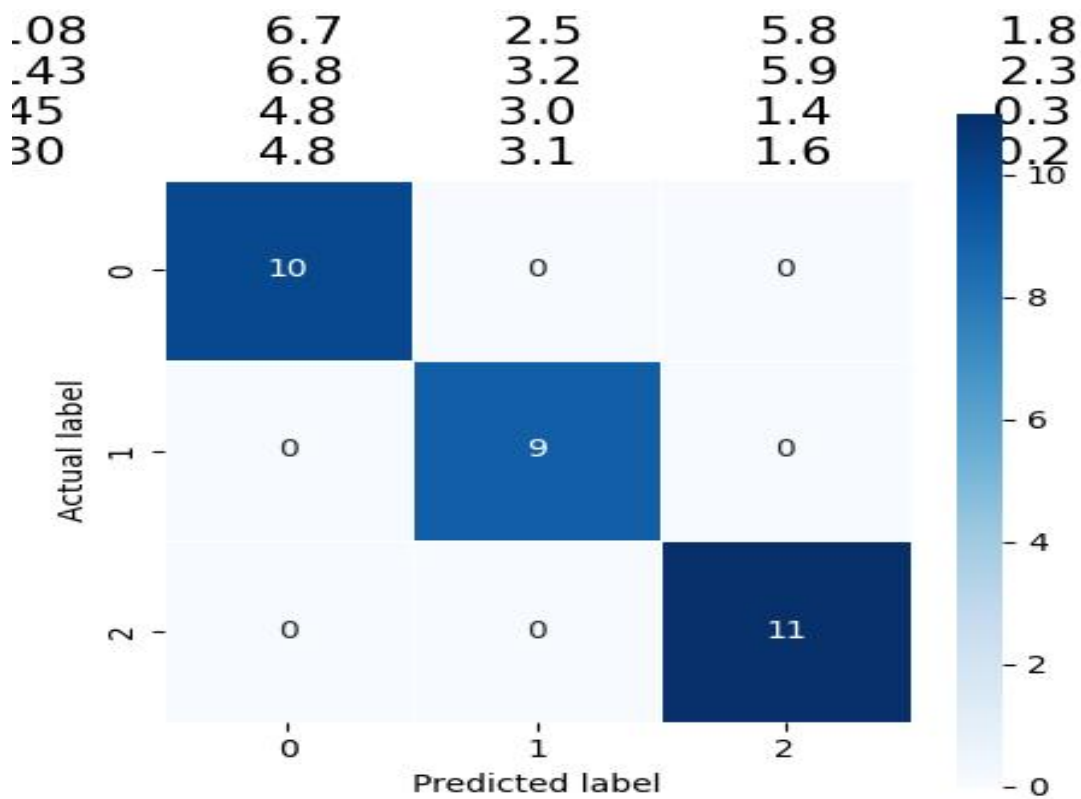
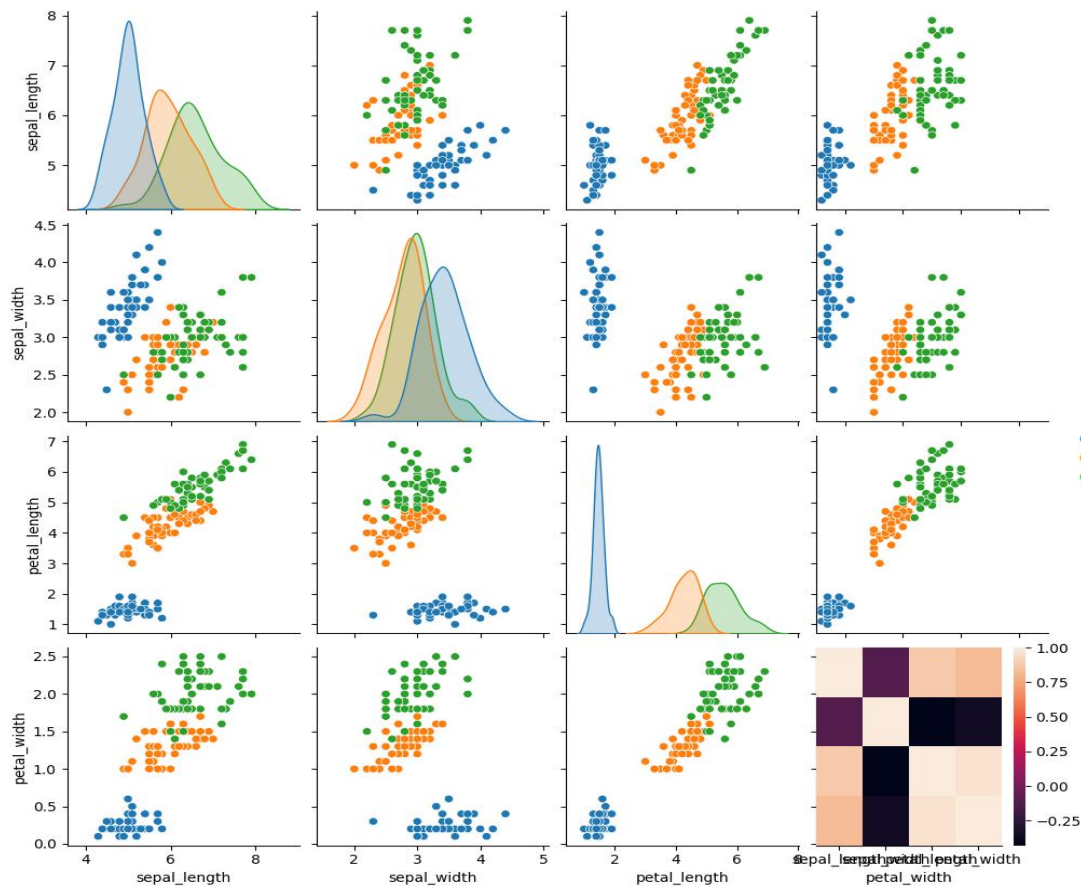
print('Decision Tree Classifier Created')
y_pred = dtree.predict(X_test)
print('Classification report - \n', classification_report(y_test, y_pred))
cm = confusion_matrix(y_test, y_pred)

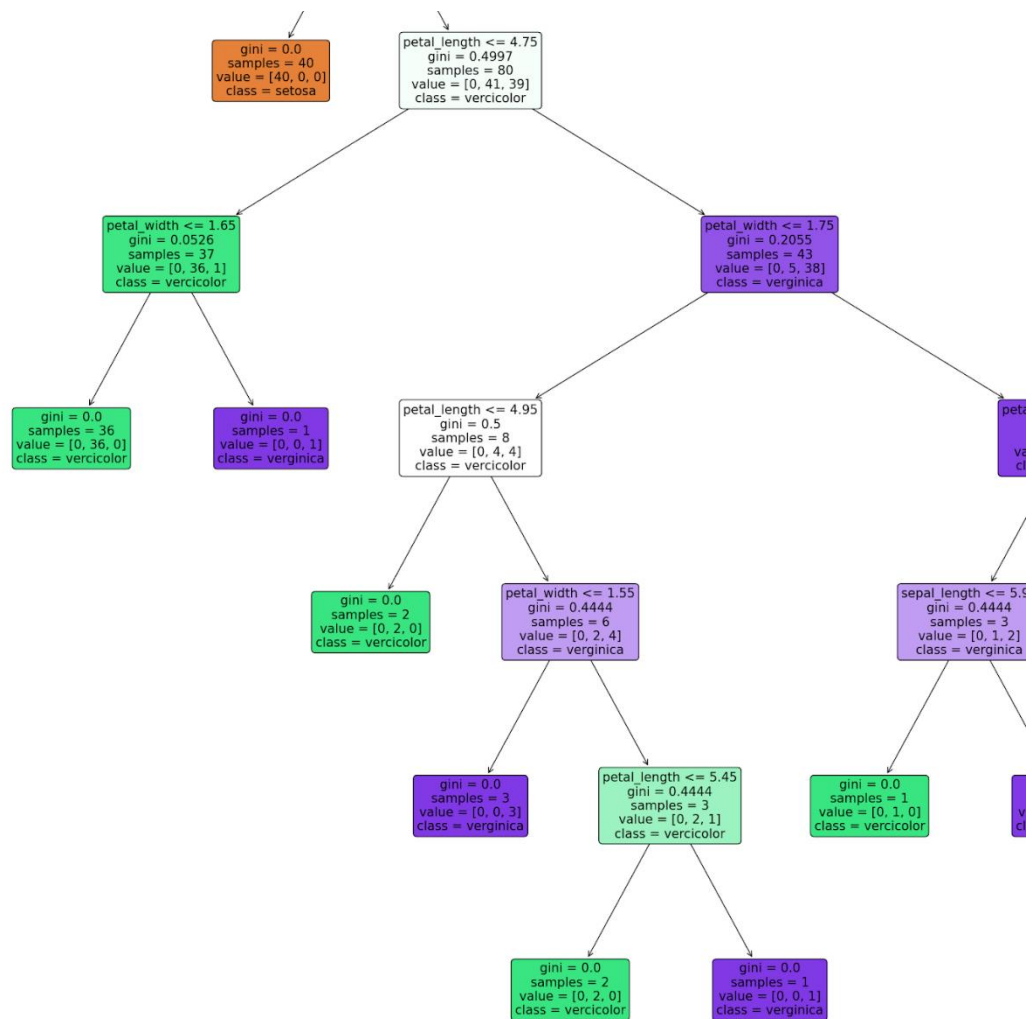
plt.figure(figsize=(5, 5))
sns.heatmap(data=cm, linewidth=.5, annot=True, square=True, cmap='Blues')
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
all_sample_title = 'Accuracy score: {0}'.format(X_test, y_test)
plt.title(all_sample_title, size=15)
plt.savefig('two.png')

plt.figure(figsize=(20, 20))
dec_tree = plot_tree(decision_tree=dtree, feature_names=df1.columns,

```





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 np
import matplotlib.pyplot as plt
import pandas as pd # Importing the dataset

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=0)
    kmeans.fit(X)
    wcss_list.append(kmeans.inertia_)

plt.plot(range(1, 11), wcss_list)
plt.title('The Elbow Method Graph')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()

kmeans = KMeans(n_clusters=5, init="k-means++", random_state=42)
y_predict = kmeans.fit_predict(X)
print(y_predict)

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

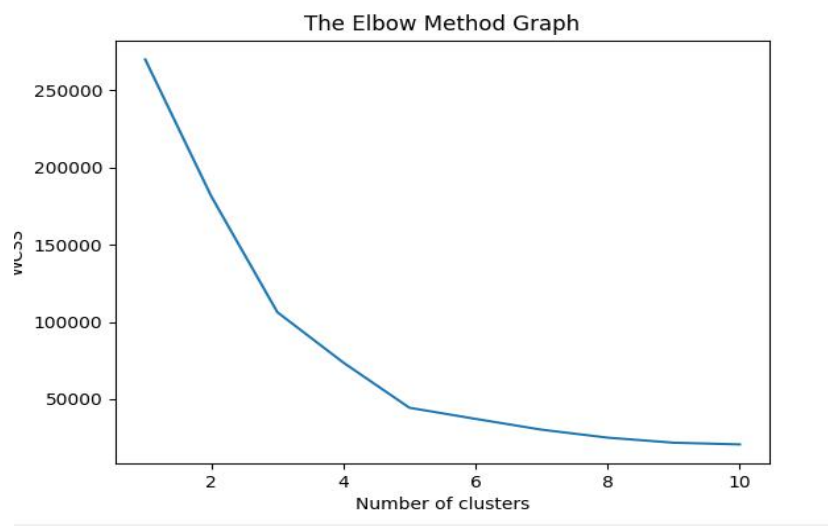


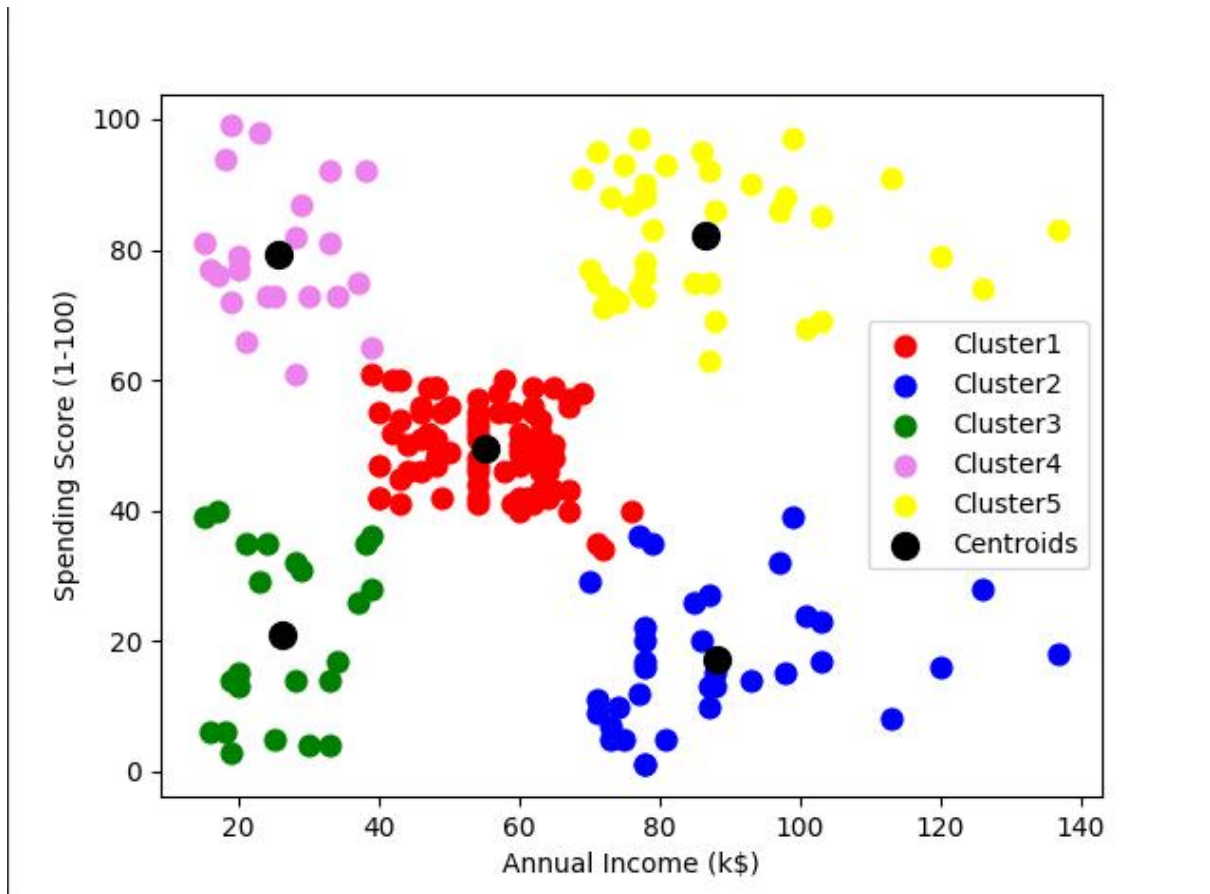
```
plt.scatter(X[y_predict == 3, 0], X[y_predict == 3, 1], s=60, c='violet', label='Cluster4')
plt.scatter(X[y_predict == 4, 0], X[y_predict == 4, 1], s=60, c='yellow', label='Cluster5')
plt.scatter(kmeans.cluster_centers_[0], kmeans.cluster_centers_[0], s=100, c='black',
label='Centroids')
```

```
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.legend()
plt.show()
```

Output:

```
C:\Users\ajcemca\PycharmProjects\Anilect\venv\Scripts\python.exe C:/Users/ajcemca/PycharmProjects/Anilect/k-mean/kmean.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]
 [ 21  35]
 [ 21  66]
 [ 23  29]
 [ 23  98]
 [ 24  35]
 [ 24  73]
```





20MCA241 Data Science Lab

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 np
import matplotlib.pyplot as plt
import pandas as pd

dataset = pd.read_csv('lati_log.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_)

plt.plot(range(1, 11), wcss_list)
plt.title('The Elbow Method Graph')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()

kmeans = KMeans(n_clusters=3, init="k-means++", random_state=42)
y_predict = kmeans.fit_predict(X)
print(y_predict)

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

```
plt.scatter(X[y_predict == 2, 0], X[y_predict == 2, 1], s=60, c='green', label='Cluster3')
plt.scatter(kmeans.cluster_centers_[0, 0], kmeans.cluster_centers_[0, 1], s=100, c='black',
label='Centroids')
```

```
plt.xlabel('latitude')
plt.ylabel('longitude')
plt.legend()

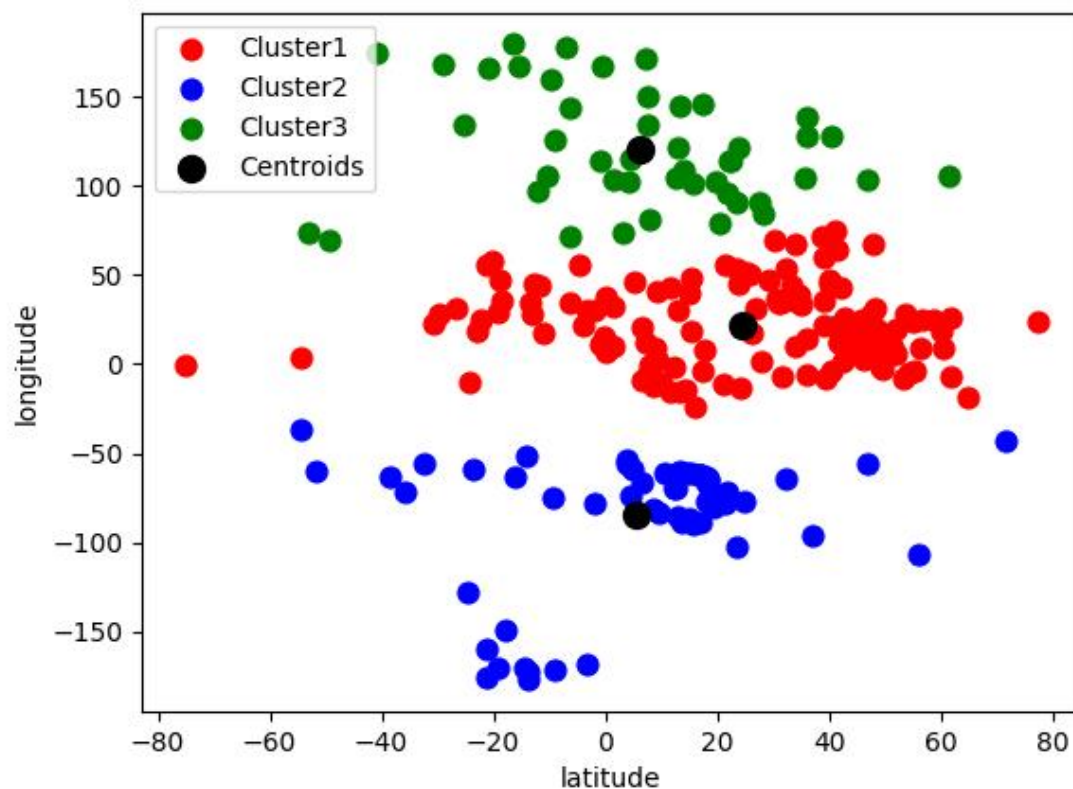
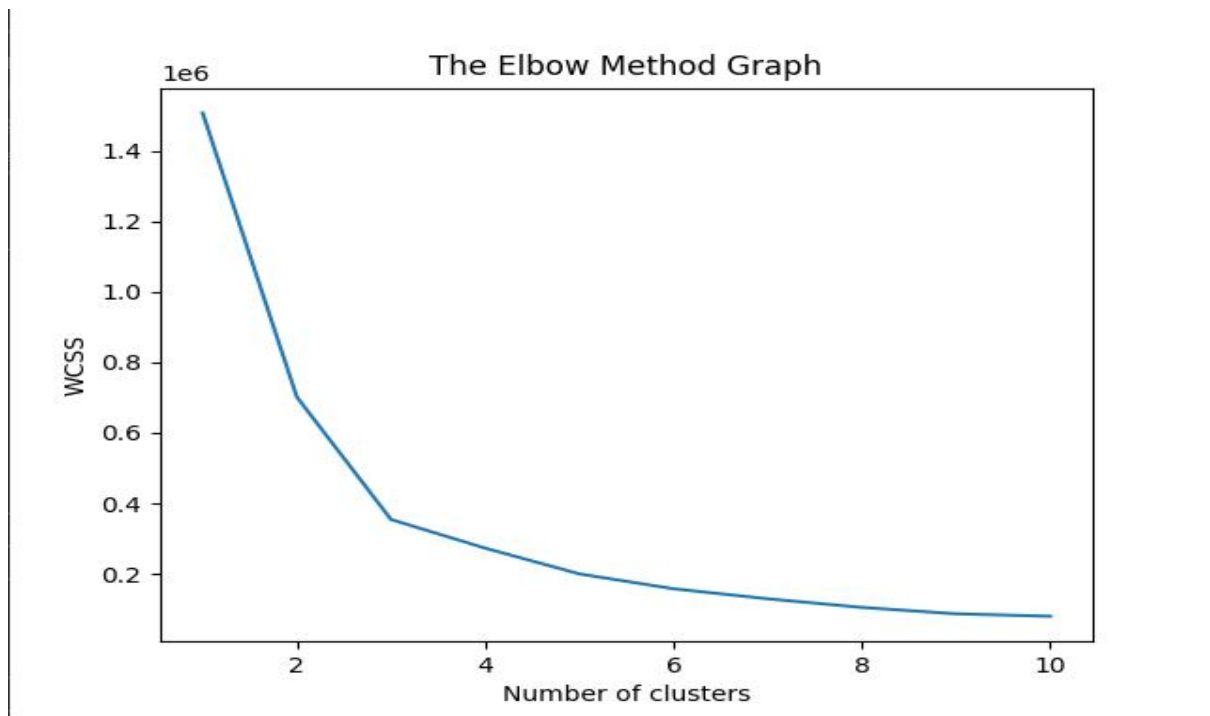
plt.show()
```

Output:

```
C:\Users\ajcemca\PycharmProjects\Anilect\venv\Scripts\python.exe C:/Users/ajcemca/PycharmProjects/Anilect/k-mean/kmean2.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]
```

```
[-1.90154380e+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 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 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]
```

Process finished with exit code 0



Result: The program has been executed and output verified

Program no: 13**Date: 02-02-2022**

Aim: Program 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')

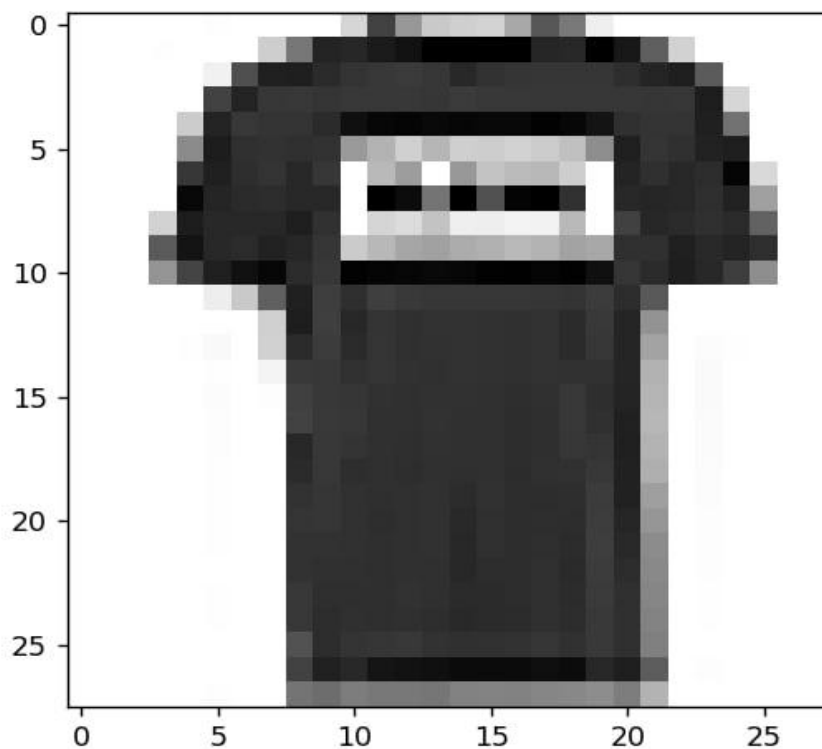
```

```
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:

```
Epoch 32 of 100: 60000/100000 (60000, 28, 28) (10000, 28, 28)
```





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 (MaxPooling2D) (None, 7, 7, 64)       0

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

max_pooling2d_2 (MaxPooling2D) (None, 3, 3, 32)       0

=====
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
```

```

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 (MaxPooling2D) (None, 7, 7, 64)       0
conv2d_2 (Conv2D)            (None, 7, 7, 32)         18464
max_pooling2d_2 (MaxPooling2D) (None, 3, 3, 32)       0
flatten (Flatten)            (None, 288)              0
dense (Dense)                (None, 128)              36992
dense_1 (Dense)              (None, 64)               8256
dense_2 (Dense)              (None, 10)               650
=====

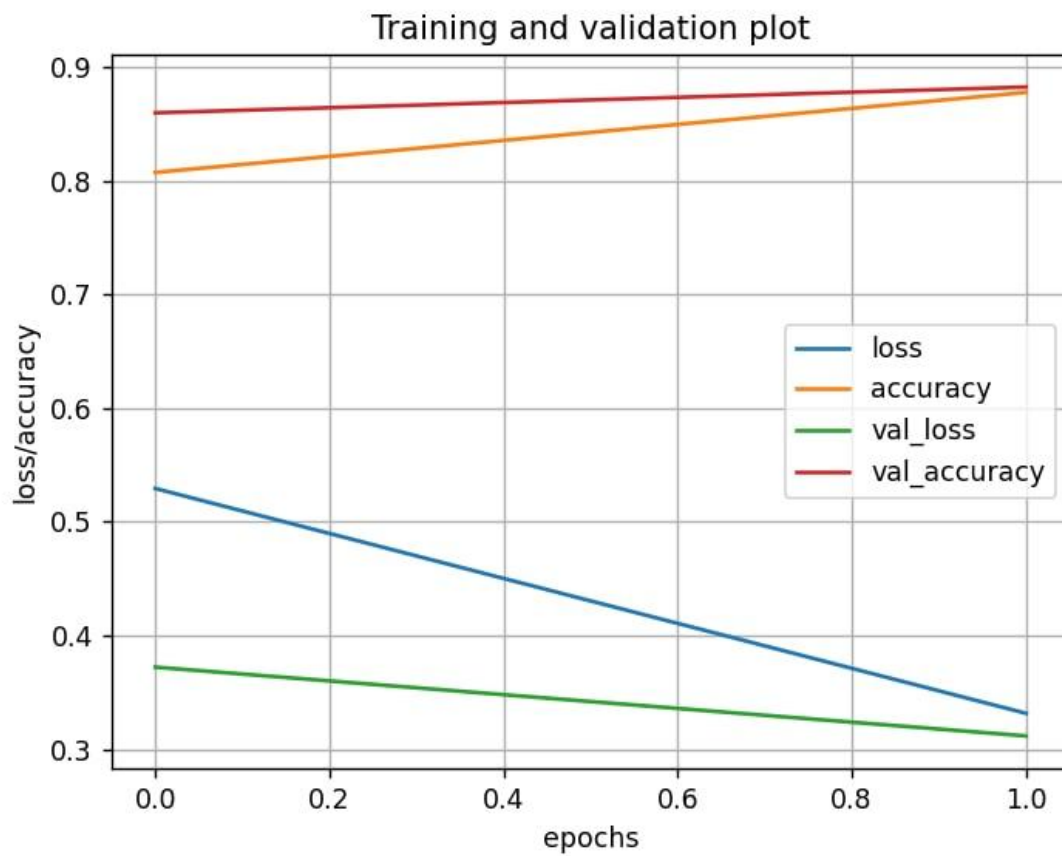
```

```

=====
Total params: 84,458
Trainable params: 84,458
Non-trainable params: 0
-----
Epoch 1/2
1688/1688 [=====] - 53s 31ms/step - loss: 0.5294 - accuracy: 0.8071 - val_loss: 0.3724 - val_accuracy: 0.8595
Epoch 2/2
1688/1688 [=====] - 52s 31ms/step - loss: 0.3318 - accuracy: 0.8775 - val_loss: 0.3120 - val_accuracy: 0.8822
313/313 [=====] - 3s 10ms/step - loss: 0.3342 - accuracy: 0.8752
Test Loss :0.3342011570930481, Test Accuracy : 0.8751999735832214

Process finished with exit code 0

```



Result: The program has been executed and output verified

Program no: 14**Date:** 16-02-2022**Aim:** Program to implement a simple web crawler using python**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
movies_Url = urls

movie_f = requests.get(movies_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:

```

Anillect - webcrawler.py
C:\Users\ajcemca\PycharmProjects\Anillect\venv\Scripts\python.exe C:/Users/ajcemca/PycharmProjects/Anillect/webcrawler.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/blackklansman">
    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">
    Sunset Boulevard (1950)</a>, <a class="unstyled articleLink" href="/m/1000626-all_about_eve">
    All About Eve (1950)</a>, <a class="unstyled articleLink" href="/m/the_cabinet_of_dr_caligari">
    The Cabinet of Dr. Caligari (Das Cabinet des Dr. Caligari) (1920)</a>, <a class="unstyled articleLink" href="/m/philadelphia_story">
    The Philadelphia Story (1940)</a>, <a class="unstyled articleLink" href="/m/roma_2018">
    Roma (2018)</a>]

Zootopia (2016)</a>, <a class="unstyled articleLink" href="/m/alien">
    Alien (1979)</a>, <a class="unstyled articleLink" href="/m/1011615-king_kong">
    King Kong (1933)</a>, <a class="unstyled articleLink" href="/m/1010688-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_wizard_of_oz',
 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

```

Result: The program has been executed and output verified

Program no: 15**Date:** 16-02-2022**Aim:** Program to implement a simple web crawler using python**Program:**

```
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 screenshot shows an IDE window titled 'Anillect - data.csv'. The main editor displays a CSV file with 25 rows of data. Each row contains a Wikipedia article title, followed by the word 'Wikipedia;', and then a path to the article's main page. A tooltip is visible over the 16th row, which reads: 'Header: /wiki/Wikipedia Index: 3'. The tooltip also shows the text '/wiki/Science_communication'.

Index	Wikipedia Article	Path
1	Wikipedia, the free encyclopedia; Main Page;	/wiki/Main_Page
2	Wikipedia, the free encyclopedia; Main Page;	/wiki/Main_Page
3	Free content - Wikipedia; Free content;	/wiki/Definition_of_Free_Cultural_Works
4	Definition of Free Cultural Works - Wikipedia; Definition of Free Cultural Works;	/wiki/Free_content_movement
5	Free-culture movement - Wikipedia; Free-culture movement;	/wiki/Free_culture_(disambiguation)
6	Free Culture - Wikipedia; Free Culture;	/wiki/Free_Culture_(book)
7	Free Culture (book) - Wikipedia; Free Culture (book);	/wiki/Lawrence_Lessig
8	Lawrence Lessig - Wikipedia; Lawrence Lessig;	/wiki/Lawrence_Lessig
9	Lawrence Lessig - Wikipedia; Lawrence Lessig;	/wiki/Science_writer
10	Science journalism - Wikipedia; Science journalism;	/wiki/Scientific_journalism
11	Scientific journalism - Wikipedia; Scientific journalism;	/wiki/Science_journalism
12	Science journalism - Wikipedia; Science journalism;	/wiki/Science_writing
13	Scientific writing - Wikipedia; Scientific writing;	/wiki/Science_writing
14	Science journalism - Wikipedia; Science journalism;	/wiki/Science_communication
15	Science communication - Wikipedia; Science communication;	/wiki/Science_publish
16	Scientific literature - Wikipedia; Scientific literature;	/wiki/Medical_literat
17	Medical literature - Wikipedia; Medical literature;	/wiki/Edwin_Smith_Papyrus
18	Edwin Smith Papyrus - Wikipedia; Edwin Smith Papyrus;	/wiki/New_York_Academy_of
19	New York Academy of Medicine - Wikipedia; New York Academy of Medicine;	/wiki/Eclecticism_in_architecture
20	Eclecticism in architecture - Wikipedia; Eclecticism in architecture;	/wiki/Basilica
21	Basilica - Wikipedia; Basilica;	/wiki/Basilicas_in_the_Catholic_Church
22	Basilicas in the Catholic Church - Wikipedia; Basilicas in the Catholic Church;	/wiki/List_of_Catholic_basilicas
23	List of Catholic basilicas - Wikipedia; List of Catholic basilicas;	/wiki/Catholic_Church
24	Catholic Church - Wikipedia; Catholic Church;	/wiki/Catholic_Church_(disambiguation)
25	Catholic Church (disambiguation) - Wikipedia; Catholic Church (disambiguation);	/wiki/Catholic_(disambiguation)

Result: The program has been executed and output verified

Program no: 16**Date: 16-02-2022****Aim:** Program to implement scrap of any webpage**Program:**

```
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:
```



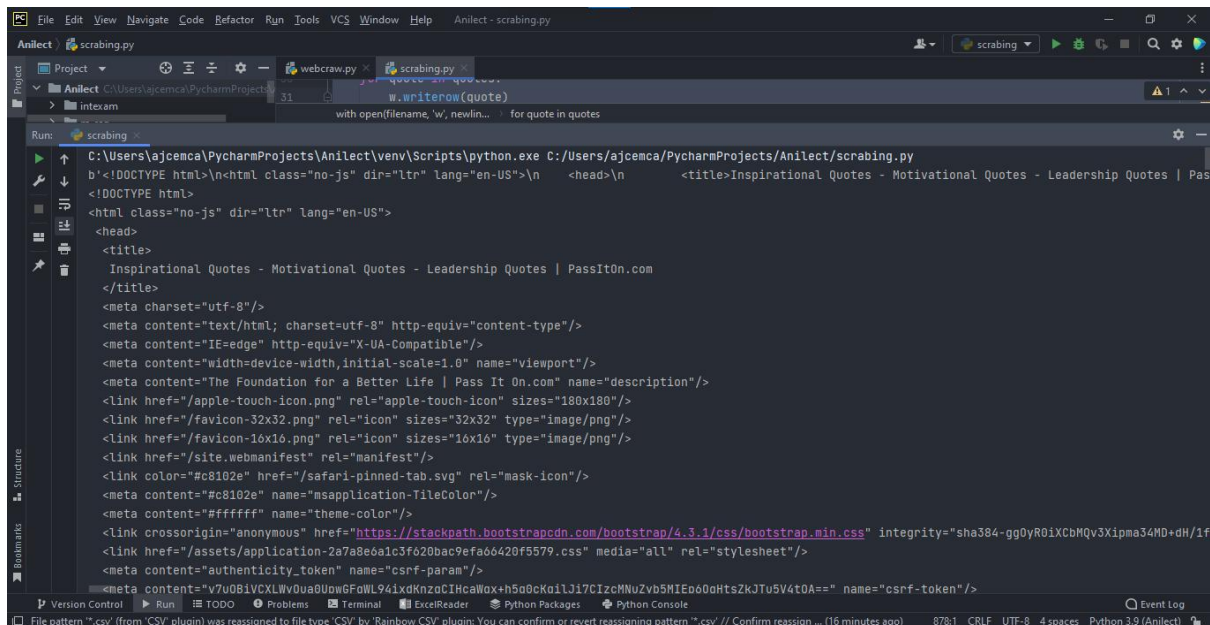
```
w = csv.DictWriter(f, ['theme', 'url', 'img', 'lines', 'author'])
```

```
w.writeheader()
```

```
for quote in quotes:
```

```
    w.writerow(quote)
```

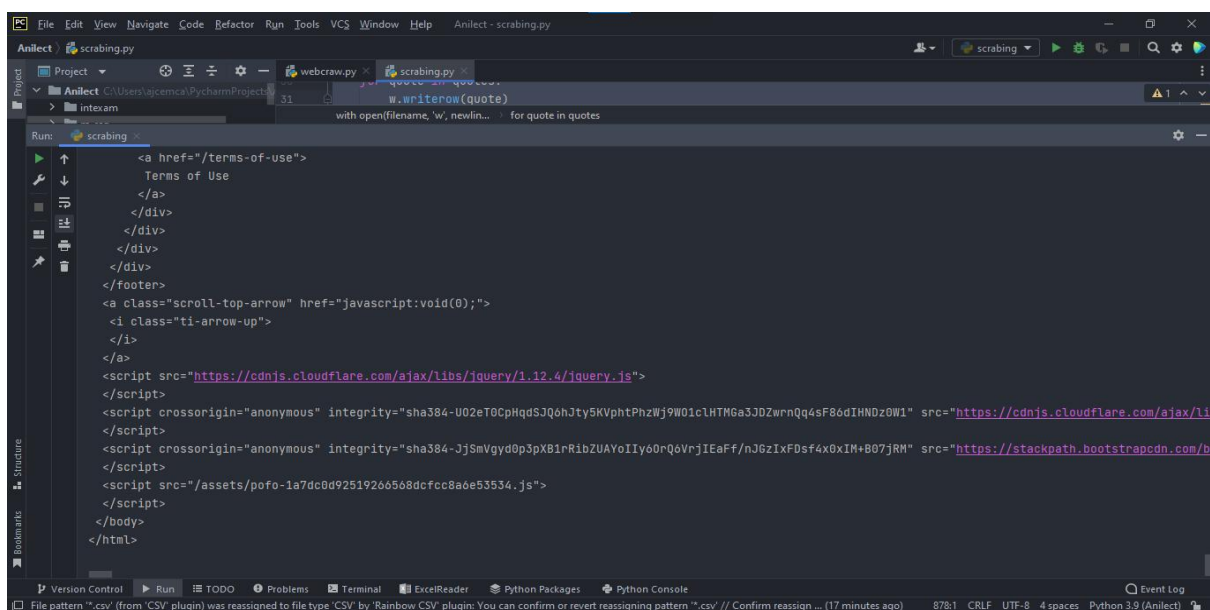
Output:



```

C:\Users\ajcemca\PycharmProjects\Anilect\venv\Scripts\python.exe C:/Users/ajcemca/PycharmProjects/Anilect/scrabing.py
b'<!DOCTYPE html>\n<html class="no-js" dir="ltr" lang="en-US">\n  <head>\n    <title>Inspirational Quotes - Motivational Quotes - Leadership Quotes | Pas
<!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-ggOyR01xCbMQv31xipma34MD+dH/1f
<link href="/assets/application-2a7a8e6a1c3f020bac9efa06420f5579.css" media="all" rel="stylesheet"/>
<meta content="authenticity_token" name="csrf-param"/>
<meta content="v7U0BiVCXLWV0ua8UomGFoWL94ixdKnz0CIHcaWox+h5o9cKoiLj17CIzcMNUZvb5MIEo60HtsZk3Ju5V4t0A==" name="csrf-token"/>

```



```

<a href="/terms-of-use">
  Terms of Use
</a>
</div>
</div>
</div>
</div>
<div>
<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-U02eT0CpHqds3Q6hJty5KvPhzWj9W01c1HTMga3JDZwrnQq4sF86dIHNDz0W1" src="https://cdnjs.cloudflare.com/ajax/li
</script>
<script crossorigin="anonymous" integrity="sha384-JjSmVgyd0p3pK81RribZUAyoIyVoR9Q6VrJIEeFf/nJGzIxFOsf4x0xIM+807jRM" src="https://stackpath.bootstrapcdn.com/b
</script>
<script src="/assets/pofo-1a7dc0d92519260568dcfcc8a0e53534.js">
</script>
</body>
</html>

```

```

File Edit View Navigate Code Refactor Run Tools VCS Window Help Anilnet - inspirational_quotes.csv
Project: inspirational_quotes.csv
1 theme, url, img, lines, author
2 LOVE, /inspirational-quotes/7444-where-there-is-love-there-is-life, https://assets.passiton.com/quotes/quote_artwork/7444/medium/20220215_tuesday_quote_alternate
3 LOVE, /inspirational-quotes/7439-at-the-touch-of-love-everyone-becomes-a-poet, https://assets.passiton.com/quotes/quote_artwork/7439/medium/20220214_monday_quote
4 FRIENDSHIP, /inspirational-quotes/8304-a-friend-may-be-waiting-behind-a-stranger-s-face, https://assets.passiton.com/quotes/quote_artwork/8304/medium/20220211_fri
5 FRIENDSHIP, /inspirational-quotes/3331-wherever-we-are-it-is-our-friends-that-make, https://assets.passiton.com/quotes/quote_artwork/3331/medium/20220210_thursda
6 FRIENDSHIP, /inspirational-quotes/8303-find-a-group-of-people-who-challenge-and, https://assets.passiton.com/quotes/quote_artwork/8303/medium/20220209_wednesday_g
7 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_tuesda
8 FRIENDSHIP, /inspirational-quotes/7435-there-are-good-ships-and-wood-ships-ships-that, https://assets.passiton.com/quotes/quote_artwork/7435/medium/20220207_monda
9 PERSISTENCE, /inspirational-quotes/6377-at-211-degrees-water-is-hot-at-212-degrees, https://assets.passiton.com/quotes/quote_artwork/6377/medium/20220204_friday_g
10 PERSISTENCE, /inspirational-quotes/8301-the-key-of-persistence-opens-all-doors-closed, https://assets.passiton.com/quotes/quote_artwork/8301/medium/20220203_thursd
11 PERSISTENCE, /inspirational-quotes/7918-you-keep-putting-one-foot-in-front-of-the, https://assets.passiton.com/quotes/quote_artwork/7918/medium/20220202_wednesday_g
12 PERSISTENCE, /inspirational-quotes/7919-to-persist-with-a-goal-you-must-treasure-the, https://assets.passiton.com/quotes/quote_artwork/7919/medium/20220201_tuesda
13 PERSISTENCE, /inspirational-quotes/8300-failure-cannot-cope-with-persistence, https://assets.passiton.com/quotes/quote_artwork/8300/medium/20220131_monday_quote_b
14 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_frida
15 INSPIRATION, /inspirational-quotes/8297-a-highly-developed-values-system-is-like-a, https://assets.passiton.com/quotes/quote_artwork/8297/medium/20210127_thursda
16 INSPIRATION, /inspirational-quotes/7066-just-don-t-give-up-t /inspirational-quotes/8297-a-highly-developed-values-system-is-like-a .com/quotes/quote_artwork/7066/medium/20210126_wedne
17 INSPIRATION, /inspirational-quotes/8296-when-we-strive-to-be m/quotes/quote_artwork/8296/medium/20210125_tuesda
18 INSPIRATION, /inspirational-quotes/8299-the-most-important-t .com/quotes/quote_artwork/8299/medium/20210124_monday
19 OVERCOMING, /inspirational-quotes/6828-bad-things-do-happen- quotes/quote_artwork/6828/medium/20220121_friday_qu
20 OVERCOMING, /inspirational-quotes/8294-show-me-someone-who-has-done-something, https://assets.passiton.com/quotes/quote_artwork/8294/medium/20220120_thursday_quot
21 OVERCOMING, /inspirational-quotes/6137-its-not-the-load-that-breaks-you-down-its-the, https://assets.passiton.com/quotes/quote_artwork/6137/medium/20220119_wednesd
22 OVERCOMING, /inspirational-quotes/6805-getting-over-a-painful-experience-is-much-like, https://assets.passiton.com/quotes/quote_artwork/6805/medium/20220118_tuesd
23 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_monday
24 CREATIVITY, /inspirational-quotes/5577-the-creative-is-the-place-where-no-one-else-has, https://assets.passiton.com/quotes/quote_artwork/5577/medium/20220114_frida
25 CREATIVITY, /inspirational-quotes/7345-creativity-is-allowing-yourself-to-make, https://assets.passiton.com/quotes/quote_artwork/7345/medium/20220113_thursday_quo
26 CREATIVITY, /inspirational-quotes/7487-creativity-requires-the-courage-to-let-go-of, https://assets.passiton.com/quotes/quote_artwork/7487/medium/20220112_wednesd

Header: url
Index: 2

```

File Pattern '*.csv' (from 'CSV' plugin) was reassigned to file type 'CSV' by 'Rainbow CSV' plugin: You can confirm or revert reassigning pattern '*.csv' // Confirm reassign... (23 minutes ago) 1:1 CRLF windows-1252 Tab Python 3.9 (Anilnet) Event Log

Result: The program has been executed and output verified

Program no: 17**Date: 16-02-2022****Aim:** Program for Natural 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='Hello there, Welcome to DS Lab Record', WordsToCombine=3)
```

```
print(x)
```

Output:

```
C:\Users\ajcemca\PycharmProjects\Anilect\venv\Scripts\python.exe C:/Users/ajcemca/PycharmProjects/Anilect/ngram.py
[['Hello', 'there,', 'Welcome'], ['there,', 'Welcome', 'to'], ['Welcome', 'to', 'DS'], ['to', 'DS', 'Lab'], ['DS', 'Lab', 'Record']]

Process finished with exit code 0
```

Result: The program has been executed and output verified

Program no: 18**Date: 23-02-2022**

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

Program:

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

samplText = 'This is a very good book to study'

GRAMS = ngrams(sequence=nltk.word_tokenize(samplText), n=2)
for grams in GRAMS:
    print(grams)
```

Output:

```
C:\Users\ajcemca\PycharmProjects\Anilect\venv\Scripts\python.exe C:/Users/ajcemca/PycharmProjects/Anilect/ngramnltk.py
showing info https://raw.githubusercontent.com/nltk/nltk\_data/gh-pages/index.xml
('This', 'is')
('is', 'a')
('a', 'very')
('very', 'good')
('good', 'book')
('book', 'to')
('to', 'study')

Process finished with exit code 0
```

Result: The program has been executed and output verified

Program no: 19

Date: 23-02-2022

Aim: Program for Natural 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."

# sent_tokenize is one of instances of
# PunktSentenceTokenizer from the nltk.tokenize.punkt module

tokenized = sent_tokenize(txt)

for i in tokenized:
    # Word tokenizers is used to find the words
    # and punctuation in a string
    wordsList = nltk.word_tokenize(i)

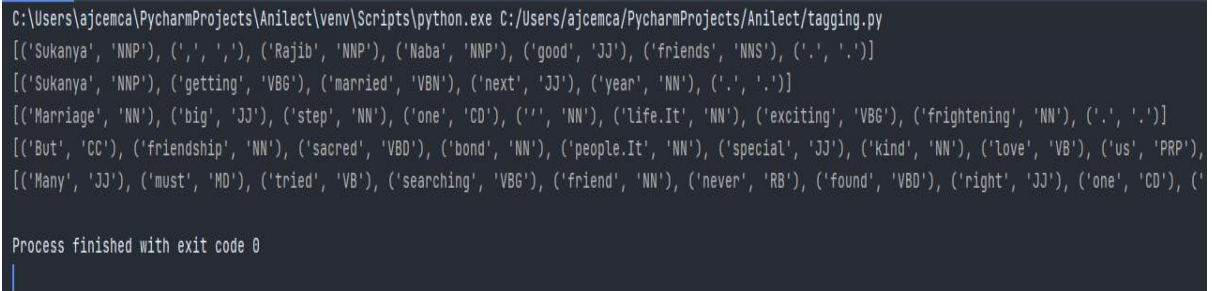
    # removing stop words from wordList
    wordsList = [w for w in wordsList if not w in stop_words]

    # Using a Tagger. Which is part-of-speech
```

```
# tagger or POS-tagger.
tagged = nltk.pos_tag(wordsList)

print(tagged)
```

Output:



```
C:\Users\ajcemca\PycharmProjects\Anilect\venv\Scripts\python.exe C:/Users/ajcemca/PycharmProjects/Anilect/tagging.py
[('Sukanya', 'NNP'), ('Rajib', 'NNP'), ('Naba', 'NNP'), ('good', 'JJ'), ('friends', 'NNS'), ('.', '.')]
[('Sukanya', 'NNP'), ('getting', 'VBG'), ('married', 'VBN'), ('next', 'JJ'), ('year', 'NN'), ('.', '.')]
[('Marriage', 'NN'), ('big', 'JJ'), ('step', 'NN'), ('one', 'CD'), ('life', 'NN'), ('exciting', 'VBG'), ('frightening', 'NN'), ('.', '.')]
[('But', 'CC'), ('friendship', 'NN'), ('sacred', 'VBD'), ('bond', 'NN'), ('people', 'NN'), ('special', 'JJ'), ('kind', 'NN'), ('love', 'VB'), ('us', 'PRP'), ('.', '.')]
[('Many', 'JJ'), ('must', 'MD'), ('tried', 'VB'), ('searching', 'VBG'), ('friend', 'NN'), ('never', 'RB'), ('found', 'VBD'), ('right', 'JJ'), ('one', 'CD'), ('.', '.')]

Process finished with exit code 0
```

Result: The program has been executed and output verified

Program no: 20

Date: 23-02-2022

Aim: Write a python program for natural program language processing with chunking

Program:

```
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 = "NP: {<DT>?<JJ>*<NN>}"

chunkParser = nltk.RegexpParser(grammer)

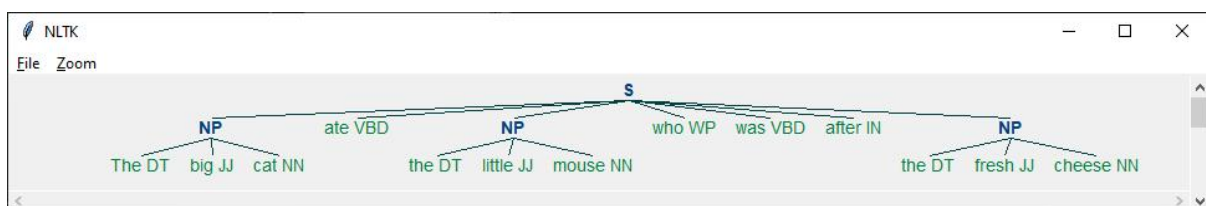
chunked = chunkParser.parse(new_tag)

print(chunked)

chunked.draw()
```

Output:

```
C:\Users\ajcemca\PycharmProjects\Anilect\venv\Scripts\python.exe C:/Users/ajcemca/PycharmProjects/Anilect/chunk.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', 'IN'), ('the', 'DT'), ('fresh', 'JJ'), ('cheese', 'NN')]
(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))
```



Result: The program has been executed and output verified

Program no: 21

Date: 23-02-2022

Aim: Write a python program for natural program language processing with chunking

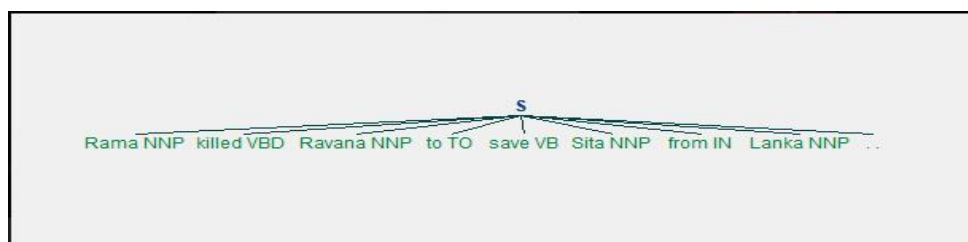
Program:

```
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)
    # 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
  ./.)
(S
  The/DT
  Legend/NN
  of/IN
  the/DT
  Ramayan/NNP
  is/VBZ
  the/DT
  most/RBS
  popular/JJ
  Indian/JJ
  epic/NN
  ./.)
(S
```



Result: The program has been executed and output verified